ISaGRAF

Wersja 3.4

PODRĘCZNIK UŻYTKOWNIKA

CJ INTERNATIONAL

Informacje zawarte w niniejszym dokumencie mogą zostać zmienione bez powiadomienia i nie są przedmiotem zobowiązania ze strony CJ International. Oprogramowanie opisane w niniejszym dokumencie, w tym także informacje znajdujące się w którejkolwiek z baz danych, dostarczane jest na podstawie umowy licencyjnej lub umowy o nieujawnianiu danych i może być wykorzystywane lub kopiowane jedynie zgodnie z warunkami umowy. Kopiowanie tego oprogramowania, za wyjątkiem szczególnych przypadków dopuszczonych w umowie licencyjnej lub umowie o nieujawnianiu, jest niezgodne z prawem. Żadna część niniejszego podręcznika nie może być reprodukowana w jakimkolwiek celu, w jakikolwiek sposób czy też jakimikolwiek środkami elektronicznymi czy mechanicznymi, obejmującymi wykonywanie fotokopii i nagrań, bez wyraźnej pisemnej zgody CJ International.

© 2001 CJ International. All rights reserved.
Printed in France by CJ International.
3 Rue Hector Berlioz
F-38600 FONTAINE
Phone: 33 (0)4 76 26 87 30
Fax: 33 (0)4 76 26 87 39

ISaGRAF jest zastrzeżonym znakiem towarowym firmy CJ International.
MS-DOS jest zastrzeżonym znakiem towarowym firmy Microsoft Corporation.
Windows jest zastrzeżonym znakiem towarowym firmy Microsoft Corporation.
Windows NT jest zastrzeżonym znakiem towarowym firmy Microsoft Corporation.
OS-9 i ULTRA-C są zastrzeżonymi znakami towarowymi firmy Microware Corporation.
VxWorks i Tornado są zastrzeżonymi znakami towarowymi firmy Wind River Systems, Inc.

Wszystkie pozostałe nazwy firmowe lub nazwy wyrobów są znakami towarowymi lub zastrzeżonymi znakami towarowymi ich właścicieli.

Spis treści

A. PODRĘCZNIK UŻYTKOWNIKA A-11

A.1 Uruchomienie A-12
A.1.1 INSTALACJA ISAGRAF A-12
A.1.2 KORZYSTANIE Z INFORMACJI BEZPOŚREDNIEJ A-15
A.1.3 PRZYKŁADOWA APLIKACJA A-15

A.2 Zarządzanie projektami A-19

A.3 Zarządzanie programami A-23
A.3.1 KOMPONENTY PROJEKTU A-23
A.3.2 PRACA Z PROGRAMAMI A-25
A.3.3 URUCHAMIANIE NARZĘDZI GENEROWANIA KODU A-29
A.3.4 INNE NARZĘDZIA ISAGRAF A-30
A.3.5 DODAWANIE KOMEND DO MENU NARZĘDZIA A-31
A.3.6 SYMULACJA I DEBAGOWANIE APLIKACJI A-32

A.4 Korzystanie z edytora SFC A-35
A.4.1 GŁÓWNE POJĘCIA JĘZYKA SFC A-35
A.4.2 WPROWADZANIE GRAFU SFC A-38
A.4.3 OPRACOWYWANIE ISTNIEJĄCEGO GRAFU SFC A-40
A.4.4 WPROWADZANIE PROGRAMU POZIOMU 2 A-41
A.4.5 KORZYSTANIE Z GALERII SFC A-45

A.5 Korzystanie z edytora Flow Chart A-46
A.5.1 PODSTAWY JĘZYKA FC A-46
A.5.2 WPROWADZENIE GRAFU PRZEPŁYWOWEGO A-47
A.5.3 PRACA NA ISTNIEJĄCYM GRAFIE A-50
A.5.4 WPROWADZANIE PROGRAMÓW POZIOMU 2 A-51
A.5.5 PROGRAMOWANIE W LD A-52
A.5.6 OPCJE WYŚWIETLANIA A-53

A.6 Korzystanie z edytora LD A-55

A.6.1 PODSTAWY JĘZYKA LD A-55
A.6.2 WPROWADZANIE DIAGRAMU LD A-58
A.6.3 PRACA NA ISTNIEJĄCYM DIAGRAMIE A-61
A.6.4 OPCJE WYŚWIETLANIA A-62

A.7 Korzystanie z edytora FBD/LD A-64
A.7.1 PODSTAWY JĘZYKÓW FBD/LD A-64
A.7.2 WPROWADZANIE DIAGRAMU FBD A-67
A.7.3 OPRACOWYWANIE ISTNIEJĄCEGO DIAGRAMU A-69
A.7.4 OPCJE WYŚWIETLANIA A-71
A.7.5 STYLE I ŚLEDZENIE MODYFIKACJI A-71

A.8 Korzystanie z edytora tekstowego A-73
A.8.1 KOMENDY EDYCYJNE A-73
A.8.2 OPCJE A-74

A.9 Więcej informacji o edytorach programowych A-75
A.9.1 WYWOŁYWANIE INNYCH NARZĘDZI ISAGRAF A-75
A.9.2 PARAMETRY PROGRAMU A-75
A.9.3 INNE KOMENDY MENU "PLIK" A-76
A.9.4 AKTUALIZACJA DZIENNIKA PROGRAMU A-77
A.9.5 WYBIERANIE ZMIENNEJ ZE SŁOWNIKA A-78
A.9.6 OKNO INFORMACJI WYJŚCIOWYCH A-79

A.10 Korzystanie z edytora słownika A-80
A.10.1 OKNO GŁÓWNE SŁOWNIKA A-82
A.10.2 ZARZĄDZANIE ZMIENNYMI A-83
A.10.3 OPIS OBIEKTÓW A-85
A.10.4 SZYBKIE DEKLAROWANIE A-86
A.10.5 MAPA ADRESÓW MODBUS SCADA A-87
A.10.6 WYMIANA INFORMACJI Z INNYMI APLIKACJAMI A-88

A.11 Korzystanie z edytora konfiguracji We/Wy A-93
A.11.1 DEFINIOWANIE KART WE/WY A-94
A.11.2 KONFIGURACJA PARAMETRÓW KARTY A-95
A.11.3 PODŁĄCZANIE KANAŁÓW WE/WY A-96
A.11.4 ZMIENNE O REPREZENTACJI BEZPOŚREDNIEJ A-96
A.11.5 NUMEROWANIE A-97
A.11.6 USTAWIANIE ZABEZPIECZEŃ INDYWIDUALNYCH A-97

A.12 Tworzenie tablic konwersji A-99
A.12.1 KOMENDY PODSTAWOWE A-99
A.12.2 WPROWADZANIE PUNKTÓW TABLICY A-100
A.12.3 ZASADY I LIMITY A-101

A.13 Korzystanie z generatora kodu A-102
A.13.1 KOMENDY GŁÓWNE A-102
A.13.2 OPCJE KOMPILATORA A-103
A.13.3 TWORZENIE KODU ŹRÓDŁOWEGO C A-106
A.13.4 PRZEGLĄDANIE INFORMACJI A-106
A.13.5 DEFINIOWANIE ZASOBÓW A-107

A.14 Odsyłacze A-113

A.15 Korzystanie z debagera graficznego A-115
A.15.1 OKNO DEBAGERA A-115
A.15.2 KONTROLOWANIE PRACY APLIKACJI A-116
A.15.3 OPCJE A-118
A.15.4 KOMENDY „WPISZ” A-119
A.15.5 MODYFIKACJA NATYCHMIASTOWA A-121
A.15.6 MECHANIZM WYMIANY DDE A-124

A.16 Zmienne śledzone A-126

A.17 Debagowanie programów ST i IL A-128

A.18 Synoptyka A-129
A.18.1 BUDOWANIE ZOBRAZOWANIA A-129
A.18.2 WIDOK A-132
A.18.3 DEFINIOWANIE STYLU ELEMENTU A-132
A.18.4 KOMENDY MENU "PLIK" A-133
A.18.5 UWAGI DLA UŻYTKOWNIKÓW ISAGRAF V3.2 A-134

A.19 Pobieranie aplikacji A-135
A.19.1 POBIERANIE PROJEKTU A-135
A.19.2 USTAWIENIA PARAMETRÓW KOMUNIKACJI A-136
A.19.3 PRZYGOTOWYWANIE PROJEKTU DO POBRANIA A-136
A.19.4 PRZECHOWYWANIE SPAKOWANEGO KODU ŹRÓDŁOWEGO W OPROGRAMOWANIU WBUDOWANYM A-137

A.19.5 WYMAGANIA DOTYCZĄCE PAMIĘCI OPROGRAMOWANIA WBUDOWANEGO A-137
A.19.6 INFORMACJE DOTYCZĄCE POBIERANEGO PROJEKTU A-137
A.19.7 PROBLEMY KOMPATYBILNOŚCI A-138

A.20 Korzystanie z narzędzia diagnostycznego A-139

A.21 Korzystanie z symulatora ISaGRAF A-140
A.21.1 POWIĄZANIA Z DEBAGEREM A-140
A.21.2 SYMULACJA WE/WY A-140
A.21.3 KOMPONENTY BIBLIOTEKI A-141
A.21.4 OPCJE A-142
A.21.5 ZAPISYWANIE I PRZYWRACANIE STANÓW WEJŚCIOWYCH A-142
A.21.6 PROFILER CYKLU A-143
A.21.7 SKRYPTY SYMULACYJNE A-144

A.22 Korzystanie z Menedżera Biblioteki A-152
A.22.1 ZARZĄDZANIE ELEMENTAMI BIBLIOTEKI A-152
A.22.2 KONFIGURACJE WE/WY A-155
A.22.3 ZESPOŁY WE/WY A-156
A.22.4 KARTY WE/WY A-156
A.22.5 FUNKCJE I BLOKI NAPISANE W JĘZYKACH IEC A-158
A.22.6 FUNKCJE I BLOKI FUNKCYJNE "C" A-160
A.22.7 FUNKCJE KONWERSJI A-160

A.23 Używanie programu archiwizacji A-162
A.23.1 WYWOŁYWANIE MENEDŻERA ARCHIWIZACJI A-162
A.23.2 OPCJE A-163
A.23.3 KOPIA ZAPASOWA I PRZYWRACANIE DANYCH Z KOPII A-163
A.23.4 PLIKI ARCHIWALNE A-164

A.24 Drukowanie całej dokumentacji A-165
A.24.1 DOSTOSOWYWANIE SPISU TREŚCI A-165
A.24.2 OPCJE A-167

A.25 Ochrona hasłem A-169

A.26 Zaawansowane techniki programowania A-173
A.26.1 BLIŻSZE INFORMACJE NA TEMAT NARZĘDZI ISAGRAF A-173
A.26.2 ZABLOKOWANE WE/WY I WIRTUALNE WE/WY A-173

A.26.3 KONTROLA POPRAWNOŚCI POŁĄCZENIA PC-PLC A-176
A.26.4 KATALOGI ISAGRAF A-177
A.26.5 SYMBOLE APLIKACJI A-179
A.26.6 OGRANICZENIA PAKIETU ISAGRAF, WERSJA "LARGE" (WDL) A-183

B. JĘZYKI PROGRAMOWANIA B-1

B.1 Architektura projektu B-2
B.1.1 PROGRAMY B-2
B.1.2 OPERACJE CYKLICZNE I SEKWENCYJNE B-2
B.1.3 PROGRAMY POTOMNE SFC I FC B-3
B.1.4 FUNKCJE I PODPROGRAMY B-4
B.1.5 BLOKI FUNKCYJNE B-5
B.1.6 JĘZYK OPISU B-6
B.1.7 SKŁADOWE CYKLU STEROWANIA B-6

B.2 Obiekty wspólne B-8
B.2.1 TYPY PODSTAWOWE B-8
B.2.2 WYRAŻENIA STAŁE B-8
B.2.3 ZMIENNE B-10
B.2.4 KOMENTARZE B-13
B.2.5 ZDEFINIOWANE SŁOWA B-14

B.3 Język SFC B-15
B.3.1 GŁÓWNY FORMAT SCHEMATU SFC B-15
B.3.2 PODSTAWOWE KOMPONENTY SFC B-15
B.3.3 ROZBIEŻNOŚCI I ZBIEŻNOŚCI B-18
B.3.4 ETAPY TYPU MAKRO B-21
B.3.5 OPERACJE W RAMACH KROKÓW B-22
B.3.6 WARUNKI DOŁĄCZONE DO PRZEJŚĆ B-28
B.3.7 ZASADY DYNAMICZNE SFC B-30
B.3.8 HIERARCHIA PROGRAMU SFC B-30

B.4 Język FC B-32
B.4.1 KOMPONENTY FC B-32
B.4.2 PRZYKŁADY STRUKTUR ZŁOŻONYCH FC B-36
B.4.3 ZACHOWANIE DYNAMICZNE FC B-37
B.4.4 KONTROLA FC B-37

B.5 Język FBD B-38

B.5.1 GŁÓWNY FORMAT SCHEMATU FBD B-38
B.5.2 INSTRUKCJA RETURN B-39
B.5.3 SKOKI I ETYKIETY B-39
B.5.4 NEGACJA BINARNA B-40
B.5.5 WYWOŁYWANIE FUNKCJI LUB BLOKU FUNKCYJNEGO Z POZIOMU FBD B-41

B.6 Język LD B-42
B.6.1 SZYNY ZASILAJĄCE I LINIE ŁĄCZĄCE B-42
B.6.2 POŁĄCZENIE WIELOKROTNE B-43
B.6.3 PODSTAWOWE STYKI I CEWKI LD B-44
B.6.4 INSTRUKCJA RETURN B-50
B.6.5 SKOKI I ETYKIETY B-50
B.6.6 BLOKI W LD B-51

B.7 Język ST B-53
B.7.1 GŁÓWNA SKŁADNIA ST B-53
B.7.2 WYRAŻENIE I NAWIASY B-53
B.7.3 WYWOŁANIA FUNKCJI I BLOKÓW FUNKCYJNYCH B-54
B.7.4 OPERATORY BINARNE SPECYFICZNE DLA ST B-56
B.7.5 PODSTAWOWE INSTRUKCJE ST B-57
B.7.6 ROZSZERZENIA ST B-63

B.8 Język IL B-69
B.8.1 GŁÓWNA SKŁADNIA IL B-69
B.8.2 OPERATORY IL B-70

B.9 Standardowe operatory, bloki funkcyjne i funkcje B-72
B.9.1 STANDARDOWE OPERATORY B-72
B.9.2 STANDARDOWE BLOKI FUNKCYJNE B-94
B.9.3 STANDARDOWE FUNKCJE B-113

C. PODRĘCZNIK UŻYTKOWNIKA OPROGRAMOWANIA WBUDOWANEGO C-1

C.1 Wprowadzenie C-2

C.2 Instalacja C-3

C.3 Uruchomienie oprogramowania wbudowanego ISaGRAF pod DOS C-4
C.3.1 WYKONYWANIE ISAGRAF: ISA.EXE C-4
C.3.2 WŁASNOŚCI SPECJALNE C-5

C.4 Uruchomienie oprogramowania wbudowanego ISaGRAF pod OS9 C-9
C.4.1 WYKONYWANIE JEDNOZADANIOWEJ IMPLEMENTACJI ISAGRAF: ISA C-9
C.4.2 WYKONYWANIE WIELOZADANIOWEJ IMPLEMENTACJI ISAGRAF: ISAKER, ISATST, ISANET C-10
C.4.3 WŁASNOŚCI SPECJALNE C-14

C.5 Uruchomienie oprogramowania wbudowanego ISaGRAF pod VxWorks C-19
C.5.1 MENEDŻER ZASOBÓW SYSTEMU: ISASSR.O C-19
C.5.2 TYPOWE WŁASNOŚCI ISA.O, ISAKERSE.O I ISAKERET.O C-19
C.5.3 WYKONYWANIE JEDNOZADANIOWEJ IMPLEMENTACJI ISAGRAF: ISA.O C-20
C.5.4 WYKONYWANIE WIELOZADANIOWEJ IMPLEMENTACJI ISAGRAF: ISAKERSE.O I ISAKERET.O C-22
C.5.5 WŁASNOŚCI SPECJALNE C-26

C.6 Uruchomienie oprogramowania wbudowanego ISaGRAF pod Windows NT C-31
C.6.1 WYKONYWANIE ISAGRAF C-31
C.6.2 OGÓLNE INFORMACJE O OPCJACH C-31
C.6.3 WŁASNOŚCI SPECJALNE C-36
C.6.4 INTERFEJS UŻYTKOWNIKA C-41

C.7 Programowanie w języku "C" C-47
C.7.1 WPROWADZENIE C-47
C.7.2 FUNKCJE KONWERSJI "C" C-48
C.7.3 FUNKCJE "C" C-53
C.7.4 BLOKI FUNKCYJNE "C" C-60
C.7.5 TECHNIKI KOMPILOWANIA I ŁĄCZENIA C-74

C.8 Łącze Modbus C-80
C.8.1 SIEĆ I PROTOKÓŁ MODBUS C-80
C.8.2 IMPLEMENTACJA ISAGRAF C-81

C.9 Postępowanie w przypadku awarii zasilania C-87
C.9.1 INFORMACJE PODSTAWOWE C-87
C.9.2 KOPIE ZAPASOWE ZMIENNYCH APLIKACJI C-88
C.9.3 KOPIA ZAPASOWA DANYCH O STANIE PROGRAMU C-91

C.10 Załącznik: Wykaz błędów i ich opisy C-93

D. GLOSARIUSZ D-1

E. INDEKS E-1

A. Podręcznik użytkownika

A.1 Uruchomienie

Niniejszy rozdział obejmuje instalację pakietu ISaGRAF. Zawiera on również krótki przykład aplikacji ISaGRAF, która umożliwia użytkownikowi poznanie głównych cech pakietu ISaGRAF i pozwala na natychmiastowe jego stosowanie.

A.1.1 Instalacja ISaGRAF

Niniejszy rozdział obejmuje instalację pakietu ISaGRAF oraz sposób konfigurowania komputera do opracowywania aplikacji.

Wymagania sprzętowe i programowe

Pakiet ISaGRAF można zainstalować na dowolnym komputerze, spełniającym wymagania minimalne dla systemu Windows wersja 3.1. Jednak dla potrzeb opracowywania aplikacji zalecana jest konfiguracja:

- Komputer osobisty z procesorem 80486 lub wyższym (zaleca się procesor Pentium)
- 8 megabajtów pamięci konwencjonalnej i rozszerzonej (zaleca się 16 megabajtów)
- Jedna stacja dysków 3,5 cala (1,44 megabajta)
- Jeden dysk twardy z minimum 20 megabajtami dostępnego obszaru
- Karta graficzna VGA lub SVGA oraz kompatybilny monitor
- Mysz (wymagana do graficznych narzędzi programisty)
- Port równoległy LPT1 (wymagany do klucza zabezpieczającego)

Przed instalacją pakietu ISaGRAF w systemie powinno znajdować się już następujące oprogramowanie:

- Windows wersja 3.1 działająca w trybie rozszerzonym 386
- Windows 95
- Windows NT wersja 3.51 lub 4.00

Korzystanie z programu instalacyjnego

Pakiet ISaGRAF instaluje się przy użyciu programu instalacyjnego ISaGRAF o nazwie INSTALL. Program ten kopiuje oprogramowanie ISaGRAF z dysku ISaGRAF CD-ROM lub z dyskietek na dysk twardy komputera użytkownika. INSTALL dodaje również do okna Menedżera Programów grupę "ISaGRAF" i tworzy plik inicjalizacyjny o nazwie "ISA.ini" w instalowanym podkatalogu **EXE**.

INSTALL jest programem systemu Windows, który musi być uruchamiany z poziomu Menedżera Programów Windows lub przy użyciu komendy Uruchom z menu Start w Windows 95. Aby zainstalować ISaGRAF należy wykonać następujące czynności:

- Włożyć CD-ROM lub dyskietkę #1 ISaGRAF do odpowiedniego napędu

• Z poziomu Menedżera Programów lub z menu Start uruchomić "SETUP.EXE" w folderze głównym CD-ROM lub "A:\INSTALL.EXE" w przypadku dyskietek.
• Wykonać instalacje postępując zgodnie z instrukcjami. Zaleca się instalację pakietu ISaGRAF w nowym katalogu, aby uniknąć pomieszania plików z plikami innych wersji ISaGRAF.

INSTALL zapyta czy wymagane są następujące składniki:

• programy wykonywalne ISaGRAF
• Pliki informacji i pomocy bezpośredniej
• Standardowe biblioteki ISaGRAF
• Przykładowe aplikacje ISaGRAF

Gdy ISaGRAF jest instalowany po raz pierwszy, wysoce zalecane jest zainstalowanie wszystkich składników. Dalsze składniki można jednak dodać później, przy ponownej instalacji pakietu ISaGRAF.

Domyślną nazwą głównego katalogu ISaGRAF jest "\ISAWIN". Pozwala to na łatwą instalację ISaGRAF dla Windows na tym samym dysku, co wersja ISaGRAF dla MS-DOS. Dalsze informacje na temat struktury dyskowej ISaGRAF podano w punkcie “Katalogi ISaGRAF" w rozdziale "Techniki zaawansowane". Po skopiowaniu wszystkich tych plików do okna Menedżera Programów dodawana jest grupa:

Znajdują się tu główne ikony ISaGRAF:

Projekty:	Zarządzanie projektami
Biblioteki:	Zarządzanie bibliotekami
Książka:	Informacje na temat ISaGRAF (forma elektroniczna)
Diagnostyka:	Narzędzie diagnostyczne dla użytkownika końcowego
Czytajto:	Informacje na temat nowej wersji ISaGRAF
Raport:	Standardowy formularz zgłoszenia błędów

W razie napotkania problemów, należy skorzystać ze standardowego formularza zgłoszenia błędów. Należy go otworzyć, podać potrzebne dane i używając komendy menu Plik/Zapisz jako... zapisać go w pliku o wybranej nazwie. Następnie, przy użyciu faksu lub poczty elektronicznej, należy przesłać ten plik do firmy CJ International.

Uaktualnianie plików systemowych

Po zakończeniu instalacji, przed ponownym uruchomieniem komputera, należy uaktualnić plik CONFIG.SYS. Nazwy ścieżki dostępu do katalogu ISaGRAF nie trzeba umieszczać w zmiennej PATH.
ISaGRAF nie korzysta z żadnych zmiennych środowiska MS-DOS. Jednakże do pliku CONFIG.SYS można dodać następujące zdania:

```
files=20
buffers=20
```

Do komunikowania się ze sterownikiem PLC z oprogramowaniem wbudowanym pakiet ISaGRAF wykorzystuje port szeregowy. Domyślnym portem szeregowym dla ISaGRAF jest COM1. Jeśli mysz również korzysta z portu szeregowego, należy wybrać dla niej port COM2 tak, aby domyślne ustawienie portu COM1 było takie samo dla wszystkich nowych aplikacji ISaGRAF.

Po uaktualnieniu pliku CONFIG.SYS konieczne jest ponowne uruchomienie komputera w celu zainstalowania zmian.

⇒ **Ważne dla użytkowników Windows NT:**
Gdy pakiet ISaGRAF jest używany pod systemem Windows NT 3.51 lub 4.00, w sekcji [WS001] pliku ISA.ini znajdującego się w katalogu \ISAWIN\EXE, należy dodać następujące wiersze:

```
[WS001]
NT=1
Isa=C:\ISAWIN
IsaExe=C:\ISAWIN\EXE
IsaApl=C:\ISAWIN\APL1
IsaTmp=C:\ISAWIN\TMP
```

Jest to absolutnie niezbędne do komunikacji poprzez port szeregowy.

Klucz zabezpieczający

Oprogramowanie ISaGRAF jest zabezpieczone przed nielegalnym kopiowaniem za pomocą klucza sprzętowego. Jednakże większość funkcji pakietu ISaGRAF jest dostępna bez zainstalowanego klucza. Klucz zabezpieczający określa również wersję pakietu ISaGRAF oraz maksymalną wielkość opracowywanych aplikacji. Kiedy klucz nie jest włożony do gniazda lub jest niewłaściwie podłączony, niektóre z funkcji pakietu ISaGRAF nie działają. Takie zachowanie jest NORMALNE. Aby upewnić się, że klucz jest podłączony właściwie, należy wybrać pozycję "**Informacje...**" z menu "**Pomoc**" w którymkolwiek z okien ISaGRAF. Zostanie wyświetlona dostępna wersja pakietu ISaGRAF.

Klucz można podłączyć do dowolnego portu równoległego w komputerze. Jeśli w komputerze jest więcej niż jeden port równoległy, lepiej jest podłączyć klucz i drukarkę do różnych portów. W przypadku niektórych konfiguracji komputer /drukarka klucz może nie być wykrywany, gdy na jego wyjściu podłączona jest drukarka w trybie "oddziel" (“off-line”). W takich wypadkach należy odłączyć drukarkę lub włączyć w tryb "przydziel" ("on-line") i ponownie uruchomić pakiet ISaGRAF.

Proszę zwrócić uwagę, że do pakietu **ISaGRAF-32** nie jest wymagany żaden klucz.

⇒ **Ważne dla użytkowników Windows NT:**
Aby klucz zabezpieczający był wykrywany w systemach Windows NT trzeba zainstalować sterownik Sentinel/Rainbow™. Dostarczono go na osobnej dyskietce.

A.1.2 Korzystanie z informacji (forma elektroniczna)

Wraz z pakietem ISaGRAF są instalowane pliki zawierające informacje dotyczące następujących tematów:
- Informacje na temat języka ISaGRAF
- Kompletny podręcznik użytkownika (dla każdego narzędzia ISaGRAF)
- Opisy techniczne dotyczące elementów znajdujących się w bibliotekach

Informacje (forma elektroniczna) są wyświetlane po wybraniu odpowiedniej opcji w menu "**Pomoc**" z dowolnego okna ISaGRAF.

A.1.3 Przykładowa aplikacja

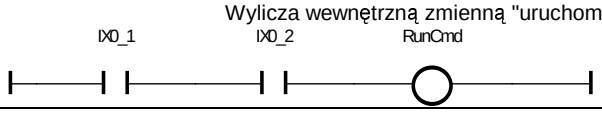
W tym rozdziale wyjaśniono krok po kroku wszystkie podstawowe operacje konieczne do wykonania, zaprojektowania, wygenerowania i przetestowania krótkiej, lecz kompletnej, wielojęzycznej aplikacji.

Poniżej znajduje się pełny opis tej aplikacji, łączącej odwzorowanie LD i SFC:

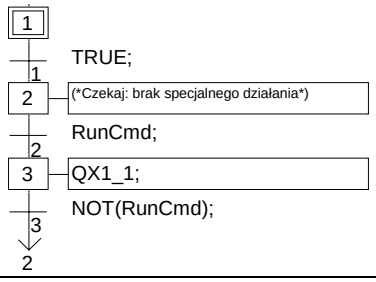
Zmienne binarne:

IX0_1, IX0_2:	zmienne wejściowe sterowania procesem
RunCmd:	wewnętrzna zmienna "uruchom/stop"
QX1_1:	zmienna wyjściowa: status procesu

Wypracowanie sterowania: Cykliczna sekcja początkowa – język LD
Wylicza wewnętrzną zmienną "uruchom/stop"



Program RunStop: Sekcja sekwencyjna – język SFC
Steruje procesem



Start ***Uruchomienie pakietu ISaGRAF***

Aby uruchomić pakiet ISaGRAF, należy wybrać komendę “Projekty" z grupy "ISaGRAF" w menu Start systemu Windows.

Tworzenie projektu

Utwórz projekt (o nazwie "RunStop") używając komendy "Nowy" w menu "Plik" lub przycisku Nowy. W otwartym oknie dialogowym:

Wpisz nazwę projektu: "**RunStop**"
Wybierz konfigurację We/Wy: "**Sim_Boo**"
Naciśnij przycisk "**OK**".
W ten sposób projekt został utworzony.

Otwieranie projektu

Programy projektu definiowane są poprzez otwarcie okna zarządzania programami ISaGRAF. Użyj komendy "Otwórz" z okna zarządzania projektami lub dwukrotnie kliknij nazwę projektu, albo użyj przycisku "Otwórz".

Tworzenie programów

Okno Zarządzanie Programami jest teraz otwarte i puste (brak zdefiniowanych programów). Pierwszy program tworzy się przy użyciu komendy "Nowy" w menu "Plik" lub przycisku Nowy. W otwartym oknie dialogowym:

Wpisz nazwę programu: "**Command**".
Wybierz język " **LD**".
Wybierz sekcję "**Początek**".
Naciśnij przycisk "**OK**" aby utworzyć program.

Ta sama operacja musi zostać powtórzona dla drugiego programu:
Użyj komendy "Nowy" w menu "Plik" lub przycisku Nowy. W otwartym oknie dialogowym:

Wpisz nazwę programu: "**RunStop**".
Wybierz język "**SFC**".
Wybierz sekcję "**Sekwencyjnie**".
Naciśnij przycisk "**OK**" aby utworzyć program.

Programy zostały utworzone. Pojawią się one w oknie Programy.

Deklarowanie zmiennych

Przed wprowadzeniem programów musi zostać zadeklarowana zmienna wewnętrzna wykorzystywana w programowaniu. Wykonuje się to przy użyciu komendy "Słownik" w menu "Plik" lub przycisku Słownik. Zmienne We/Wy są deklarowane automatycznie przy tworzeniu projektu.

Otwarte zostaje okno słownika. Wyboru globalnego słownika zmiennych binarnych dokonuje się wybierając menu "Plik", podmenu "Inne", podmenu "Zmienne globalne" i pole "Zmienne binarne". Do tego samego celu można wykorzystać przyciski Obiekty globalne i zakładkę Zmienne binarne.

Pole “Nowy” w menu "Edycja" jest stosowana do tworzenia nowych zmiennych binarnych. Można użyć również przycisku Wstaw obiekty. W otwartym oknie dialogowym wprowadź opis zmiennej wewnętrznej:

nazwa:	**RunCmd**
komentarz:	**zmienna Run/Stop: wewnętrzna**
atrybut:	Wybierz atrybut "**Wewnętrzna**"

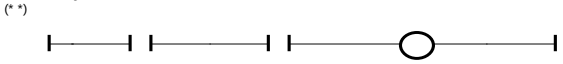
Naciśnij przycisk "**Zapisz**": zmienna została utworzona.
Naciśnij przycisk "**Anuluj**" aby wyjść z okna dialogowego.
Na koniec wyjdź z edytora słownika i zapisz wprowadzone modyfikacje: Menu "**Plik**" - pole "**Wyjście**". Kliknij "**TAK**" aby zapisać modyfikacje.

Edycja programu LD

Aby rozpocząć edycję programu LD "Command" dwukrotnie kliknij jego nazwę w oknie Zarządzanie Programami lub skorzystaj z przycisku Edytuj program.

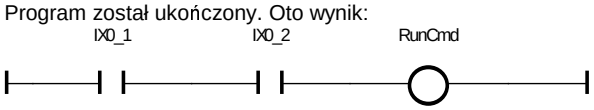
Zostaje otwarte okno Edytor LD ISaGRAF. Aby powiększyć obszar roboczy zmień rozmiar okna tak, aby wykorzystać całą powierzchnię ekranu.

F2 F3 Naciśnij klawisze F2 i F3:



Przyporządkuj zmienne symbolom LD: przesuń kursor używając klawiszy ze strzałkami. Umieść kursor na każdym z symboli i naciśnij klawisz Enter. Zostaje otwarte okno dialogowe wyboru zmiennych.
Dla pierwszego styku wpisz: IX0_1 <Enter> w polu wyboru zmiennej.
Dla drugiego styku wpisz: IX0_2 <Enter> w polu wyboru zmiennej.
Dla cewki wpisz: RunCmd <Enter> w polu wyboru zmiennej.

Program został ukończony. Oto wynik:



Wyjdź z edytora i zapisz wprowadzone modyfikacje: Menu "**Plik**" - pole "**Wyjście**". Kliknij "**TAK**" aby zapisać modyfikacje.

Edycja programu SFC

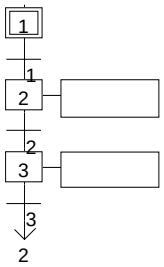
Aby rozpocząć edycję programu SFC "RunStop" kliknij dwukrotnie jego nazwę w oknie Zarządzanie Programami lub użyj przycisku "Edytuj program".

Otwarte zostaje okno Edytora SFC. Aby powiększyć obszar roboczy zmień rozmiar okna tak, aby wykorzystać całą powierzchnię ekranu:

Etap wstępny już istnieje i jest wybrany. Naciśnij klawisz strzałki w dół, aby wybrać pustą komórkę po etapie wstępnym (0,1)

F4 F3 Naciśnij F4, a potem F3 aby wstawić etap i przejście.
F4 F3 Naciśnij F4, a potem F3 aby wstawić jeszcze jeden etap i przejście.
F5 Naciśnij F5 aby do etapu wstawić skok i wybierz GS2 jako miejsce docelowe skoku.

Graf jest ukończony. Naciśnij przycisk "Powiększenie" na pasku narzędzi, aby zwiększyć rozmiar komórek i zapewnić miejsce do wyświetlenia instrukcji poziomu 2. Oto graf:



Aby rozpocząć programowanie przejścia "poziomu 2" wybierz je używając klawiszy ze strzałkami i naciśnij klawisz "Enter". Okno programowania poziomu 2 zostaje otwarte. Zaprogramuj poziom 2 dla przejścia 2:
RunCmd;

^TAB Naciśnij klawisze "Ctrl + Tab" aby przejść z powrotem do grafu SFC, zaznacz etap 3 i naciśnij klawisz "Enter", aby edytować jego tekst na poziomie 2:
QX1_1;
Zrób to samo, aby wprowadzić tekst przejścia 3:
Not (RunCmd);

^F4 Naciśnij klawisze " Ctrl + F4", aby zamknąć okno poziomu 2.

Program SFC jest ukończony. Wyjdź z edytora przy pomocy menu "**Plik**" i komendy "**Wyjście**" oraz zapisz wprowadzone modyfikacje klikając "**TAK**".

Budowanie kodu aplikacji

Aby stworzyć kod aplikacji użyj menu "**Kompilacja**" i komendy "**Utwórz kod aplikacji**" w oknie Zarządzanie Programami lub przycisku na pasku narzędzi.

Kiedy generowanie kodu zostanie ukończone, pojawi się okno dialogowe z zapytaniem czy zakończyć generowanie kodu **teraz** czy **kontynuować** pracę nad nim: Naciśnij przycisk "**Wyjdź**".

Symulacja

Aby uruchomić symulator jądra ISaGRAF użyj menu "**Debaguj**" i komendę "**Symuluj**" w oknie Programy lub przycisku na pasku narządzi.

Kiedy pojawi się okno Symulator, aplikacja może zostać przetestowana. W podanym przykładzie należy doprowadzić oba sygnały wejściowe 1 i 2 (zielone przyciski), aby uruchomić proces (czerwona dioda na wyjściu świeci się).

Zamknij okno **Debagera,** aby zakończyć symulację: menu "**Plik**" - pole "**Wyjście**".

A.2 Zarządzanie projektami

Aby uruchomić narzędzie zarządzania projektami ISaGRAF należy podwójnie kliknąć myszą na ikonie "Projekty" grupy ISaGRAF. Otwiera się wtedy okno "Zarządzanie Projektem".

Zmiana rozmiaru okien

Wystarczy kliknąć na separatorze (podziale) pomiędzy listą projektów a opisem aby zmienić rozmiar odpowiednich okien. Okno opisu nie może być całkowicie zasłonięte. Zawiera ono zawsze przynajmniej jedną linie tekstu.

Wstawianie separatorów

Linia separatora może zostać wstawiona przed dowolną nazwą projektu. Pozwala to pogrupować niektóre projekty połączone z jedną aplikacją w ramach układu listy. Komendy "**Edycja / Przełącz separator**" używa się do wstawienia lub usunięcia separatora przed wybranym projektem.

Przemieszczanie projektów na liście

Aby przemieścić projekt na liście, najpierw trzeba go zaznaczyć (podświetlić). Następnie należy kliknąć na jego nazwie i przeciągnąć go w nowe miejsce na liście. Przy przeciąganiu projektu, niewielka strzałka na lewym marginesie wskazuje miejsce, w którym zostanie on umieszczony. Można również wykorzystać komendę "**Przesuń do góry na liście**" menu "**Edycja**" do przenoszenia wybranego projektu linia po linii. Proszę zauważyć, iż jeśli separator zostanie umieszczony przed wybranym projektem będzie on przeniesiony wraz z projektem.

A.2.1.1 Tworzenie projektów i praca z nimi

Komendy menu menedżera projektu są używane do tworzenia nowych projektów, ich edycji i zarządzania istniejącymi projektami.

Tworzenie nowego projektu

Aby utworzyć nowy projekt należy najpierw wprowadzić jego nazwę. Tworzony jest wtedy pusty projekt, nie zawierający żadnych obiektów. Do nowo utworzonego projektu można dołączyć konfigurację We/Wy. Konfiguracja ta musi zostać zdefiniowana w bibliotece. Jeśli konfiguracja zostanie wybrana, ISaGRAF automatycznie skonfiguruje połączenie We/Wy i zadeklaruje odpowiadające mu

zmienne We/Wy w słowniku projektu. Przy tworzeniu lub zmianie nazwy projektu, należy stosować się do następujących zasad nadawania nazw:

- nazwa nie może być dłuższa niż **8** znaków
- pierwszy znak musi być **literą**
- kolejne znaki mogą być **literami**, **cyframi** lub znakiem podkreślenia
- nazwa projektu nie uwzględnia wielkości liter

Po stworzeniu projektu, komendy "**Edycja / Komentarz tekstowy**" używa się do wprowadzenia tekstu, który ma być wyświetlany wraz z nazwą projektu na liście.

Edycja opisu projektu

Komenda "**Projekt / Opis projektu**" jest używana do edycji opisu tekstowego projektu. Dokument ten w pełni identyfikuje projekt, odróżniając go od innych projektów z listy. Opis projektu może także być używany do zapisu wszelkich uwag dotyczących projektu w czasie jego istnienia.

Edycja projektu

Komenda "**Plik / Otwórz**" otwiera okno Zarządzania Programami dla wybranego projektu. Okno to pozwala na zarządzanie całą zawartością projektu (programami, parametrami aplikacji...). Można również podwójnie kliknąć na nazwie projektu, aby dokonać jego edycji.

Historia modyfikacji

"

System ISaGRAF zapisuje wszelkie modyfikacje związane z komponentem projektu w pliku historii. Każda modyfikacja identyfikowana jest w historii przy pomocy tytułu, daty i czasu. Plik historii zawiera ostatnich **500** modyfikacji. Każdy projekt posiada jeden plik historii. Historia modyfikacji projektu jest uzupełnieniem plików "dziennika" dołączanych do programów projektu. Komenda "**Projekt / Historia**" pozwala użytkownikowi przeglądać lub drukować historię modyfikacji dla wybranego projektu. Użytkownik może wybrać jeden lub więcej elementów na liście głównej i nacisnąć następujące przyciski:

OK..... zamyka okno
Drukuj..... wysyła zawartość lisy do drukarki
Pomoc..... wyświetla pomoc dotyczącą tego okna dialogowego
[usuń] Wybrane.. usuwa(kasuje) wybrane linie listy
[usuń] Wszystkie czyści całą listę
Znajdź ..... znajduje wzór na liście

Pole wprowadzania danych nad przyciskiem "**Znajdź**" jest używane do wprowadzania poszukiwanego wzoru. Funkcja ta nie uwzględnia wielkości liter. Kiedy wyszukiwanie osiągnie koniec listy, jest ono kontynuowane od początku listy, do momentu osiągnięcia miejsca rozpoczęcia.

Wydruk pełnego dokumentu

Komenda "**Projekt / Drukuj**" pozwala użytkownikowi zbudować i wydrukować pełny dokument dotyczący wybranego projektu. Dokument ten może zawierać dowolne komponenty (programy, zmienne, parametry...) wybranego projektu. Aby zbudować specyficzny (niepełny) dokument użytkownik musi jedynie określić jego spis treści.

Zabezpieczenie hasłem

Komenda "**Projekt / Ustaw hasło**" pozwala użytkownikowi zdefiniować zabezpieczenie hasłem dla narzędzi i danych wybranego projektu. Dalsze informacje na temat poziomów haseł i zabezpieczenia danych znajdują się w sekcji "**Ochrona hasłem**" pod koniec pierwszej części niniejszego podręcznika. Hasła odnoszą się jedynie do wybranego projektu. Nie mają one wpływu na inne projekty czy biblioteki ISaGRAF.

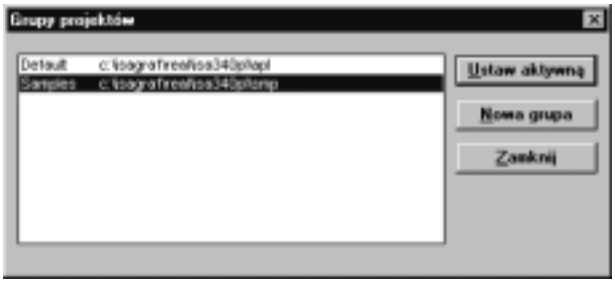
A.2.1.2 Praca z kilkoma grupami projektów

Projektowi ISaGRAF odpowiada jeden katalog na dysku, w którym przechowywane są wszystkie pliki projektu. "Grupie projektów" odpowiada lista katalogów projektu zebranych razem w jednym katalogu głównym. Grupa projektów identyfikowana jest przy pomocy nazwy. ISaGRAF tworzy domyślnie dwie grupy projektów:

"**Default**"	w "\ISAWIN\APL"	pulpit roboczy
"**Samples**"	w "\ISAWIN\SMP"	aplikacje przykładowe, dostarczone wraz z pakietem ISaGRAF

Nazwa wybranej aktualnie grupy projektów widnieje na pasku narzędziowym, obok przycisku służącego do wyboru grupy projektów:

Aby wybrać istniejącą grupę, lub stworzyć nową, można wybrać komendę "**Plik / Zmień grupę projektów**". Otwiera się poniższe okno dialogowe:



Wybierz grupę z listy i naciśnij "**Ustaw aktywną**" aby uaktywnić ją na liście zarządzania projektem. Aby wybrać grupę, można również kliknąć podwójnie na jej nazwie. Komendy "**Nowa grupa**" używa się do stworzenia nowej grupy. Komendy tej można użyć do przypisania nazwy grupy do istniejącego już katalogu lub do stworzenia nowej grupy w nowym katalogu.

Kiedy otwarte są inne okna ISaGRAF (menedżer programów, edytory...), nie można wybrać żadnej grupy.

A.2.1.3 Opcje

Komendy menu "**Opcje**" są używane do pokazywania lub chowania paska narzędzi, wybierania czcionki znakowej dla tekstu oraz włączania trybu „automatycznego zamykania" Menedżera Projektu. Wybrana czcionka znakowa jest stosowana do wyświetlania opisu projektu oraz jest wykorzystywana przez wszystkie edytory tekstowe ISaGRAF.

Kiedy wyłączona zostanie opcja "**Pozostaw otwartego Menedżera Projektów**", okno Menedżera Projektu jest automatycznie zamykane po wprowadzeniu projektu.

A.2.1.4 Narzędzia

Komendy menu "**Narzędzia**" są wykorzystywane do uruchamiania innych aplikacji ISaGRAF. Komenda "**Narzędzia / Archiwizuj projekty**" uruchamia menedżera archiwizacji ISaGRAF, pozwalającego zapisywać i przywracać projekty. Komenda " **Narzędzia / Archiwizuj wspólne dane**" jest używana do zapisywania i przywracania plików, wykorzystywanych we wszystkich projektach (takich jak wspólne słowa zdefiniowane).

Komenda " **Narzędzia / Biblioteki**" uruchamia w osobnym oknie menedżera bibliotek ISaGRAF.

Komenda " **Narzędzia / Importuj program IL**" może zostać użyta do importowania projektu, opisanego w pojedynczym programie IL program, w pliku tekstowym, zgodnie z formatem wymiany plików PLC Open.

A.3 Zarządzanie programami

Okno Programy zawiera programy (zwane również modułami lub jednostkami programowania) aplikacji i grupuje w swych menu dostępne komendy, pozwalające na tworzenie architektury projektu, uruchamianie edytorów, kompilatora i debagera. Okno to stanowi jądro pakietu ISaGRAF podczas opracowywania aplikacji. Okno Programy otwiera się po uruchomieniu komendy "Otwórz" okna Zarządzanie Projektami.

A.3.1 Komponenty projektu

Komponenty projektu zwane są **programami**. Program, to jednostka logiczna, opisująca jedną część realizacji sterowania. Zmienne globalne (takie jak zmienne We/Wy) mogą być wykorzystywane przez każdy program aplikacji. Zmienne lokalne mogą być wykorzystywane tylko przez jeden program. Programy są przedstawione w formie **drzewa**, podzielonego na odrębne **sekcje logiczne**. Okno przedstawia programy i powiązania pomiędzy nimi. Programy "**Najwyższego poziomu**" znajdują się po lewej stronie drzewa hierarchii.

Programy najwyższego poziomu

Programy najwyższego poziomu znajdują się po lewej stronie drzewa hierarchii. Programy najwyższego poziomu pierwszych trzech sekcji pozostają zawsze aktywne i są wykonywane w następującej kolejności podczas cyklu wykonywania (skanowania):

- (Odczyt wejść)
- Wykonanie programów najwyższego poziomu sekcji **POCZĄTEK**
- Wykonanie programów najwyższego poziomu sekcji **SEKWENCYJNIE**
- Wykonanie programów najwyższego poziomu sekcji **KONIEC**
- (Odświeżenie wyjść)

Programy sekcji "**Początek**" lub "**Koniec**" opisują operacje cykliczne. Nie są one zależne od zmiennej czasowej. Programy sekcji "**Sekwencyjnie**" opisują operacje sekwencyjne, w których w sposób wyraźny pojawia się zależność od czasu (w celu wyróżnienia operacji podstawowych). Programy główne sekcji "**Początek**" są wykonywane systematycznie na początku każdego cyklu realizacji. Programy główne sekcji "**Koniec**" są wykonywane systematycznie na końcu każdego cyklu. Programy główne sekcji "**Sekwencyjnie**" są wykonywane w oparciu o zasady **SFC** lub **FC** i muszą być napisane w języku **SFC** lub **FC**. Programy sekcji cyklicznych nie mogą być opisywane w języku **SFC** ani **FC**. Dowolny program każdej sekcji może posiadać jeden lub więcej **podprogramów**.

Funkcje i bloki funkcyjne

Programy sekcji "**Funkcje**" mogą być wywoływane przez każdy program dowolnej sekcji projektu. Funkcja to algorytm przetwarzający kilka wartości wejściowych na jedną wartość wyjściową. Algorytm funkcji działa tylko w przypadku ulotnych zmiennych pośrednich, kasowanych pomiędzy wywołaniami. Oznacza to, iż funkcja nie powinna nigdy wywoływać bloku funkcyjnego. Program sekcji "**Funkcje**" nie może być napisany w języku **SFC** ani **FC**.
W odróżnieniu od funkcji, "**Bloki funkcyjne**" zawierają algorytm działający na ukrytych statycznych wartościach wejściowych, kopiowanych (instancyjnych) przez system w przypadku każdego z przypadków użycia bloku funkcyjnego. Programy sekcji "**Bloki funkcyjne**" mogą być wywoływane przez każdy program dowolnej sekcji projektu. Nie mogą być one napisane w języku **SFC** ani **FC**.

Podprogramy

Podprogramy to funkcje przyporządkowane jednemu z programów-rodziców (SFC, FC lub innemu). Podprogram może być wykonywany (wywoływany) tylko przez swój program-rodzica. Każdy program każdej sekcji może posiadać jeden lub więcej podprogramów. Do opisu podprogramu może zostać wykorzystany dowolny program poza **SFC** i **FC**.

Programy potomne SFC oraz FC

Program potomny SFC to program równoległy, który może zostać uruchomiony lub zamknięty przez swój program-rodzica. Zarówno program-rodzic, jak i program potomny muszą być opisane w języku **SFC**.
Kiedy program-rodzic uruchamia program potomny **SFC**, wstawia do każdego etapu początkowego tego programu **znacznik SFC**. Kiedy program-rodzic zamyka program potomny **SFC**, czyści wszystkie znaczniki znajdujące się w etapach programu potomnego.
Dowolny program **FC** sekcji sekwencyjnej może sterować innymi podprogramami **FC**. Program-rodzic **FC** jest zablokowany (czeka) podczas wykonywania podprogramu FC. Nie jest możliwe równoczesne wykonywanie operacji w programie-rodzicu FC i jednym z jego podprogramów FC.

Powiązania pomiędzy programami a podprogramami:

Hierarchia programów jest obrazowana przez hierarchiczne drzewo, gdzie podprogramy i programy potomne są połączone na ze sobą przy pomocy linii. Powiązanie pomiędzy programem **SFC** i programem potomnym **SFC** zakończone jest strzałką. Należy zauważyć, że połączenie tego typu oznacza operacje **równoległe**.

Języki programowania

Każdy program jest opisywany przy pomocy tylko jednego **języka**. Język ten jest wybierany przy tworzeniu programu i nie może zostać później zmieniony. Jednakże schematy **FBD** mogą zawierać części schematów **LD**, a schematy **LD** mogą zawierać wywołania bloków funkcyjnych. Dostępne języki graficzne to: **SFC** (Graf Funkcji Sekwencyjnych), **FC** (Graf przepływowy) **FBD** (Funkcjonalny Diagram Blokowy) oraz **LD** (Diagram Drabinkowy lub Język stykowy). Dostępne języki tekstowe, to: **ST** (Tekst Strukturalny) oraz **IL** (Lista Instrukcji). Języki **SFC** i **FC** są zarezerwowane dla programów-rodziców i programów potomnych sekcji sekwencyjnej. Język każdego z programów jest przedstawiany przy pomocy ikony znajdującej się obok nazwy programu w oknie Programy. Poniżej przedstawiono ikony stosowane do oznaczania języków:

SFC ..... Graf Funkcji Sekwencyjnych
FC ..... Graf Przepływowy
FBD ..... Diagram Bloków Funkcyjnych
LD..... Diagram Drabinkowy (język stykowy); (wprowadzany przy pomocy edytora LD)
ST..... Tekst Strukturalny
IL ..... Lista Instrukcji

A.3.2 Praca z programami

Menu "**Plik**" łączy wszystkie komendy wykorzystywane do tworzenia, aktualizowania i modyfikowania programów. Uruchamia ono również odpowiednie edytory pozwalające na wprowadzanie treści programów aplikacji.

Tworzenie nowego programu

Funkcja "**Nowy**" menu "**Plik**" pozwala na tworzenie programów najwyższego poziomu, programów potomnych lub podprogramów w każdej sekcji programu. Pierwszą informacją, którą należy wprowadzić jest nazwa nowego programu, zgodna z następującymi zasadami nadawania nazw:

- maksymalna długość nazwy to **8** znaków
- pierwszym znakiem musi być **litera**
- kolejnymi znakami muszą być **litery**, **cyfry** lub znak '_'
- nadawanie programom nazw nie uwzględnia wielkości liter

Następnie należy wybrać język edycji , w którym decydujemy się opisywać nowy program:

SFC..... Graf Funkcji
FC ..... Graf Przepływowy
FBD..... Funkcjonalny Schemat Blokowy (może zawierać fragmenty w LD)
LD ..... Schemat Drabinkowy wprowadzany przy pomocy edytora LD
ST ..... Tekst Strukturalny
IL..... Lista Instrukcji

W końcu, należy wybrać styl wykonania programu:

Początek ..... najwyższy poziom sekcji "Początek"
Sekwencyjnie ..... najwyższy poziom sekcji "Sekwencyjnie"
Koniec ..... najwyższy poziom sekcji "Koniec"
Funkcja ..... w sekcji "Funkcje"
Blok funkcyjny .... w sekcji "Bloki funkcyjne"
Potomek..... program potomny SFC lub FC albo podprogram istniejącego programu

Wybierając jedną z pierwszych pięciu opcji, umieszczamy program na najwyższym poziomie sekcji **Początek**, **Koniec**, **Sekwencyjnie**, **Funkcja** lub **Blok funkcyjny**. Wybór ostatniej opcji oznacza, że program jest programem potomnym **SFC** lub podprogramem **FC** albo zwykłym podprogramem. Należy pamiętać, iż programy sekwencyjne najwyższego poziomu muszą być opisywane w języku **SFC** lub **FC** . Języki **SFC** i **FC** nie mogą zostać wykorzystane do tworzenia programów wykonywanych cyklicznie i ich podprogramów.

Wprowadzanie komentarzy dla każdego programu

ISaGRAF pozwala na dołączanie tekstu opisu do każdego programu w projekcie. Ten tekst komentarza jest wyświetlany małymi czcionkami obok nazwy programu. Komendy "**Plik / Komentarz do programu**" używa się do wprowadzania lub zmiany komentarza dołączonego do wybranego programu.

Edycja zawartości programu

Komenda ta pozwala modyfikować zawartość programu. Typ edytora używanego do wprowadzania programu zależy od języka w którym jest napisany. Edycja programu odbywa się w osobnych oknach. Dzieje się tak by możliwe było edytowanie więcej niż jeden programu (w różnych oknach). Naciśnięcie klawisza **ENTER** pozwala edytować podświetlony program. Aby rozpocząć edycję programu, użytkownik może również dwukrotnie kliknąć jego nazwę.

Edycja pliku „dziennika”

Plik dziennika jest dołączany do każdego programu. Jest to plik tekstowy, zawierający wszystkie uwagi dotyczące wprowadzanych w programie modyfikacji od chwili jego utworzenia. Plik dziennika można w dowolnym momencie edytować, swobodnie modyfikować i drukować. Przy wyjściu z edytora wykorzystywanego do modyfikacji kodu źródłowego programu, okno dziennika jest otwierane automatycznie. Umożliwia to wprowadzenie uwag do listy dziennika. Uwagi wstawiane są do pliku dziennika wraz z datą i czasem.

Słownik ze zmiennymi

Komenda "**Plik / Słownik**" uruchamia edytor słownika, w którym deklarowane są zmienne projektu. Zmienne mogą być globalne (znane dla wszystkich programów projektu) lub lokalne dla wybranego programu. Edytor słownika może być również wykorzystywany do deklarowania **zdefiniowanych słów**, które są semantycznymi zamiennikami stosowanymi w zastępstwie nazw lub wyrażeń w kodzie źródłowym programu.

Parametry funkcji, podprogramu lub bloku funkcyjnego

Komenda "**Plik / Parametry**" pozwala użytkownikowi określić parametry wywołania lub parametry zwracane wybranego podprogramu, funkcji lub bloku funkcyjnego. Komenda nie odnosi skutku, jeśli w oknie Programy wybrany jest program główny sekcji "**Początek**" lub "**Koniec**" albo program SFC.
Podprogramy, funkcje lub bloki funkcyjne mogą posiadać do **32** parametrów (wejściowych lub wyjściowych). Funkcja lub podprogram zawsze posiada jeden (i tylko jeden) parametr zwracany, który dla zgodności z regułami zapisu języka ST, musi posiadać taką samą nazwę jak funkcja.

Lista w lewej górnej części okna przedstawia parametry w kolejności wywołania: najpierw parametry wywołania, na końcu parametry zwracane (powrotu). Dolna część okna pokazuje szczegółowy opis parametru wybranego aktualnie z listy. Jako parametr mogą służyć wszystkie typy danych ISaGRAF. Parametry zwracane muszą być umieszczone na liście po parametrach wywołania. Nadawanie nazw parametrom musi być zgodne z następującymi zasadami:

- długość nazwy nie może przekraczać 16 znaków
- pierwszy znak musi być literą
- kolejne znaki muszą być literami, cyframi lub znakami podkreślenia
- nazwy nie uwzględniają wielkości liter

Komendy "**Wstaw**" używa się do wstawienia nowego parametru przed parametrem wybranym. Komendy "**Kasuj**" używa się do wykasowania wybranego parametru. Komenda "**Ułóż**" automatycznie porządkuje (sortuje) parametry, umieszczając parametry zwracane (powrotu) na końcu listy.

Przemieszczanie programu w drzewie hierarchii

Komenda "**Zmień nazwę/Przenieś**" w menu "**Plik**" używana jest do zmiany nazwy programu lub do przeniesienia go do innej sekcji drzewa hierarchii. Język opisu istniejącego już programu nie może zostać zmieniony. Przy wybraniu tej komendy otwierane jest to samo okno, którego używa się przy tworzeniu programów, a wszystkie pola zostają ustawione zgodnie z atrybutami wybranego programu. Nazwa programu może zostać zmodyfikowana. Można również zmienić sekcję i program-rodzica, co spowoduje przemieszczenie programu w drzewie hierarchii..

Komenda "**Uszereguj programy**" menu "**Plik**" jest używana do wyraźnego uporządkowania listy programów tego samego poziomu i posiadających jednego program-rodzica. Jeśli wybrany program znajduje się na najwyższym poziomie, komenda ta jest uporządkuje programy najwyższego poziomu wybranej sekcji. Jeśli

wybrany program znajduje się na niższym poziomie, komenda porządkuje jedynie programy potomne SFC i podprogramy, które posiadają tego samego rodzica, co wybrany program.
Kiedy otwarte zostanie okno dialogowe "**Uszereguj programy**", należy wybrać program, który ma zostać przemieszczony i nacisnąć przycisk "**Do góry**" lub "**W dół**". Spowoduje to przemieszczenie programu na liście.

Kopiowanie programów

Aby wykonać kopie programu, należy wybrać program-źródło z listy programów i wybrać komendę "**Plik / Kopiuj**". Przy wybraniu tej komendy otwierane jest to samo okno, które używa się przy tworzeniu programów. Wszystkie pola okna zostają ustawione zgodnie z atrybutami wybranego programu. Należy wprowadzić nazwę programu docelowego i jego lokalizację w sekcjach drzewa hierarchii. Jeśli program docelowy nie istnieje, zostaje utworzony w wybranej lokalizacji. Jeśli program docelowy już istnieje, zostaje zastąpiony. Wszystkie lokalne deklaracje i zdefiniowane słowa są kopiowane wraz z programem. Język opisu programu docelowego musi być taki sam jak ten użyty w programie-źródle. Naciśnięcie przycisku "**OK**" kopiuje program.

Komenda "**Kopiuj do innego projektu**" w menu "**Plik**" kopiuje wybrany program do innego projektu z zachowaniem nazwy. Programy potomne SFC i podprogramy wybranego programu mogą zostać skopiowane wraz z nim. Nazwy wybranych programów i ich programów potomnych nie mogą występować w docelowym projekcie. Przy użyciu tej komendy programy nie mogą być nadpisane. Wszystkie dołączone lokalne deklaracje i zdefiniowane słowa są kopiowane wraz z programami.

Usuwanie programów

Aby usunąć program, należy najpierw wybrać go z listy programów, a następnie wybrać komendę "**Plik / Usuń**". Program posiadający program potomny lub podprogramy nie może zostać usunięty. Aby usunąć program posiadający programy potomne lub podprogramy, należy najpierw usunąć jego programy potomne i podprogramy. Wszystkie lokalne deklaracje i zdefiniowane słowa są usuwane wraz z programem.

Importowanie funkcji i bloków funkcyjnych z biblioteki

Komenda "**Narzędzia / Importuj z biblioteki**" jest używana do kopiowania funkcji lub bloków funkcyjnych napisanych w językach IEC i umieszczonych w bibliotece w sekcji "**Funkcje**" lub " **Bloki funkcyjne**" otwartego projektu. Lokalne zmienne i zdefiniowane słowa dołączone do importowanej funkcji są kopiowane wraz z nią. Jeśli funkcja została prawidłowo skopiowana z biblioteki, można umieścić ja w innym miejscu lub w innej sekcji drzewa hierarchii przy użyciu komendy "**Plik / Zmień nazwę/Przesuń**". Aby uniknąć zbieżności nazw, importowane funkcje lub bloki funkcyjne muszą zostać przemianowane przy imporcie do projektu. W

przypadku funkcji nie można też zapomnieć o zmianie nazwy parametru zwracanego.

Eksportowanie funkcji i bloków funkcyjnych do biblioteki

Komenda "**Narzędzia / Eksportuj do biblioteki**" jest używana do programu z sekcji " **Funkcje**" lub "**Bloki funkcyjne**" (w otwartym projekcie) do odpowiedniej biblioteki. Lokalne zmienne i słowa zdefiniowane dołączone do eksportowanej funkcji są kopiowane wraz z nią. Eksportowana funkcja lub blok funkcyjny będą musiały zostać poddane ponownej kompilacji (zweryfikowane) z poziomu Menedżera Biblioteki ISaGRAF. Ma zapewnić to możliwość użycia ich w środowisku biblioteki. Funkcje i bloki funkcyjne biblioteki nie mogą wykorzystywać zmiennych globalnych.

A.3.3 Uruchamianie narzędzi generowania kodu

Komendy menu "**Kompilacja**" używane są do uruchamiania generatora kodu oraz do wprowadzania opcji i danych dodatkowych wykorzystywanych przy tworzeniu kodu aplikacji. Dalsze informacje na temat tych narzędzi znajdują się w rozdziale "**Korzystanie z Generatora Kodu**".

Tworzenie kodu aplikacji

Komenda "**Utwórz kod aplikacji**" uruchamia generator kodu projektu. Opcje generowania kodu dla oprogramowania sterownika muszą zostać prawidłowo ustawione przed uruchomieniem tej komendy. Przed wygenerowaniem kodu wykonywalnego, każdy program który nie został dotąd zweryfikowany jest sprawdzany pod kątem występowania błędów składni. ISaGRAF zawiera kompilator przyrostowy, który nie rekompiluje programów już skompilowanych.

Weryfikacja wybranego programu

Komenda "**Weryfikuj**" pozwala użytkownikowi na weryfikację składni programu wybranego aktualnie z listy. Kiedy program zostanie zweryfikowany i nie zostaną wykryte żadne błędy. Nie będzie on ponownie weryfikowany podczas generowania kodu dopóki nie zostanie zmieniona jego zawartość lub przynależne od niego zdefiniowane słowa.

Symulowanie modyfikacji

Komenda "**Uznaj źródła za modyfikowane**" symuluje zmianę w każdym programie tak, aby wszystkie programy zostały ponownie skompilowane podczas następnego generowania kodu.

Opcje pracy aplikacji

Komenda „**Opcje pracy aplikacji**” otwiera okno dialogowe, w którym wprowadza się główne parametry wykonawcze działania aplikacji. Obejmują one programowanie parametrów czasowych cyklu, zarządzanie błędami wykonania, tryb uruchamiania oraz implementację sprzętową umożliwiającą zachowywanych zmiennych. Więcej wyjaśnień dotyczących tej komendy podano w rozdziale "Korzystanie z Generatora Kodu".

Opcje kompilatora

Komenda ta jest używana do konfiguracji opcji wykorzystywanych w Generatorze Kodu ISaGRAF do tworzenia i optymalizacji kodu oprogramowania wbudowanego. Więcej wyjaśnień dotyczących tej komendy podano w rozdziale "Korzystanie z Generatora Kodu".

Definiowanie zasobów

"**Zasób**" to dane zdefiniowane przez użytkownika (na przykład plik), które mają zostać włączone do kodu wynikowego tak, aby można było załadować je wraz z tym kodem. Więcej wyjaśnień na temat pliku definicji zasobów podano w rozdziale "Korzystanie z Generatora Kodu".

A.3.4 Inne narzędzia ISaGRAF

Menu "**Projekt**" grupuje komendy uruchamiające narzędzia ISaGRAF dla wybranego projektu. Więcej informacji na temat tych narzędzi podano w odpowiednich rozdziałach tego dokumentu.

Konfigurowanie We/Wy

Komenda "**Konfiguruj We/Wy**" uruchamia edytor połączeń wejść i wyjść ISaGRAF. Narzędzie to jest wykorzystywane do ustalenia związków pomiędzy zmiennymi We/Wy zadeklarowanymi w słowniku projektu a odpowiadającym im sprzętem.

Działanie edytora odsyłaczy

Komenda "**Odsyłacze**" pozwala użytkownikowi obliczać, przeglądać lub drukować odsyłacze projektu. Odsyłacze przedstawiają użytkownikowi wszystkie wystąpienia każdej zmiennej w kodzie źródłowym programów całego projektu. Funkcja ta jest bardzo przydatna przy wykrywaniu dostępu do zmiennej lub dowolnego zasobu globalnego lub do sporządzania listy wszystkich wystąpień zmiennej globalnej w kodzie źródłowym.

- ***Wprowadzanie opisu projektu***

Komenda "**Opis projektu**" jest używana do edycji opisu projektu. Dokument ten w pełni identyfikuje projekt odróżniając go od innych projektów z listy. Opisu projektu można również używać do zapisu dowolnych uwag w czasie istnienia projektu. Opis projektu jest wyświetlany w oknie Menedżera Projektów.

- ***Wydruk pełnej dokumentacji***

Komenda "**Drukuj dokumentację projektu**" pozwala użytkownikowi zbudować i wydrukować pełny dokument dotyczący wybranego projektu. Dokument ten może łączyć dowolne składniki (program, zmienne, parametry, itp.) wybranego projektu. Aby utworzyć szczególny (niepełny) dokument, użytkownik musi jedynie określić jego zawartość.

- ***Historia modyfikacji***

Komenda ta otwiera okno dialogowe, w którym wyświetlana jest historia modyfikacji projektu. Więcej wyjaśnień dotyczących tej komendy podano w rozdziale "Zarządzanie Projektami".

A.3.5 Dodawanie komend do menu Narzędzia

ISaGRAF zapewnia możliwość wstawiania innych komend do menu "**Narzędzia**". Komendy zdefiniowane przez użytkownika, które mają być dodane do menu "**Narzędzia**", wpisane są w pliku tekstowym "**\ISAWIN\COM\ISA.MNU**". Można dodać do 10 komend. W każdej linii można wstawiać komentarze, rozpoczynające się znakiem "**;**". Każda komenda jest opisana w dwóch liniach tekstu, zgodnie z następującą składnią:

```
M=łańcuch_menu
C=linia_komend
```

Łańcuch menu, to tekst wyświetlany w menu "**Narzędzia**". Linia komend zawiera nazwę dowolnego pliku wykonywalnego systemu MS-DOS lub Windows i (opcjonalnie) parametry wywołania. W linii komend można używać łańcucha "**%A**", który zostanie zamieniony na nazwę otwartego projektu oraz łańcucha "**%P**", który zostanie zamieniony na nazwę wybranego programu. Poniższy przykład uruchamia "Notatnik" w celu edycji wybranego programu (do użytku z programami ST i IL):

```
M=Edytuj za pomocą Notatnika
C=Notepad.exe \isawin\apl\%A\%P.lsf
```

A.3.6 Symulacja i debagowanie aplikacji

Komendy Menu "**Debagowanie**" są używane do uruchamiania debagera graficznego ISaGRAF w trybie symulacji lub w trybie połączenia rzeczywistego.

Symulacja

Komenda "**Symuluj**" otwiera debager w trybie symulacji. W trybie tym pojawia się następne okno, zwane symulatorem. Komenda ta jest bardzo przydatna do testowania dowolnej aplikacji, kiedy sterownik nie jest dostępny. Uruchomienie symulatora zamyka okno Programy. Okno Programy jest ponownie otwierane w trybie debagowania po otwarciu zarówno okna debagera, jak i okna symulacji. Symulatora nie można uruchomić jeśli nie został wygenerowany kod wynikowy. Symulatora nie można uruchomić, jeśli okna potomne (edytory, generowanie kodu, konfigurator We/Wy, itp.) są otwarte. Przed uruchomieniem tej komendy, wszystkie te okna muszą zostać zamknięte. Komenda ta jest również dostępna z menu edytorów ISaGRAF.

Debagowanie w trybie rzeczywistym

Komenda "**Debaguj**" otwiera główne okno debagera i zamyka okno programu. Okno programu jest ponownie otwierane w trybie debagowania, kiedy tylko zostanie nawiązana komunikacja pomiędzy debagerem i aplikacją oprogramowania wbudowanego. Debager nie może zostać uruchomiony jeśli nie został wygenerowany kod wynikowy. Debagera nie można uruchomić również, jeśli okna potomne (edytory, generowanie kodu, konfigurator We/Wy, itp.) są otwarte. Przed uruchomieniem tej komendy, wszystkie one muszą zostać zamknięte. Komenda ta jest również dostępna z menu edytorów ISaGRAF.

Przygotowywanie pulpitu roboczego debagera

Komenda "**Debaguj / Pulpit roboczy**" pozwala określić listę dokumentów (okien) otwieranych wraz z debagerem. Dokumenty takie mogą być programami, grafiką Synoptyki, listami zmiennych. Grafika i lista diagramów czasowych z poprzednich wersji ISaGRAF są również ujmowane na liście projektu. Dokumenty określone w pulpicie roboczym debagera są automatycznie otwierane, gdy uruchamiana jest symulacja lub monitorowanie.

Powyższe okno dialogowe przedstawia istniejące dokumenty projektu po lewej i dokumenty wybrane do początkowego stanu pulpitu roboczego po prawej. Używając przycisków ">>" oraz "<<" można przenosić dokumenty z jednej listy na drugą. Każdy projekt posiada własną listę dokumentów pulpicie roboczym debagera.

Konfiguracja połączenia

Komenda "**Parametry połączenia**" pozwala użytkownikowi określić parametry połączenia używanego do komunikacji pomiędzy debagerem w komputerze PC a oprogramowaniem sterownika (systemem ISaGRAF w sterowniku).

"**Numer sterownika**" określa system lub zadanie oprogramowania ISaGRAF sterownika. Jest on liczbą z przedziału od **1** do **255**. Numer sterownika jest podany w podręczniku dostarczonym ze sterownikiem od dostawcy..
"**Port komunikacyjny**" określa sprzęg pomiędzy pakietem ISaGRAF a systemem sterownika. Może być to nazwa portu szeregowego lub nazwa sieci "**Ethernet**". Oprogramowanie sieciowe używa protokołu TCP-IP poprzez "Winsock" w wersji 1.1.
"**Limit czasu**" to czas pozostały systemowi oprogramowania wbudowanego na wykonanie operacji komunikacyjnych pomiędzy końcem zapytania debagera a początkiem odpowiedzi. Czas ten jest określany w **milisekundach**. Pole "Powtórzenia" oznacza ilość automatycznych prób ponowienia operacji transmisyjnych wykonywanych przez debager przed wygenerowaniem błędu komunikacji.

Konfiguracja łącza szeregowego

Kiedy wybrany zostanie port szeregowy (COM1-4), przycisk "**Ustawienia**" jest używany do uzyskania dostępu do parametrów komunikacyjnych łącza szeregowego.
Wybrać można szybkość transmisji, parzystość oraz format. Gdy w polu "**Kontrola przepływu**" wybrana jest opcja "**sprzęt**", pakiet ISaGRAF steruje liniami CTS i DSR, umożliwiając sprzętowe uzgodnienia podczas operacji transmisyjnych (sprzętowe sterowanie przepływem).

Konfiguracja łącza Ethernet

Kiedy jako port komunikacyjny wybrano "Ethernet", przycisku "**Ustawienia**" używa się do wprowadzenia "Adresu sieciowego" (IP) oraz numeru "Portu" używanego przez protokół TCP/IP.
Pola te wykorzystują formaty standardowe, definiowane przez interfejs Socket. Pakiet ISaGRAF komunikuje się poprzez protokół TCP/IP używając biblioteki WINSOCK.DLL w wersji 1.1. Plik ten musi być prawidłowo zainstalowany na dysku.
Jeśli uruchamiając system sterownika ISaGRAF nie określono innego, domyślnym używanym numerem portu jest "**1100**".

A.4 Korzystanie z edytora SFC

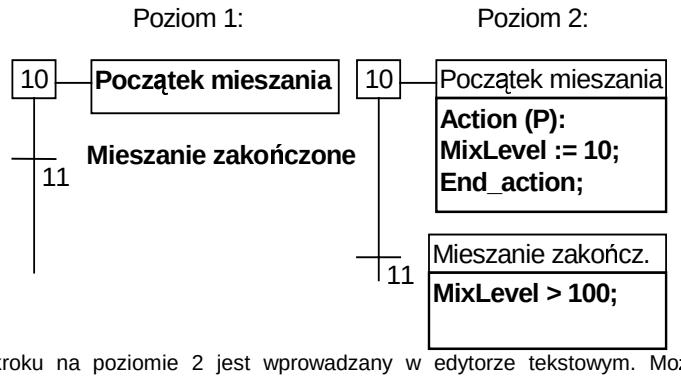
Język SFC jest używany do opisu operacji procesu sekwencyjnego. Wykorzystuje on prostą reprezentację graficzną poszczególnych kroków procesu oraz warunków, pozwalających na zmianę kroków aktywnych. Program SFC wpisywany jest przy użyciu edytora graficznego SFC ISaGRAF. SFC to rdzeń normy IEC 1131-3. Inne języki opisują zazwyczaj działania w ramach kroków oraz warunki logiczne przejść. Edytor graficzny SFC ISaGRAF pozwala użytkownikowi na wprowadzanie pełnych programów SFC. Stanowi on kombinację graficznych i tekstowych możliwości edycyjnych, umożliwiając w ten sposób wprowadzanie zarówno grafów SFC, jak i odpowiadających im działań i warunków.

A.4.1 Główne pojęcia języka SFC

Język SFC wykorzystywany jest do reprezentacji procesów sekwencyjnych. Dzieli on cykl procesu na kilka dokładnie zdefiniowanych, następujących po sobie **kroków** (elementów samodzielnych), oddzielonych **przejściami**. Więcej szczegółów dotyczących języków SFC można znaleźć w Podręczniku Informacyjnym ISaGRAF dotyczącym Języków.
Komponenty SFC połączone są za pomocą **linii zorientowanych**. Domyślna orientacja linii to **z góry** do **dołu**. Oto podstawowe komponenty graficzne, wykorzystywane do budowy grafu SFC:

..... Krok początkowy
..... Krok
..... Przejście
..... Skok do kroku
..... Krok typu makro
..... Początek kroku makro
..... Koniec kroku makro

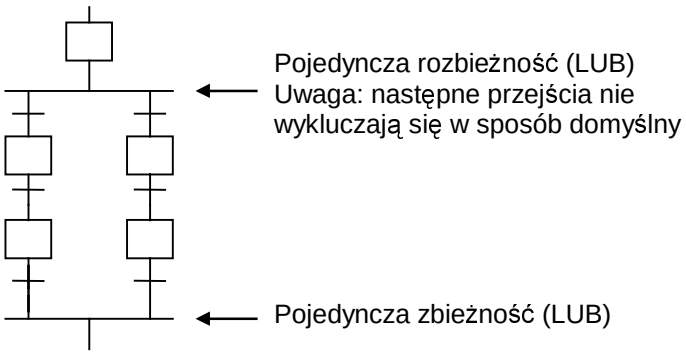
Programowanie w SFC posiada zazwyczaj na dwa odrębne poziomy: **Poziom 1** przedstawia schematy graficzny, numery odniesienia kroków i przejść oraz komentarze dołączane do kroków i przejść. **Poziom 2** to programowanie w **ST** lub **IL** działań w ramach kroków lub warunków dotyczących przejść. Działania i warunki mogą odwoływać się do **podprogramów** napisanych w innych językach (**FBD**, **LD**, **ST** lub **IL**). Poniżej znajduje się przykład programowania poziomu 1 i 2:



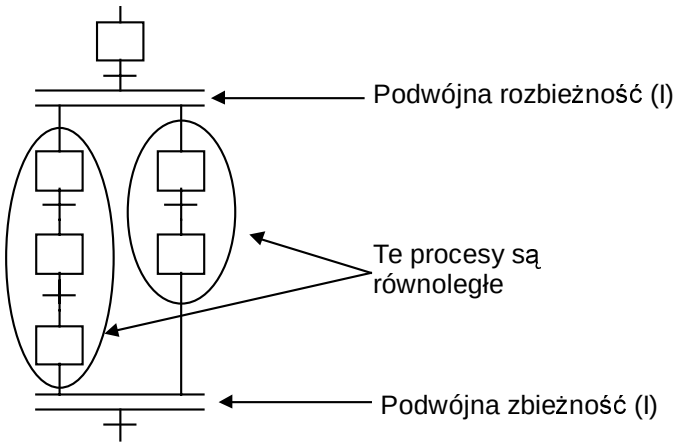
Kod kroku na poziomie 2 jest wprowadzany w edytorze tekstowym. Może on obejmować bloki operacji utworzone w ST lub IL. Kod przejścia na poziomie 2 może być wprowadzany poprzez języki tekstowe IL lub ST, albo też przy pomocy edytora LD.

Rozbieżności i zbieżności

Rozbieżności i zbieżności wykorzystywane są do przedstawienia **wielu połączeń** pomiędzy krokami i przejściami. Proste rozbieżności i zbieżności reprezentują różne możliwości **włączania** pomiędzy różnymi częściami składowymi procesu.

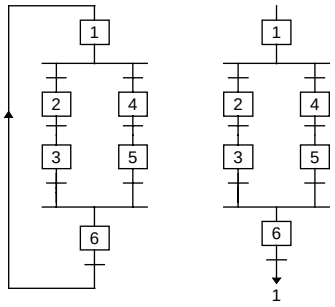


Podwójne rozbieżności reprezentują procesy **równoległe**.



Skok do kroku

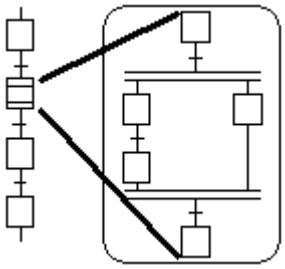
Edytor SFC pozwala użytkownikowi jedynie na łączenie w kierunku **z góry** do **dołu**. **Skok** do kroku można wykorzystać do przedstawienia połączenia z górną częścią grafu. Poniższe grafy są równoważne:



Skok do przejścia jest zabroniony i musi zostać wyraźnie przedstawiony jako podwójna zbieżność (I).

Kroki typu makro

Krok typu makro to **nie powtarzająca się** reprezentacja **samodzielnej** grupy kroków i przejść. Krok typu makro rozpoczyna się **początkiem kroku makro**, a kończy **końcem kroku makro**.



Szczegółowa reprezentacja kroku typu makro musi być opisana w tym samym programie SFC. Symbol kroku typu makro musi posiadać taki sam **numer odniesienia** jak początek kroku makro. Opis kroku typu makro może zawierać inny krok makro.

A.4.2 Wprowadzanie grafu SFC

Aby narysować graf SFC, użytkownik musi po prostu wprowadzić znaczące komponenty grafu. Wszystkie linie pojedyncze, łączące jeden lub więcej elementów (pionowo czy poziomo) są rysowane automatycznie przez edytor SFC. Aby umieścić na grafie komponent SFC, użytkownik musi przesunąć kursor na właściwe miejsce i wybrać komponent z paska narzędzi edytora. Symbol jest wstawiany w miejscu aktualnego położenia kursora. Można również wykorzystać następujące sekwencje klawiszy:

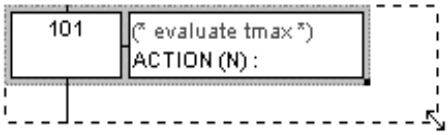
F2:	Wstaw krok rozpoczynający
F3:	Wstaw pojedynczy krok
F4:	Wstaw przejście
F5:	Wstaw skok do kroku
F6: F7:	Wstaw rozbieżność lub zbieżność LUB / Dodaj gałęzie
↑F6: ↑F7:	Wstaw rozbieżność lub zbieżność I / Dodaj gałęzie
F8:	Wstaw krok typu makro
F9: ↑F9:	Wstaw krok rozpoczynający lub kończący treść kroku typu makro

(Symbol "↑" oznacza kombinację z klawiszem SHIFT)

Siatka edycyjna pokazuje **komórki macierzy**. Opcja edytora pozwala użytkownikowi pokazać lub ukryć siatkę podczas edycji. Siatka jest bardzo przydatna przy tworzeniu początkowego układu grafu SFC lub przy wybieraniu elementów składowych grafu. Komendę "**Opcje / Ustawienia widoku**" używa się do pokazania/ukrycia siatki.

Edytor SFC ISaGRAF zawsze pokazuje bieżącą pozycję w macierzy. Wybrana komórka jest zaznaczona kolorem szarym. Niewielki kwadrat w jej prawym dolnym

rogu może służyć do swobodnej zmiany rozmiarów komórki. W ten sposób można również zmieniać stosunek szerokości komórki do jej wysokości.



Tworzenie rozbieżności lub zbieżności

Rozbieżności(rozejścia) lub zbieżności(zejścia) są zawsze rysowane **od lewej do prawej**. Aby narysować rozbieżność lub zbieżność należy umieścić jej **lewe gałęzie** na obszarze grafu. Typ rysunku (pojedynczy lub podwójny) jest ustawiany przez wybór powyższych przycisków na pasku narzędzi.

Dodawanie gałęzi do rozbieżności

Pozycję **początkową** i **końcową** każdej **gałęzi pomocniczej** umieszcza się na linii rozbieżności lub zbieżności używając powyższych przycisków z paska narzędzi. Lewy róg rozbieżności lub zbieżności musi już istnieć przed wstawieniem nowych gałęzi. Prawe rogi mają taki sam styl (pojedynczy lub podwójny) jak główny lewy róg. Nie można umieszczać rogów prawych, jeśli nie dodano głównego rogu lewego.

Wstawianie kroku typu makro

Przycisku tego używa się wstawiania w grafie głównym kroku typu makro. Treść kroku makro musi zostać wprowadzona w innym miejscu tego samego programu SFC.

Treść kroku typu makro

Kroki makro muszą być opisywane w tym samym programie SFC, w którym jest graf główny. Krok makro musi rozpoczynać się **początkiem kroku makro**, a kończyć się **końcem kroku makro**. Podgraf opisany jako implementacja makro musi być **samodzielny**. Krok rozpoczynający makro musi posiadać taki sam **numer odniesienia** jak symbol kroku makro na gałęzi głównej.

A.4.3 Opracowywanie istniejącego grafu SFC

Używając myszy lub klawiszy kierunkowych zaznacz na grafie prostokątny obszar. Cały zaznaczony obszar ma kolor szary. Można teraz stosować komendy menu "**Edycja**":

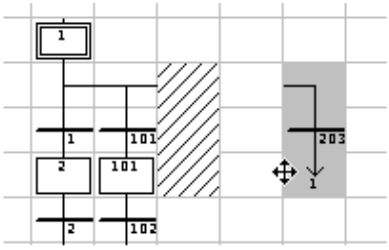
Komendy wytnij / kopiuj / wyczyść / wklej

Następujące komendy z menu "**Edycja**" są dostępne, kiedy na pasku narzędziowym edytora wybrany jest przycisk "**strzałki**":
Wytnij ..... Przenoszenie zaznaczonego prostokąta z ekranu do schowka SFC
Kopiuj ..... Kopiowanie zaznaczonego prostokąta z ekranu do schowka SFC
Kasuj ..... Czyszczenie (kasowanie) zaznaczonego prostokąta
Wklej ..... Wstawianie zawartości schowka SFC w bieżącej pozycji

Komenda "**Edycja / Wklej**" kopiuje schowek SFC na ekran. Komendy Kopiuj / Wklej działają zarówno na grafie SFC, jak i na kodzie poziomu 2 kroku/przejścia. Możliwe jest również skopiowanie grafu w jednym programie i wklejenie go w innym programie SFC. Elementy są wstawiane przed aktualnie zaznaczoną pozycją.

Przemieszczanie elementów

Kiedy elementy SFC zostaną wybrane na grafie SFC, można przenieść je w inne miejsce grafu przez przeciągnięcie zaznaczenia przy pomocy myszy. Podczas przeciągania zaznaczenia pierwotna lokalizacja wybranych elementów zostaje zakreskowana.



Obszar docelowy przenoszonych elementów musi być pusty. Wstawianie nie jest możliwe podczas przemieszczania symboli SFC.

Ponowne numerowanie kroków i przejść

Każdy krok lub przejście jest identyfikowany za pomocą numeru logicznego na grafie SFC. Komenda "**Edycja / Przenumeruj**" pozwala użytkownikowi

automatycznie ustawić kolejne numery odniesienia dowolnego z kroków lub przejść aktualnie edytowanego programu SFC. Po zmianie numeru kroku wszystkie skoki do tego kroku zostają automatycznie zaktualizowane o nowy numer odniesienia. (dotyczy to również kroków typu makro i kroków rozpoczynających)

Dostęp bezpośredni do kroku lub przejścia

Komenda "**Edycja / Skocz do**" pozwala użytkownikowi na dostęp do istniejącego kroku lub przejścia. Miejsce przewijania jest automatycznie zaktualizowane, aby krok lub przejście były widoczne.

Znajdowanie i zamiana tekstów

Komenda "**Edycja / Znajdź / Zamień**" może zostać wykorzystana do zamiany ciągów znaków w całym programie (wszystkie kroki i przejścia). Okno dialogowe Znajdź/Zamień jest używane do wpisania poszukiwanego tekstu i bezpośredniego otwierania sekcji kodowej poziomu 2, w której tekst ten się pojawia.

A.4.4 Wprowadzanie programu poziomu 2

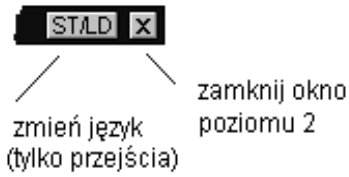
Aby wstawić tekst poziomu 2 użytkownik musi dwukrotnie kliknąć krok lub przejście. Program poziomu 2 jest wyświetlany po prawej stronie okna SFC. Linia podziału pomiędzy grafem SFC a programem poziomu 2 może być dowolnie przemieszczana.

Można otworzyć jednocześnie jeden lub dwa obszary poziomu 2. Następujące komendy są dostępne z klawiatury, za pośrednictwem myszy, lub dzięki pozycjom menu "Edycja":

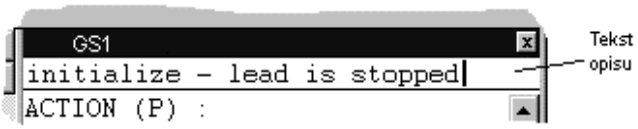
	Klawiatura	*Mysz*	*Menu "Edycja"*
Otwórz w ostatnim oknie domyśl.	Enter	Dwukrotne kliknięcie	Edycja poziomu 2
Otwórz w osobnym oknie	Ctrl+Enter	Ctrl + Dwukrotne klikn.	Edycja poziomu 2 w osobnym oknie

Kiedy widoczne są dwa okna poziomu 2, linię podziału pomiędzy nimi można swobodnie przemieszczać. Przycisk po prawej stronie paska tytułowego poziomu 2 jest używany do zamykania okna poziomu 2.

Językiem domyślnym programowania Poziomu 2 jest **ST** (Tekst Strukturalny). W przypadku przejść, kod poziomu 2 może być również wprowadzany przy pomocy edytora **LD**. Przycisku "**ST/LD**" na pasku tytułowym poziomu 2 używa się do zmiany aktywnego języka. Komenda ta jest dostępna tylko wtedy, gdy okno programowania poziomu 2 jest puste.



W górnej części okna poziomu 2 pojawia się pole do edycji pojedynczej linia. Pole to służy do wprowadzenia krótkiego tekstu opisu. Tekst ten będzie wyświetlany jako komentarz IEC przy rysowaniu symboli SFC. Jest on bardzo przydatny, ponieważ wykorzystywany jest on także przez inne komendy, takie jak "Skocz do..." oraz przy drukowaniu kroków i przejść SFC do dokumentu.



Komenda "**Opcje / Odśwież**" może być użyta w dowolnym momencie, kiedy otwarte są okna poziomu 2. Służy do odświeżenia głównego grafu SFC zmodyfikowanymi programami poziomu 2.

Wstawianie nazwy zmiennej

Programując w języku tekstowym należy nacisnąć ten przycisk w celu wybrania zmiennej zadeklarowanej w słowniku projektu i wstawienia jej nazwy w miejscu aktualnego położenia znaku „^”. Programując w LD należy nacisnąć ten przycisk w celu wybrania zmiennej skojarzonej z wybranym stykiem lub parametrem bloku We/Wy.

Wstawianie do kroku impulsowego bloku wykonawczego

Programując na poziomie 2 kroku należy nacisnąć ten przycisk w celu wstawienia szablonu impulsowego bloku wykonawczego w miejscu aktualnego położenia znaku „^”. Poniżej przedstawiono format impulsowego bloku wykonawczego:

```
Action (P) :
   ST statement;
   ...
End_Action;
```

Operacje impulsowe to instrukcje wykonywane tylko jeden raz w chwili uaktywniania kroku. Dalsze szczegóły dotyczące programowania SFC znajdują się w informacjach na temat języka ISaGRAF.

N *Wstawianie do kroku nie przechowywanego bloku wykonawczego*

Programując na poziomie 2 kroku należy nacisnąć ten przycisk w celu wstawienia szablonu nie przechowywanego(nie zachowywanego) bloku wykonawczego w miejscu aktualnego położenia znaku „^". Poniżej przedstawiono format nie przechowywanego bloku wykonawczego:

```
Action (N) :
   ST statement;
   ...
End_Action;
```

Operacje nie przechowywane to instrukcje wykonywane w każdym cyklu PLC, kiedy krok jest aktywny. Dalsze szczegóły dotyczące programowania SFC znajdują się w informacjach na temat języka ISaGRAF.

P0 P1 *Nowe kwantyfikatory operacji P0 i P1*

ISaGRAF obsługuje nowe kwantyfikatory operacji **P0** i **P1**. Programując na poziomie 2 kroku należy nacisnąć ten przycisk w celu wstawienia szablonu bloku wykonawczego P0 lub P1 w miejscu aktualnego położenia znaku „^". Poniżej przedstawiono format bloku takich operacji:

```
Action (P0) :              Action (P1) :
   ST statement;              ST statement;
   ...                        ...
End_Action;                End_Action;
```

Operacje P1 to instrukcje wykonywane tylko jeden raz w chwili uaktywniania kroku (tak samo jak operacje impulsowe). Operacje P2 to instrukcje wykonywane tylko jeden raz w chwili zamykania kroku. Dalsze szczegóły dotyczące programowania SFC znajdują się w informacjach na temat języka ISaGRAF.

Operacje binarne

Inna semantyka tekstowa umożliwia bezpośrednie operacje na zmiennych binarnych w zależności od operacji kroku. Operacje takie polegają na kojarzeniu **sygnału aktywności kroku** z wewnętrzną lub wyjściową zmienną binarną. Oto struktura podstawowych operacji binarnych:

<zmienna_binarna> (N);	przyporządkowuje sygnał aktywności kroku do zmiennej
< zmienna_binarna>;	to samo (atrybut N jest opcjonalny)
/ < zmienna_binarna>;	przyporządkowuje negację sygnału aktywności kroku do zmiennej

Inne funkcje pozwalają ustawiać lub kasować zmienną binarną w chwili uaktywniania kroku. Oto struktura operacji binarnych ustawiania i kasowania:

< zmienna_binarna> (S);	ustawia zmienną na wartość PRAWDA, kiedy sygnał aktywności kroku ma wartość PRAWDA
< zmienna_binarna> (R);	przestawia wartość zmiennej na FAŁSZ, kiedy sygnał aktywności kroku ma wartość PRAWDA

Operacje SFC

Inna semantyka tekstowa pozwala sterować wykonywaniem programu potomnego SFC. Operacja SFC to sekwencja potomna SFC, uruchamiana lub zamykana w zależności od stanu sygnału aktywności kroku. Operacja SFC może posiadać kwantyfikator **N** (Nie przechowywana), **S** (Ustaw) lub **R** (Resetuj). Oto budowa podstawowych operacji SFC:

<program_potomny> (N);	uruchamia sekwencje potomną kiedy krok uaktywnia się i zamyka sekwencje potomną, gdy krok staje się nieaktywny
< program_potomny>;	skutek taki sam jak w przypadku poprzedniej składni (atrybut N jest opcjonalny)
< program_potomny> (S);	uruchamia sekwencje potomną kiedy krok uaktywnia - gdy krok staje się nieaktywny, nic się nie dzieje
< program_potomny> (R);	zamyka sekwencje potomną kiedy krok uaktywnia - gdy krok staje się nieaktywny, nic się nie dzieje

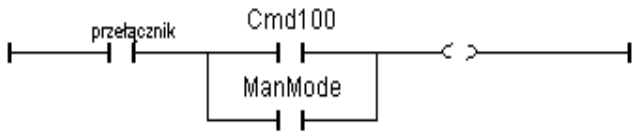
Sekwencja SFC określona jako operacja, musi być istniejącym **programem potomnym SFC** aktualnie edytowanego programu utworzonego przy pomocy Menedżera Programów ISaGRAF.

Przejścia napisane w ST

Przejścia poziomu 2 to wyrażenie binarne. Aby zaprogramować je w języku ST, wystarczy wprowadzić warunki logiczne zgodnie ze składnią ST. Na końcu wyrażenia opcjonalnie można dodać średnik.

Przejścia napisane w LD

Edytor LD umożliwia programowanie warunków przejścia na poziomie 2. W takim przypadku schemat tworzy tylko jeden szczebel z tylko jedną cewką reprezentującą przejście. Nazwa przejścia nie jest powtarzana przy symbolu cewki. Poniżej znajduje się przykład warunku dla przejścia zaprogramowanego w LD.



Programując w LD należy użyć klawiaturowych klawiszy ze strzałkami do przesuwania zaznaczenia siatki logicznej, a następnie wstawia się symbol stosując następujące skróty:
F2:..... wstawia styk za wybranym symbolem / zapoczątkowuje szczebel
F3:..... wstawia styk przed wybranym symbolem
F4:..... wstawia styk równolegle do wybranego symbolu
F6:..... wstawia blok za wybranym symbolem
F7:..... wstawia blok przed wybranym symbolem
F8:..... wstawia blok równolegle do wybranego symbolu

Aby wybrać zmienną lub wprowadzić wartość stałą, naciśnij klawisz ENTER gdy wybrany jest styk lub parametr We/Wy bloku. Aby wybrać typ bloku funkcyjnego, naciśnij klawisz ENTER gdy wybrany jest blok funkcyjny. Ten sam skutek przynosi dwukrotne kliknięcie danego symbolu.

Aby zmienić typ styku (bezpośredni, negowany lub z wykrywaniem impulsu), naciśnij SPACJĘ gdy jest wybrany dany styk. Więcej szczegółów na temat możliwości LD podano w rozdziale "Korzystanie z edytora LD" w tym dokumencie.

A.4.5 Korzystanie z galerii SFC

Edytor SFC ISaGRAF zawiera galerię SFC: jest to kolekcja struktur SFC, które można wstawić do dowolnego grafu SFC. Elementy galerii SFC mogą być opcjonalnie osadzane w krokach i przejściach kodu poziomu 2. Można stosować następujące komendy menu "**Narzędzia**":

Kopiuj do galerii SFC skopiuj wybrane elementy do galerii SFC
Wstaw z galerii SFC wklej element galerii SFC w aktualnie wskazywanym miejscu

Kopiując do galerii SFC (tj. tworząc nowy element galerii SFC) można opcjonalnie zażądać osadzanie kodu poziomu 2 dla wybranych symboli SFC.

A.5 Korzystanie z edytora Flow Chart

Edytor graficzny grafów Przepływowych (Flow Chart) ISaGRAF pozwala użytkownikowi na wprowadzanie kompletnych programów z grafami przepływowymi wraz z operacjami i testami (decyzjami) zaprogramowanymi w języku ST, IL lub LD. Graf przepływowy to diagram decyzyjny, który może być również wykorzystany do opisu operacji sekwencyjnych, ponieważ oferuje pewne funkcje zaawansowane, takie jak skoki wstecz, które nie powodują blokowania.

A.5.1 Podstawy języka FC

Flow Chart (**FC**) to język graficzny służący do opisu **operacji sekwencyjnych**. Schemat grafu przepływowego tworzą **Operacje** i **Testy**. Pomiędzy Operacjami a testami występują **łącza zorientowane,** reprezentujące przepływ danych. Poniżej przedstawiono komponenty graficzne języka Flow Chart:

Rozpoczynanie grafu FC: Na początku programu Flow Chart musi znajdować się symbol "początek". Jest on niepowtarzalny i nie może zostać pominięty. Reprezentuje on stan początkowy grafu w momencie jego uruchomienia.

Zakończenie grafu FC: Symbol "koniec" musi znajdować się na końcu programu Flow Chart. Jest on niepowtarzalny i nie może zostać pominięty. Możliwe jest, iż do symbolu "Koniec" nie jest poprowadzone żadne połączenie (graf z pętlą wieczną), jednak pomimo tego symbol "Koniec" musi zostać umieszczony na końcu grafu. Reprezentuje on stan końcowy grafu po zakończeniu jego wykonania.

Połączenia FC: Połączenie to linia reprezentująca przepływ pomiędzy dwoma punktami schematu. Połączenie jest zawsze zakończone strzałką. Z danego punktu źródłowego nie mogą wychodzić dwa połączenia.

Operacje FC: Symbol operacji reprezentuje działania, które mają zostać wykonane. Operacja jest identyfikowana przy pomocy numeru i nazwy. Dwa różne obiekty tego samego grafu nie mogą mieć takiej samej nazwy ani numeru logicznego. Operacje mogą być programowane w takich językach, jak ST, LD lub IL. Operacja jest zawsze powiązana z połączeniami: jednym dochodzącym do niej i jednym z niej odchodzącym.

Testy FC: Test reprezentuje wielkość binarną. Test jest identyfikowany przy pomocy numeru i nazwy. W zależności od oceny przyłączonego wyrażenia ST, LD lub IL, przepływ kierowany jest na ścieżkę "TAK" lub "NIE". Gdy jest używany język tekstowy ST, wyrażenia mogą być opcjonalnie zakończone średnikiem. W przypadku używania języka LD, wartość stanu reprezentuje charakterystyczna cewka.

Podprogram FC: System umożliwia opisywanie hierarchicznej struktury programów FC. Programy FC są zorganizowane w postaci drzewa. Każdy program FC może wywołać inny program FC. Program taki jest zwany programem potomnym programu FC, który go wywołał. Programy FC wywołujące inne podprogramy FC są zwane programami-rodzicami. Programy FC są połączone z sobą w formie głównym drzewie hierarchicznego w oparciu o relację rodzic-potomek. Symbol podprogramu w Flow Chart reprezentuje wywołanie podprogramu Flow Chart. Wykonywanie programu wywołującego FC jest zawieszane do chwili zakończenia wykonywania podprogramu.

Operacja FC charakterystyczna dla We/Wy: Symbol operacji charakterystycznej dla We/Wy reprezentuje działania, które mają zostać wykonane. Podobnie jak inne operacje, operacje charakterystyczne dla We/Wy są identyfikowane numerem i nazwą. W operacjach standardowych i operacjach charakterystycznych dla We/Wy stosuje się tę samą składnię. Celem operacji charakterystycznych dla We/Wy jest jedynie zwiększenie czytelności grafu i skupienie się na nieprzenoszalnych elementach grafu. Stosowanie operacji charakterystycznych dla We/Wy stanowi funkcję opcjonalną. Bloki charakterystycznych dla We/wy zachowują się dokładnie tak samo, jak operacje standardowe.

Łączniki FC: Łączniki są używane do reprezentowania połączenia pomiędzy dwoma punktami schematu bez jego kreślenia. Łącznik jest przedstawiany jako koło i jest dołączony do źródła przepływu. Rysowanie łącznika odbywa się, po właściwej stronie (w zależności od kierunku przepływu danych) przez wskazanie punktu docelowego (generalnie nazwy symbolu docelowego). Łącznik zawsze jest skierowany do zdefiniowanego symbolu FC. Symbol docelowy jest identyfikowany numerem logicznym.

Komentarze FC: Blok komentarza zawiera tekst, bezsensowny z punktu widzenia składni grafu. Można go wstawić w dowolnym, wolnym miejscu okna dokumentu Flow Chart i jest on używany do dokumentowania programu.

A.5.2 Wprowadzenie grafu przepływowego

Aby wprowadzić graf należy w polu graficznym umieścić elementy (operacje, testy decyzyjne, łączniki, itp.) i wykreślić pomiędzy nimi połączenia.

Wstawianie obiektów

Aby wstawić obiekt do wykresu, należy wybrać odpowiadający mu przycisk na pasku narzędzi i kliknąć miejsce w polu graficznym, w którym ma zostać wstawiony. Element można wstawić w pustym polu lub na grafie przepływowym, klikając połączenie. Wstawianie połączenia jest dozwolone tylko w przypadku połączeń pionowych góra-dół.

Wstawić można następujące elementy podstawowe:

operacja zaprogramowana w ST, IL lub LD

operacja charakterystycznych dla We/Wy (podświetla określoną operację nieprzenoszalną)

test (decyzja) zaprogramowana w ST, IL lub LD

łącznik

wywołanie podprogramu FC

komentarz (tekst opisowy)

Edytor Flow Chart ISaGRAF oferuje również listę klasycznych struktur grafu przepływowego. Struktury takie mogą być wstawiane jedynie na istniejącym połączeniu. Nie można wstawiać ich w pustym miejscu:

If / Then / Else (wybór warunkowy)

Repeat until (wykonuj pętlę aż do....warunek będzie prawdziwy)

While (wykonywanie pętli dopóki warunek jest prawdziwy)

Wybór obiektów

Większość komend edycyjnych wymaga wybrania obiektów graficznych. Edytor graficzny FC ISaGRAF pozwala na wybór jednego lub kilku obiektów znajdujących się w obszarze grafu. Aby wybrać obiekty, na pasku narzędzi edytora musi być zaznaczona (włączona) opcja "**Wybierz**" (klawisz ze strzałką). Aby wybrać obiekt, użytkownik musi jedynie kliknąć jego symbol.

Aby wybrać listę obiektów, należy przeciągnąć myszą po grafie kreśląc prostokąt. Wszystkie obiekty graficzne w obrębie zaznaczonego prostokąta zostają oznaczone jako "**wybrane**".

Wybrany obiekt ma kolor ciemnoniebieski i niewielkie czarne kwadraciki wokół swojego symbolu graficznego. Możliwe jest również dodawanie i usuwanie jednego obiektu spośród kilku zaznaczonych poprzez kliknięcie go przy naciśniętym klawiszu **Shift** lub **Ctrl**.

Dokonanie nowego wyboru usuwa zaznaczenie z wszystkich wcześniej wybranych obiektów. Aby usunąć bieżące zaznaczenie wystarczy kliknąć myszą pusty obszar na zewnątrz prostokąta otaczającego wybrane obiekty.

Aby dokonać pojedynczego wyboru, można użyć klawiszy ze strzałkami i przenieść wybór z jednego obiektu grafu na inny. Można również wybrać połączenia.

Wstawianie komentarzy

Komentarze mogą być wstawiane w dowolnej pustej części schematu. Komentarze nie wpływają na wykonywanie programu. Zapewniają one lepszą czytelność

schematy. Aby wstawić blok komentarza, należy użyć odpowiedniego przycisku na pasku narzędzi oraz kliknąć to miejsce schematu, w które ma zostać wstawiony komentarz. Dwukrotne kliknięcie komentarza pozwala wprowadzić / zmienić jego tekst. Przy wprowadzaniu tekstu bloku komentarza nie są wymagane żadne specjalne znaki wiodące czy kończące, takie jak "**(***" i "***)**". Wymiary bloku komentarza można zmieniać po jego wybraniu przez rozciągnięcie rogów jego obramowania.

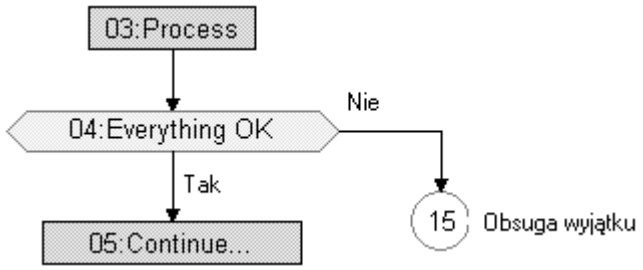
Rysowanie połączeń

Wybranie powyższego przycisku z paska narzędzi pozwala rysować połączenia pomiędzy istniejącymi elementami. Połączenie musi być zawsze rysowane zgodnie z kierunkiem przepływu. Aby wstawić połączenie, należy najpierw wybrać nie powiązany z niczym punkt wyjściowy elementu FC, a następnie przeciągnąć go myszą do punktu docelowego. Punktem docelowym może być górna część (punkt wejściowy) nie powiązanego elementu FC lub dowolne miejsce istniejącego już połączenia. Punkty zbiegania się połączeń są przedstawione w Flow Chart jako niewielkie, szare kółka. Punkty zbiegania się mogą również być wybierane i przesuwane w celu uporządkowania diagramu.

Używanie łączników

Edytor Flow Chart ISaGRAF pozwala wykorzystać łączniki graficzne do zastąpienia widocznych połączeń. Łączniki umożliwiają unikanie bardzo długich połączeń i zwiększają czytelność grafu . Łącznika nie można wykorzystać do ustanowienia połączenia z innym programem FC.

Łącznik wstawia się do grafu tak, jak inne obiekty FC. Reprezentuje go kółko zawierające liczbowy odsyłacz do elementu docelowego (kończącego połączenie). Obok kółka łącznika wyświetlany jest krótki tekst opisujący element docelowy.



Przemieszczanie obiektów

Aby przemieścić obiekty w grafie, należy je wybrać i przeciągnąć przy pomocy myszy w ramach grafu. Przemieszczać można pojedynczy element lub kilka wybranych elementów. Przemieszczane elementy nie mogą się nakładać.

Przemieszczania elementów nie można wykorzystać do powiązania ich z istniejącym połączeniem.

Kiedy przemieszczany jest pojedynczy element (operacja, test, itp.) edytor Flow Chart ISaGRAF przemieszcza automatycznie wraz z nim wszystkie elementy znajdujące się pod nim i połączone z nim. Funkcja ta nie działa w przypadku wybrania kilku elementów.

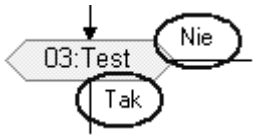
Zmiana rozmiarów obiektów

Rozmiar dowolnego elementu graficznego w grafie przepływowym oprócz symboli "Początek", "Koniec" i łączników może być swobodnie zmieniany. Aby zmienić rozmiar elementu, najpierw trzeba go wybrać, a następnie przeciągnąć myszą niewielkie kwadraty znajdujące się na jego obrzeżach.

Kiedy element jest powiązany z połączeniem, zmiana jego rozmiarów poziomych działa zarówno na prawą, jak i na lewą jego krawędź tak, że po zmianie rozmiaru element pozostaje prawidłowo wypośrodkowany względem połączenia.

Zamiana wyjść testu

Umiejscowienie wyjść TAK / NIE w teście (decyzji) może być zamieniane. Aby tego dokonać wystarczy kliknąć dwukrotnie oznaczenie "Tak" lub "Nie" wyświetlane w pobliżu symbolu testu.



A.5.3 Praca na istniejącym grafie

Komend menu "**Edycja**" używa się do zmiany i uzupełnienia istniejącego schematu. Większość z tych komend działa na elementach aktualnie wybranych na schemacie.

Poprawianie grafu

Klawisz DEL może być używany do usuwania wybranych elementów. Połączenia nie zakończone są usuwane wraz z wybranymi elementami. Do przywrócenia elementów po komendzie DEL używa się komendy "**Edycja / Cofnij**". Komenda DEL może zostać zastosowana również do grupy elementów wybranych na schemacie. Komendy "**Wytnij**", "**Kopiuj**", "**Wklej**" menu "**Edycja**" stosuje się do przemieszczania lub kopiowania wybranych elementów.

Znajdywanie i zamiana

Komendy menu "**Edycja / Znajdź / Zamień**" są stosowane do wyszukiwania i zamiany ciągów tekstu w całym programie (wszystkich operacjach i testach napisanych w ST, IL lub LD). Okno dialogowe Znajdź/Zamień używane jest do wprowadzania szukanego tekstu oraz do bezpośredniego otwierania sekcji programu, w których znaleziono ten tekst.

Bezpośredni dostęp do elementu

Komenda "**Edycja / Idź do**" pozwala użytkownikowi na dostęp do istniejącego w grafie elementu graficznego. Pozycja przewijania zostaje automatycznie zaktualizowana tak, iż element staje się widoczny. Wyszukany element zostaje wybrany.

Ponowne numerowanie elementów

Komenda "**Edycja / Numeruj ponownie**" pozwala użytkownikowi ponownie ponumerować elementy grafu przepływowego. Każdy umieszczony w grafie element jest identyfikowany przy użyciu charakterystycznego numeru odniesienia. Numery odniesienia są alokowane przez edytor po każdym wstawieniu nowego elementu. "**Numeruj ponownie**" pozwala na skorygowanie numeracji elementów, zgodnie z ich umiejscowieniem w grafie. Numerowanie odbywa się narastająco z góry w dół i z lewej na prawą.

A.5.4 Wprowadzanie programów poziomu 2

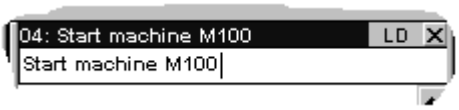
Aby wstawić program poziomu 2, użytkownik musi dwukrotnie kliknąć symbol operacji lub testu. Program poziomu 2 jest wyświetlany po prawej stronie okna FC. Linia podziału pomiędzy grafem FC a programem poziomu 2 może być dowolnie przemieszczana. Można otworzyć jednocześnie jeden lub dwa pola poziomu 2. Następujące komendy są dostępne z klawiatury, za pośrednictwem myszy, lub z menu "Edycja":

	Klawiatura	***Mysz***	***Menu "Edycja"***
Otwórz w ostatnim oknie domyśl.	Enter	Dwukrotne kliknięcie	Edycja poziomu 2
Otwórz w osobnym oknie	Ctrl+Enter	Ctrl + Dwukrotne klik.	Edycja poziomu 2 w osobnym oknie

Kiedy widoczne są dwa okna poziomu 2, linię podziału pomiędzy nimi można swobodnie przemieszczać. Przycisk po prawej stronie paska tytułowego poziomu 2 jest używany do zamykania okna poziomu 2.

Językiem domyślnym programowania Poziomu 2 jest **ST** (Tekst Strukturalny). Językiem programowania może być również **IL** lub **LD**. Nazwa wybranego języka jest wyświetlana w niewielkim oknie na pasku tytułowym poziomu 2. Aby zmienić aktywny język, należy wybrać komendę "**Opcje / Ustaw język poziomu 2**" z menu lub kliknąć to oknie. Komenda ta może być stosowana tylko wtedy, gdy okno programowania poziomu 2 jest puste.

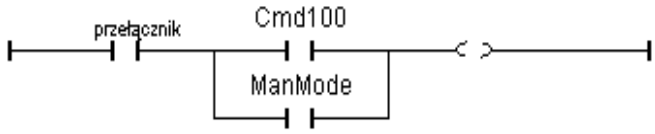
W górnej części okna poziomu 2 pojawia się pojedyncza linia edycyjna. Używana ona jest do wprowadzenia krótkiego tekstu opisu. Tekst będzie wyświetlany jako komentarz IEC przy rysowaniu symboli FC. Jest on bardzo przydatny, ponieważ jest także używany przez inne komendy takie, jak "Idź do..." oraz przy drukowaniu operacji i testów FC do dokumentu.



Komenda "**Opcje / Odśwież**" może być użyta w dowolnym momencie, kiedy otwarte są okna poziomu 2 do uzupełnienia głównego grafu przepływowego o zmodyfikowane programy poziomu 2.

A.5.5 Programowanie w LD

Edytor LD służy do programowania poziomu 2. W przypadku testu decyzyjnego, schemat LD tworzy tylko jeden szczebel z tylko jedną cewką reprezentującą decyzję. Nazwa testu nie jest powtarzana przy symbolu cewki. Poniżej znajduje się przykład test zaprogramowanego w LD.



Przy programowaniu w LD do zmiany elementu wybranego na programowej siatki logicznej używa się klawiszy ze strzałkami, a do wstawienia symbolu stosuje się następujące skróty:

F2:..... wstawia styk za wybranym symbolem / zapoczątkowuje szczebel
F3:..... wstawia styk przed wybranym symbolem
F4:..... wstawia styk równolegle do wybranego symbolu
F5:..... Dodaje cewkę równolegle do już istniejącej (nie jako testy)
F6:..... wstawia blok za wybranym symbolem
F7:..... wstawia blok przed wybranym symbolem
F8:..... wstawia blok równolegle do wybranego symbolu
F9:..... dodaje symbol skoku równolegle do wybranej cewki (nie jako testy)

Skok prowadzi do nazwy szczebla. Nazwę szczebla można wprowadzić naciskając ENTER, kiedy wybrany jest nagłówek szczebla. Edytor ISaGRAF zapamiętuje już wprowadzone etykiety szczebli niezależnie od tego, czy zostały one określone jako nazwa szczebla czy dla operacji skoku. Okno dialogowe "Skok/Etykieta"' pozwala wpisać nową etykietę lub wybrać już istniejącą. W przypadku wprowadzenia nowej nazwy, jest ona automatycznie dodawana do listy. Przycisk "**Usuń**" stosuje się do usuwania wybranej nazwy z listy. Nie usuwa on etykiety z wybranego na schemacie szczebla. Aby to uczynić, należy nacisnąć **OK,** kiedy okno edycyjne jest puste.

Zamiast naciskać klawisze funkcyjne, można nacisnąć przyciski na pasku narzędzi LD.

Aby wybrać zmienną lub wprowadzić wartość stałą, należy po wybraniu styku lub parametru We/Wy bloku nacisnąć klawisz ENTER. Aby wybrać typ bloku funkcyjnego, należy po wybraniu bloku funkcyjnego nacisnąć ENTER. Ten sam skutek przynosi dwukrotne kliknięcie danego symbolu.

Aby zmienić typ styku lub cewki (bezpośredni, zanegowany), należy po wybraniu styku nacisnąć Ctrl + SPACJA. Więcej szczegółów na temat możliwościach LD podano w rozdziale "Korzystanie z edytora LD" w tym dokumencie.

A.5.6 Opcje wyświetlania

Komenda "**Opcje / Układ**" otwiera okno dialogowe, w którym zebrane są wszystkie parametry i opcje dotyczące obszaru roboczego edytora oraz rysowania schematu. Pól kontrolnych w oknie grupowym "Obszar roboczy" używa się do pokazania lub ukrycia paska narzędzi edytora oraz paska stanu. Opcje okna grupowego "Dokument" pozwalają na pokazanie lub ukrycie punktów siatki edycyjnej oraz na wyświetlanie grafu na biało, na czarno lub kolorowo.

Przycisk "Powiększenie" na pasku narzędziowym pozwala na zmianę aktualnego wskaźnika powiększenia. Komenda ta jest również dostępna podczas opracowywania programu LD powiązanego z operacją lub testem.

Przycisku "Siatka" na pasku narzędzi używa się do pokazywania lub ukrywania punktów siatki edycyjnej. Komenda ta dostępna jest również podczas opracowywania programu LD powiązanego z operacją lub testem.

Komendy "**Opcje / Czcionka**" używa się do wyboru nazwy czcionki, która będzie używana we wszystkich dokumentach ISaGRAF. W przypadku wywołania z bloku ST lub IL można określić rozmiar czcionki. Przy wyborze czcionki do widoku graficznego (FC lub LD), jej styl i rozmiar nie są istotne i nie muszą być ustawiane. Edytory graficzne ISaGRAF zawsze obliczają rozmiar czcionki w zależności od wybranego wskaźnika powiększenia.

A.6 Korzystanie z edytora LD

Język LD pozwala na graficzną reprezentację zmiennych binarnych. Operatory logiczne AND, OR, NOT są wyrażane przez topologię diagramu. Zmienne wejściowe binarne są skojarzone ze stykami graficznymi. Wyjściowe zmienne binarne są skojarzone z cewkami graficznymi. Edytor LD ISaGRAF umożliwia łatwe wprowadzanie diagramu LD przy użyciu klawiatury lub myszy. Elementy są przez edytor LD automatycznie łączone i układane na szczeblach. Żadne z połączeń nie jest rysowane ręcznie przez użytkownika. Edytor LD układa również szczeble w diagramach zapewniając optymalizację przestrzeni zajmowanej przez diagram.

A.6.1 Podstawy języka LD

Program LD ma formę listy **szczebli**, na których są rozmieszczone styki i cewki. Poniżej podano podstawowe komponenty diagramu LD:

Nagłówek szczebla (lewa szyna zasilająca)

Każdy szczebel rozpoczyna się od lewej szyny zasilającej, która reprezentuje wstępną wartość "PRAWDA". Edytor LD ISaGRAF automatycznie tworzy lewą szynę zasilającą, kiedy użytkownik umieszcza pierwszy styk szczebla. Każdy szczebel może posiadać nazwę logiczną, która może zostać wykorzystana jako etykieta w instrukcjach skoku.

Styki

Styk zmienia przepływ danych binarnych zgodnie ze stanem zmiennej binarnej. Nazwa tej zmiennej jest wyświetlana nad symbolem styku. Edytor LD ISaGRAF obsługuje następujące typy styków:

 ..... styk bezpośredni

..... styk zanegowany

 ..... styk z wykrywaniem dodatniego (narastającego) zbocza sygnału

 ..... styk z wykrywaniem ujemnego (opadającego) zbocza sygnału

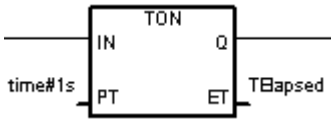
Cewki

Cewka przedstawia działanie. Stan szczebla (stan połączenia na lewo od cewki) jest wykorzystywany do wymuszenia zmiennej binarnej. Nazwa zmiennej jest wyświetlana ponad symbolem cewki. Edytor LD ISaGRAF obsługuje następujące typy cewek:

-( )- ..... cewka bezpośrednia
-(\)- ..... cewka zanegowana
-(S)- ..... cewka działania "ustaw"
-(R)- ..... cewka działania "resetuj"
-(P)- ..... cewka z wykrywaniem dodatniego (narastającego) zbocza sygnału
-(N)- ..... cewka z wykrywaniem ujemnego (opadającego) zbocza sygnału

Bloki funkcyjne

Blok na diagramie LD może przedstawiać funkcję, blok funkcyjny, podprogram lub operator. Jego pierwsze parametry wejściowe i wyjściowe są zawsze skojarzone ze szczeblem. Inne parametry wejściowe i wyjściowe są zapisane tekstem poza prostokątem bloku.



Koniec szczebla (prawa szyna zasilająca)

Szczebel kończy się prawą szyną zasilającą. Przy używaniu edytora LD, prawa szyna zasilająca jest wstawiana po umieszczeniu przez użytkownika cewki.

Symbol skoku

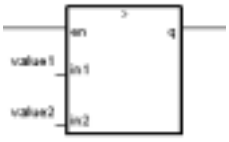
Symbol skoku odnosi się zawsze do etykiety szczebla, tj. nazwy szczebla zdefiniowanej w jakimś miejscu tego samego diagramu LD. Jest on umieszczany na końcu szczebla. Gdy szczebel ma stan PRAWDA, to sterownie programu przekazywane jest skokiem bezpośrednio do wskazanego przez skok szczebla. Należy pamiętać, że skoki do tyłu są niebezpieczne, ponieważ w niektórych przypadkach mogą doprowadzić do zablokowania się pętli. PLC.

Symbol powrotu

Symbol powrotu jest umieszczany na końcu szczebla. Jeśli stan szczebla ma wartość PRAWDA wykonywanie programu zostaje zakończone.

Wejście "EN"

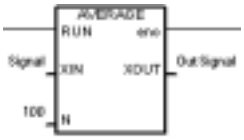
W przypadku niektórych operatorów, funkcji lub bloków funkcyjnych, pierwsze wejście nie wymaga danych binarnych. Ponieważ pierwsze wejście musi być zawsze połączone ze szczeblem, to inne wejście zwane "**EN**" jest automatycznie wstawiane na pierwszą pozycję. Blok jest wykonywany jeśli wejście **EN** ma wartość PRAWDA. Poniżej znajduje się przykład operatora porównywania oraz odpowiadający mu kod wyrażony w ST:




```
IF rung_state THEN
   q := (value1 > value 2);
ELSE
   q := FALSE;
END_IF;
(* kontynuacja szczebla ze stanem
       q *)
```

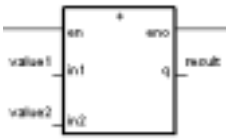
Wyjście "ENO"

W przypadku niektórych operatorów, funkcji lub bloków funkcyjnych, pierwsze wyjście nie wymaga danych binarnych. Ponieważ pierwsze wyjście musi być zawsze połączone ze szczeblem, inne wyjście zwane "**ENO**" jest wstawiane automatycznie na pierwszą pozycję. Wyjście **ENO** zawsze przyjmuje taką samą wartość jak pierwsze wejście bloku. Poniżej znajduje się przykład bloku funkcyjnego AVERAGE oraz odpowiadający mu kod zapisany w ST:




```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(* kontynuacja szczebla ze stanem
       eno *)
```

W niektórych przypadkach wymagane są zarówno **EN** jak i **ENO**. Poniżej podano przykład operatora arytmetycznego oraz odpowiadający mu kod wyrażony w ST:




```
IF rung_state THEN
   result := (value1 + value2);
END_IF;
eno := rung_state;
(* kontynuacja szczebla ze stanem
       eno *)
```

Ograniczenia edytora LD

Edytor LD ISaGRAF nie pozwala na kontynuację szczebla (wstawianie innych styków czy cewek) na prawo od cewki. Jeżeli z jednego szczebla ma prowadzić kilka wyjść, odpowiadające im cewki muszą zostać narysowane równolegle.

A.6.2 Wprowadzanie diagramu LD

Wszystkie komendy edycyjne edytora LD można wywołać za pomocą klawiatury lub myszy.

Siatka edycyjna

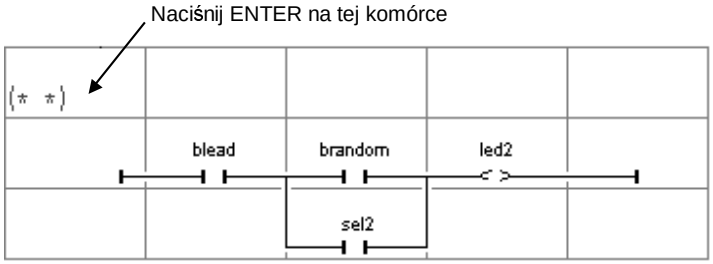
Diagram LD jest wprowadzany do macierzy logicznej. Każda komórka macierzy może zawierać maksymalnie jeden symbol LD. Do zmiany aktualnie wybranego elementu używa się klawiszy ze strzałkami lub klika się myszą. Wybrana komórka jest zaznaczona odwróceniem kolorów. Do niektórych operacji wycinania/kopiowania/wklejania, możliwe jest wybranie kilku komórek. Aby tego dokonać przy pomocy myszy, wystarczy przeciągnąć kursor myszy po diagramie. W przypadku klawiatury używa się klawiszy ze strzałkami przy naciśniętym klawiszu SHIFT.

Rozpoczynanie nowego szczebla

Aby dodać do diagramu nowy szczebel, należy wybrać element występujący poniżej ostatniego istniejącego szczebla i wstawić styk (nacisnąć klawisz F2 lub kliknąć odpowiedni przycisk paska narzędzi LD). Zostanie stworzony nowy szczebel z jednym stykiem i jedną cewką.

Wpisywanie komentarza do szczebla

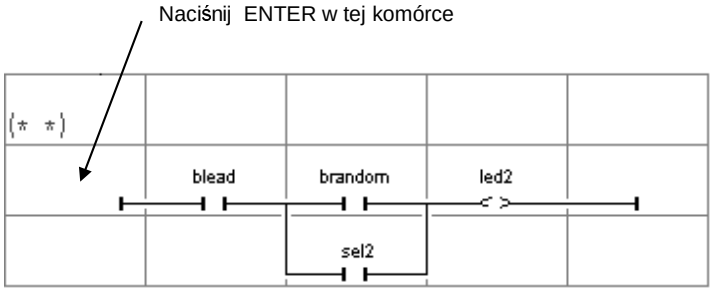
Każdy szczebel może być dokumentowany maksymalnie dwoma liniami tekstu. Aby wprowadzić tekst komentarza do szczebla, należy wybrać komórkę nad szczeblem i nacisnąć klawisz ENTER, lub też dwukrotnie kliknąć tę komórkę myszą:



Wprowadzanie etykiety szczebla

Każdy szczebel może być rozpoznawany poprzez nazwę. Nazwa ta może być również używana jako etykieta przeznaczenia w operacjach skoku. Aby wprowadzić

lub zmienić etykietę szczebla należy wybrać nagłówek szczebla oraz nacisnąć klawisz ENTER, lub też dwukrotnie kliknąć tę komórkę myszą:



Edytor LD ISaGRAF zapamiętuje wprowadzone etykiety szczebli niezależnie od tego, czy służą one jako nazwy szczebli, czy do operacji skoku. Okno dialogowe "Etykieta skoku" pozwala na wpisanie nowej etykiety lub na wybranie istniejącej.

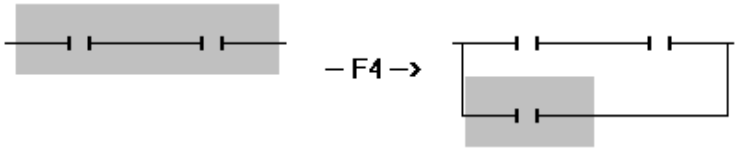
Jeśli wprowadzi się nową nazwę, zostanie ona automatycznie dodana do listy. Przycisk "**Usuń**" jest używany do usuwania wybranej nazwy z listy. Nie usuwa on etykiety szczebla wybranej na diagramie. Aby to uczynić wystarczy nacisnąć **OK**, kiedy okno edycyjne jest puste.

Wstawianie symboli na szczeblu

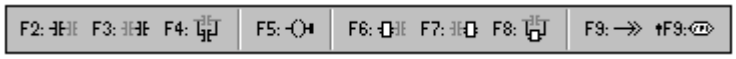
Wstawiania symboli (styków, cewek, bloków, itp.) na istniejącym szczeblu dokonuje się zawsze odpowiednio do aktualnego wyboru. Należy wybrać właściwą komórkę w obrębie szczebla i nacisnąć jeden z następujących klawiszy funkcyjnych w celu wstawienia:
F2..... styku przed wybranym symbolem (po lewej)
F3..... syku po wybranym symbolu (po prawej)
F4..... styku równolegle do wybranego symbolu
F6..... bloku przed wybranym symbolem (po lewej)
F7..... bloku po wybranym symbolu (po prawej)
F8..... bloku równolegle do wybranego symbolu

Kiedy zaznaczone jest wyjście szczebla (cewka), możliwe są następujące wybory:
F5..... dodaj cewkę równolegle do istniejącej
F9..... dodaj symbol "Skok" równolegle do istniejącego
Shift + F9 ..... dodaj symbol "Powrót" równolegle do istniejącego

Przy wstawianiu równoległym (F4/F8), jeżeli zaznaczonych jest kilka styków szczebla, symbol jest wstawiany równolegle do grupy zaznaczonych elementów. Oto przykład:



Do wstawiania symboli na diagramie można użyć również komend menu "**Wstaw**". Przy pomocy myszy można kliknąć pasek narzędzi LD na typie symbolu, który ma zostać wstawiony:



Wprowadzanie symboli

Aby skojarzyć symbol zmiennej ze stykiem lub cewką, należy dokonać wyboru i nacisnąć ENTER. Przy pomocy myszy można dwukrotnie kliknąć styk lub cewkę. Pojawia się okno wyboru zmiennej. Aby uzyskać dalsze informacje dotyczące wykorzystania tego okna, należy zapoznać się z rozdziałem "Dalsze informacje na temat edytorów programów" w tym dokumencie. Aby skojarzyć funkcję, blok funkcyjny lub operator z blokiem należy nacisnąć ENTER gdy kursor znajduje się wewnątrz prostokąta. Aby skojarzyć symbol zmiennej z parametrem wejściowym lub wyjściowym bloku kursor musi znajdować się na odpowiednim miejscu **poza** prostokątem bloku.

Okna dialogowe zawierające listy wyboru zmiennej lub bloku są zazwyczaj używane do wprowadzania tekstu. Jeśli w menu "**Opcje**" wybrany jest tryb "**Dane z klawiatury**", symbole zmiennych i nazwy bloków są wprowadzane bezpośrednio do pojedynczego tekstowego okna edycyjnego. Wprowadź nowy tekst i naciśnij klawisz "**Enter**", aby go potwierdzić, lub klawisz "**Esc**", aby zaniechać modyfikacji i zamknąć tekstowe okno edycyjne. Tekstowe okno edycyjne używane w trybie "wprowadzania ręcznego z klawiatury" nie może zostać zamknięte przy pomocy myszy.

Zmiana typów styków i cewek

"**Edycja / Zmień typ cewki/styku**" zmienia typ wybranego styku lub cewki. Styk może być bezpośredni, zanegowany, z wykrywaniem dodatniego lub ujemnego zbocza sygnału. Cewka może być bezpośrednia, zanegowana, ustawiająca lub kasująca, z wykrywaniem dodatniego lub ujemnego zbocza sygnału. Ten sam skutek odnosi naciśnięcie klawisza SPACJI.

Wstawianie szczebla do diagramu

Komenda "**Edycja / Wstaw szczebel**" wstawia do diagramu nowy szczebel przed szczeblem zaznaczonym. Inicjowany szczebel zawiera jeden styk i jedną cewkę.

A.6.3 Praca na istniejącym diagramie

Komend menu "**Edycja**" używa się do zmiany i uzupełnienia istniejącego diagramu. Większość tych komend działa na elementach aktualnie wybranych na diagramie.

Poprawianie diagramu

Klawisz DEL (komenda ‘Kasuj’) może być wykorzystywany do usuwania wybranych elementów. Niemożliwe jest usuwanie symboli cewek, skoków lub powrotów, kiedy stanowią one jedyne zakończenie szczebla. Komendy "**Edycja / Cofnij**" używa się do przywracania elementów po komendzie DEL. Komenda ‘Kasuj’ może zostać zastosowana również do grupy elementów wybranych na diagramie. Komendy ‘Kasuj’ można używać także do usunięcia tekstu komentarza do szczebla, jeśli tekst ten jest wybrany. Gdy wybrany jest nagłówek szczebla, to komenda ‘Kasuj’ usuwa cały szczebel.

Kopiowanie symboli

Komendy "**Wytnij**", "**Kopiuj**", "**Wklej**" w menu "**Edycja**" są stosowane do przemieszczania lub kopiowania zaznaczonych elementów. Komendy te nie działają w odniesieniu do komentarzy do szczebli. Komenda "**Edycja / Wklej specjalnie**" daje wybór wstawienia elementu

- przed wybranym elementem (po lewej)
- po wybranym elemencie (po prawej)
- równolegle z wybranym elementem

Zarządzanie całymi szczeblami

Wszystkie komendy edycyjne (kasuj, wytnij, kopiuj, itp.) dotyczą całego szczebla, jeśli wybrany jest nagłówek szczebla (lewa szyna zasilająca). Pozwala to w prosty sposób układać szczeble na diagramie jedynie przez zmianę wybranej pozycji w pierwszej kolumnie. Możliwe jest również rozszerzenie wyboru w pionie tak, aby wybranych zostało kilka nagłówków szczebli. W takim przypadku komendy edycyjne można stosować do całej listy szczebli.

Znajdywanie i zamiana

Komendy menu "**Edycja / Znajdź**" i "**Edycja / Zamień**" są stosowane do wyszukiwania i zamiany tekstu na diagramie. Wyszukiwać można tylko pełne nazwy. Wyszukiwanie działa w obrębie styków, cewek, nazw bloków, parametrów bloków oraz etykiet. Nie może ono być używane do odnajdywania łańcuchów tekstowych w komentarzach do szczebli. Komendy Zamień nie można wykorzystać do zmiany typu bloku. Wyszukiwanie może być wykonywane w górę lub w dół

rozpoczynając w aktualnie wybranej pozycji. Kiedy osiągnięte zostaną granice diagramu, wyszukiwanie kontynuowane jest od początku diagramu. Do szybkiego wyszukiwania nazw zmiennych można użyć następujących skrótów:

ALT + F2 znajduje następny element, którego zmienna ma taką samą nazwę, jak element aktualnie wybrany. Funkcję tę można również wykorzystać do bloków funkcyjnych i etykiet szczebli.

ALT + F5 znajduje następną cewkę, której zmienna ma taką samą nazwę, jak aktualnie wybrany element. Cecha ta jest głównie wykorzystywana w trybie debagowania do szybkiego odnalezienia szczebla, który wymusza stan podejrzanej zmiennej.

A.6.4 Opcje wyświetlania

Komendy menu "**Opcje**" są wykorzystywane do dostosowania sposobu rysowania diagramu LD na ekranie oraz do ukrywania lub pokazywania różnych informacji.

Komentarze do szczebli

Komendy "**Opcje / Komentarze do szczebli**" używa się od ukrywania lub pokazywania komentarzy do szczebli na całym diagramie. Ukrycie komentarzy do szczebli może być wymagane do uzyskania bardziej spójnego obrazu olbrzymiego diagramu, ponieważ każdy komentarz zajmuje jeden wiersz macierzy edycji. Opcja ta nie wpływa na zawartość istniejących komentarzy do szczebli i może być włączana lub wyłączana w dowolnej chwili.

Nazwy i ich zamienniki

Każda zmienna skojarzona ze stykiem, cewką lub parametrem We/Wy bloku jest identyfikowana przy pomocy symbolicznej nazwy. Edytor LD ISaGRAF wprowadza również pojęcie "**zamiennika nazwy**" dla każdej zmiennej. Zamiennik nazwy zmiennej to tekst komentarza do zmiennej, obcięty przed pierwszym znakiem '**:**' i ograniczony do 16 znaków. Oto przykłady:

komentarz do zmiennej:	*zamiennik nazwy:*
krótki tekst	krótki tekst
długi tekst bez separatora	długi tekst bez s
krótki tekst: długi opis	krótki tekst

Zamienniki nie mają wpływu na wykonywanie diagramu LD i ze składniowego punktu widzenia należy je uznać za komentarze. Zamiennik nazwy zmiennej jest automatyczne wyciągany z komentarza do zmiennej, kiedy jej nazwa zostaje wybrana z listy zmiennych. Nie można go zmienić ręcznie.
Aby wybrać tryb wyświetlania identyfikatora zmiennej, wykorzystuje się komendę "**Opcje / Styki i cewki**". Dostępne są następujące tryby:

- wyświetlaj tylko nazwy zmiennych

- wyświetlaj tylko zamienniki nazw zmiennych
- wyświetlaj zarówno nazwy jak i ich zamienniki

Edytor LD nie aktualizuje automatyczne dokumentów LD po zmianie zamienników nazw zmiennych w słowniku. Aby zaktualizować wszystkie zamienniki nazwy na edytowanym diagramie, należy wykorzystać komendę "**Opcje / Styki i cewki / Uaktualnij zamienniki nazw**". Można również ustawić opcję "**Aktualizuj przy otwarciu**" z menu "**Opcje / Styki i cewki**" aby wydać ISaGRAF polecenie automatycznego aktualizowania zamienników podczas każdego otwarcia programu LD. Uwaga: Ustawienie tej opcji może w znaczący sposób wydłużyć czas otwierania programu.

Opcje rysowania

Komenda "**Opcje / Ustawienia widoku**" otwiera okno dialogowe, w którym zebrane są wszystkie parametry i opcje dotyczące obszaru roboczego edytora oraz rysowania diagramu graficznego LD.

Pól kontrolnych w oknie grupy "Paski" używa się do pokazania lub ukrycia paska narzędzi edytora, paska statusu oraz paska narzędzi LD. Opcje okna grupy "Obszar roboczy okna" pozwalają na pokazanie lub ukrycie punktów siatki edycyjnej oraz na włączenie/wyłączenie kolorowego zobrazowania.

Opcje okna grupy "Powiększenie" pozwalają na wybór ogólnego wskaźnika powiększenia. Można również wykorzystać przycisk "powiększenie" na pasku narzędzi w celu przełączania domyślnych wskaźników powiększania.

Można również dostosować stosunek szerokości do wysokości (X/Y) komórek siatki edycyjnej. Ta ostatnia opcja może zostać wykorzystana do zmniejszenia domyślnej szerokości komórki, jeżeli zazwyczaj używane są krótkie nazwy zmiennych. Można również użyć przycisku "szerokość celi" na pasku narzędzi edytora, aby zmienić stosunek X/Y bez wchodzenia do okna dialogowego Ustawienia widoku.

Komendy "**Opcje / Czcionka**" używa się do wyboru nazwy czcionki, która ma być używana we wszystkich dokumentach graficznych ISaGRAF. Podczas wyboru czcionki jej styl i rozmiar nie są istotne i nie muszą być ustawiane. Edytory graficzne ISaGRAF zawsze obliczają rozmiar czcionki w zależności od wybranego stosunku powiększenia.

A.7 Korzystanie z edytora FBD/LD

Edytor graficzny FBD/LD ISaGRAF pozwala użytkownikowi na wprowadzanie kompletnych programów w języku FBD, które mogą zawierać fragmenty w języku LD. Łączy on możliwości edycji grafiki i tekstu i dzięki temu można wprowadzać zarówno diagramy, jak i odpowiadające im wejścia i wyjścia. Ponieważ edytor ten jest przeznaczony w większym stopniu dla języka FBD, diagramy wyłącznie w LD powinny być wprowadzane raczej przy pomocy edytora LD ISaGRAF.

A.7.1 Podstawy języków FBD/LD

Język **FBD** jest graficzną reprezentacją wielu różnych typów równań. **Operatory** są przedstawiane przy pomocy funkcji reprezentowanych prostokątami. Wejścia funkcji są podłączone do lewej strony prostokąta. Wyjścia funkcji są podłączone do jego prawej strony. Wejścia i wyjścia diagramu (**zmienne**) połączone są z blokami funkcyjnymi przy pomocy **połączeń logicznych**. Wyjście z prostokąta funkcji może być podłączone do wejścia innego prostokąta

Język **LD** pozwala na graficzną reprezentację wyrażeń binarnych. Operatory logiczne **I**, **LUB**, **NIE** są wyraźnie przedstawione w topologii diagramu. Wejściowe zmienne binarne są powiązane z **stykami** graficznymi. Wyjściowe zmienne binarne są powiązane z **cewkami** graficznymi. Styki i cewki są połączone ze sobą oraz z prawą i lewą szyną zasilającą przy pomocy **linii poziomych**. Każdy segment linii posiada stan logiczny **FALSZ** lub **PRAWDA**. Stan logiczny jest taki sam dla wszystkich segmentów połączonych bezpośrednio ze sobą. Każda linia pozioma doprowadzona do lewej **pionowej szyny zasilającej** ma stan **PRAWDA**.

Diagramy LD i FBD są zawsze interpretowane od strony lewej do prawej i z góry na dół. Dalsze szczegóły dotyczące języków LD i FBD znajdują się w Podręczniku Użytkownika Języków ISaGRAF. Poniżej są podane podstawowe elementy graficzne języków LD i FBD, które są obsługiwane przez edytor FBD/LD:

Lewa szyna zasilająca

Z lewej strony szczeble muszą być podłączone do **lewej szyny zasilającej**, która reprezentuje początkowy stan "PRAWDA". Edytor FBD ISaGRAF pozwala również połączyć z lewą szyną zasilającą dowolny symbol logiczny.

Prawa szyna zasilająca

Po prawej stronie cewki mogą być połączone z **prawą szyną zasilającą**. Przy korzystaniu z edytora FBD/LD ISaGRAF jest to funkcja opcjonalna. Jeśli cewka nie

jest połączona z prawej strony, zawiera ona na swoim własnym rysunku prawą szynę zasilającą.

Pionowe połączenie "LUB" w języku LD

Połączenie pionowe w LD dopuszcza kilka połączeń z lewej strony oraz kilka z prawej. Każde połączenie z prawej strony równoznaczne jest z kombinacją LUB połączeń z lewej.

Styki

Styk zmienia przepływ danych binarnych zależnie od stanu zmiennej binarnej. Nazwa zmiennej wyświetlona jest ponad symbolem styku. Edytor FBD/LD ISaGRAF obsługuje następujące rodzaje styków:

-| |-	styk prosty
-|\\\|-	styk zanegowany
-|P|-	styk z wykrywaniem zbocza dodatniego (narastającego)
-|N|-	styk z wykrywaniem zbocza ujemnego (opadającego)

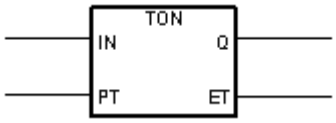
Cewki

Cewka reprezentuje operację. Musi ona być połączona z lewej strony do symbolu binarnego takiego, jak styk. Nazwa zmiennej wyświetlona jest nad symbolem cewki. Edytor FBD/LD ISaGRAF obsługuje następujące rodzaje cewek:

-( )-	cewka prosta
-(\)-	cewka zanegowana
-(S)-	cewka ustawiająca
-(R)-	cewka kasująca

Bloki funkcyjne

Na schemacie FBD blok może reprezentować funkcję, blok funkcyjny, podprogram lub operator. Wejścia i wyjścia muszą być połączone ze zmiennymi, stykami lub cewkami albo wejściami lub wyjściami innych bloków. Nazwy parametrów formalnych są wyświetlone wewnątrz prostokąta oznaczającego blok.



Etykiety

Etykiety mogą być umieszczane w dowolnym miejscu diagramu. Etykiety wykorzystywane są jako punkty docelowe instrukcji skoku, w celu zmiany kolejności wykonywania diagramu. Etykiety nie są połączone z innymi elementami. Zaleca się umieszczanie etykiet po lewej stronie diagramu, aby zwiększyć jego czytelność.

Skoki

Symbol skoku odnosi się zawsze do etykiety umieszczonej w innym miejscu diagramu. Jego lewe połączenie musi prowadzić do punktu binarnego. Jeśli połączenie lewe ma wartość PRAWDA, to z tego miejsca na schemacie jest wykonywany bezpośredni skok do etykiety docelowej. Należy zauważyć, iż skoki wstecz są niebezpieczne, ponieważ w niektórych przypadkach mogą prowadzić do zablokowania PLC w pętli.

Symbol powrotu

Symbol powrotu jest połączony z punktem binarnym. Wskazuje on, iż wykonywanie programu musi zostać zatrzymane, jeśli stanem szczebla jest PRAWDA.

Zmienne

Zmienne są na schemacie reprezentowane wewnątrz niewielkich prostokątów, połączonych z prawej i z lewej strony z innymi elementami diagramu.

Połączenia

Połączenia są rysowane pomiędzy elementami umieszczonymi na schemacie. Połączenia są rysowane zawsze od punktu wyjścia do punktu wejścia (zgodnie z kierunkiem przepływu danych).

Połączenia z negacją logiczną

Niektóre połączenia binarne są reprezentowane przy pomocy niewielkiego kółka na prawym końcu. Oznacza ono negację logiczną informacji przenoszonej przez połączenie.

Rogi zdefiniowane przez użytkownika

Punkty zdefiniowane przez użytkownika mogą być określone na połączeniach. Pozwalają one użytkownikowi na ręczne sterowanie wyborem drogi połączenia.

Jeśli nie umieszczono żadnego rogu, edytor FBD/LD ISaGRAF wykorzystuje domyślny algorytm routingu.

A.7.2 Wprowadzanie diagramu FBD

Aby wprowadzić diagram, należy umieścić elementy (bloki, zmienne, styki, cewki, itp.) w polu graficznym i narysować połączenia między nimi.

Wstawianie obiektów

Aby wstawić obiekt do diagramu, należy wybrać odpowiadający mu przycisk na pasku narzędzi i kliknąć pole graficzne, w które ma zostać wstawiony.

Wybór obiektów

W przypadku większości komend edycyjnych konieczne jest wybranie obiektów graficznych. Edytor graficzny LD/FBD ISaGRAF umożliwia wybór jednego lub większej ilości istniejących obiektów na obszarze diagramu. Aby wybrać obiekt, należy włączyć "**wybór**" (przycisk ze strzałką) na pasku narzędzi edytora. Aby wybrać jeden obiekt, użytkownik musi jedynie kliknąć jego symbol. Aby wybrać listę obiektów, należy myszą zaznaczyć na schemacie prostokątne pole. Wszystkie obiekty graficzne, które znajdują się w obrębie tego prostokąta zostają "**wybrane**". Wybrany obiekt jest otoczony niewielkimi czarnymi kwadratami wokół swojego symbolu graficznego. Dokonanie nowego wyboru, powoduje odznaczenie wszystkich wcześniej zaznaczonych obiektów. Aby usunąć istniejące zaznaczenie, wystarczy kliknąć myszą puste pole poza prostokątem otaczającym wybrane obiekty.

Wstawianie komentarzy

Komentarze mogą być wstawiane w dowolnym miejscu diagramu. Komentarze nie mają wpływu na wykonywanie programu. Pozwalają one uzyskać większą czytelność diagramu. Aby wstawić blok komentarza należy wybrać powyższy przycisk na pasku narzędzi i myszą zaznaczyć prostokątne pole, w które ma zostać wpisany komentarz. Następnie należy wprowadzić tekst komentarza. Przy wprowadzaniu tekstu bloku komentarza nie są wymagane żadne znaki wiodące ani kończące, takie jak "**(***" czy "***)**". Rozmiar bloku komentarza można zmienić przez kliknięcie myszą jego pola i ‘zaczepienie’ myszą narożnika i przesunięcie krawędzi w odpowiednim kierunku.

Przemieszczanie obiektów

Aby przemieścić obiekty na schemacie, należy je wybrać i przeciągnąć myszą w właściwe miejsce. Aby przemieścić połączone obiekty użytkownik musi jedynie

przemieścić symbole graficzne znajdujące się na schemacie. Edytor LD/FBD ISaGRAF automatycznie narysuje ponownie linie łączące między obiektami uwzględniając ich nową lokalizację.

Rysowanie połączeń

Wybranie jednego z powyższych przycisków na pasku narzędzi umożliwia rysowanie połączeń pomiędzy punktami łączenia istniejących elementów. Jeśli narysowane zostanie połączenie od pewnego punktu do pustego miejsca diagramu, to jest ono automatycznie zakończone rogiem zdefiniowanym przez użytkownika tak, aby można było kontynuować rysowanie kolejnego segmentu.

Zmiana rysowania połączeń

Komenda "**Narzędzia / Przenieś linię**" jest używana wtedy, gdy jest wybrane połączenie na schemacie do zmiany automatycznego routingu. Komenda ta nie działa, jeśli połączenie dochodzi do rogu zdefiniowanego przez użytkownika. Kiedy połączenie jest rysowane w formie trzech segmentów, komenda ta zmienia położenie drugiego segmentu. Poniżej przedstawiono przykłady:

Zmiana typu połączenia

Można łatwo zmienić typ połączenia (z negacją logiczną lub bez niej) klikając dwukrotnie myszą jego prawy koniec.

Rysowanie szczebli LD

Aby narysować nowy szczebel LD, należy najpierw wstawić lewą szynę zasilającą. Następnie należy wstawić cewkę: zostanie ona automatycznie połączona z szyną zasilającą. Inne styki i pionowe połączenia LUB mogą być wstawiane bezpośrednio na linii szczebla, bez rysowania żadnych nowych połączeń.
Kiedy w puste miejsce obszaru edycyjnego wstawiany jest nowy styk lub cewka LD, automatycznie rysowana jest nowa pozioma linia szczebla od nowego wstawionego elementu do istniejących szyn zasilających po stronie lewej i prawej. Linia ta nie jest rysowana automatycznie, jeśli nowy styk lub cewka nie są wstawione pomiędzy szynami zasilającymi. Nowo wstawiane styki lub cewki mogą być dowolnie przemieszczane na rysowanym szczeblu. Linie poziome tworzone przez edytor przy wstawianiu symbol styku lub cewki LD mogą zostać wybrane lub usunięte. Na linii poziomej istniejącego szczebla można wstawić nowy symbol styku lub cewki LD. Edytor automatycznie przecina szczebel i łączy go z prawym i lewym punktem połączenia nowo wstawianego styku lub cewki.

Połączenia wielokrotne

Połączenie wielokrotne może zostać stworzone po prawej stronie dowolnego punktu **wyjściowego**. Oznacza to, iż informacje są **transmitowane** do kilku innych punktów diagramu. Na każdy z prawych końców przesyłany jest taki sam stan. Ilość linii rysowanych po prawej stronie punktu połączenia wyjściowego nie jest ograniczona. Dwie linie łączące nie mogą posiadać prawych końców podłączonych do tego samego punktu **wejściowego**, z wyjątkiem następujących symboli LD:

.....prawa szyna zasilająca

.....połączenie wielokrotne operatora lewego (LUB)

Te symbole LD mogą posiadać nieograniczoną ilość wejść.

A.7.3 Opracowywanie istniejącego diagramu

Komendy menu "**Edycja**" są używane do zmiany lub uzupełnienia istniejącego diagramu. Większość z tych komend działa na elementy wybrane aktualnie na schemacie.

Poprawianie diagramu

Klawisz DEL może być użyty do usuwania wybranych elementów. Połączenia zawieszone są usuwane wraz z zaznaczonymi elementami. Komendy "**Edycja / Cofnij**" używa się do przywrócenia elementów po komendzie DEL. Komenda DEL może zostać zastosowana również do wybranie na schemacie grupy elementów. Komendy "**Wytnij**", "**Kopiuj**", "**Wklej**" w menu "**Edycja**" służą do przemieszczania lub kopiowania wybranych elementów.

Znajdź i zamień

Komendy menu "**Edycja / Znajdź**" i "**Edycja / Zamień**" są używane do znajdywania i zamiany tekstów na schemacie. Znaleźć można jedynie pełne nazwy. Wyszukiwania działa w ramach styków, cewek, nazw bloków, zmiennych i etykiet. Nie może ono być wykorzystywane do odnajdywania ciągu znaków w komentarzu. Komenda Zamień nie może być używana do zmieniania nazwy bloku. Wyszukiwanie może odbywać się w górę lub w dół, rozpoczynając od miejsca aktualnego zaznaczenia. Wyszukiwanie jest powtarzane od początku, w „pętli", kiedy osiągnie koniec diagramu.

Wyświetlanie kolejności wykonywania

Jeśli diagram FBD zawiera pętle wstecz, wykonywanie programu nie może być zgodne z zasadą od lewej do prawej / z góry na dół. Aby uniknąć pomyłek, można

wykorzystać komendę "**Narzędzia / Pokaż kolejność uruchamiania**" lub nacisnąć klawisze **Ctrl + F1**, aby wyświetlić kolejność wykonywania, która zostanie użyta podczas kompilacji. Znaczniki z numerami od 1 do N są wyświetlane w pobliżu symboli prowadzących do operacji (cewki, zmienne ustawień i bloki funkcyjne).

Wprowadzanie symboli i tekstów

Aby wprowadzić powiązany z elementem symbol lub tekst, należy dwukrotnie kliknąć myszą ten element. Odnosi się to do zmiennych, styków i cewek, tekstów komentarzy oraz etykiet. Zastosowane do styku lub cewki pozwala również na zmianę ich typu (prosty, zanegowany, itp.).

Okna dialogowe zawierające listy wyboru zmiennych lub bloków są zazwyczaj używane do wprowadzania tekstu. Jeśli włączony jest tryb "**Dane z klawiatury**" w menu "**Opcje**", symbole zmiennych i nazwy bloków są wprowadzane bezpośrednio w pojedynczym tekstowym oknie edycyjnym. Wprowadź nowy tekst i naciśnij "**Enter**", aby go zatwierdzić lub naciśnij klawisz "**Esc**", aby zaniechać modyfikacji i zamknąć tekstowe okno edycyjne. Tekstowe okno edycyjne używane w trybie "wprowadzania ręcznego z klawiatury" nie może zostać zamknięte przy pomocy myszy.

Jeśli włączony jest tryb "**Wejście automatyczne**" w menu "**Opcje**", symbol zmiennej musi zostać wprowadzony bezpośrednio po każdym wstawieniu nowego styku lub cewki. Symbol ten musi zawsze być wprowadzony bezpośrednio po wstawieniu zmiennej lub etykiety.

Wybór typu bloku funkcyjnego

Aby zmienić typ bloku, należy kliknąć go dwukrotnie myszą. Typ bloku wybiera się z listy dostępnych operatorów, funkcji i bloków funkcyjnych. Komenda ta pozwala również zmienić ilość punktów wejściowych w przypadku operatora przemiennego. (np. AND, OR, ADD, MUL...)

Uzyskiwanie wolnej przestrzeni

Po naciśnięciu prawego klawisza myszy w polu rysunku w języku FBD, wyświetlone zostaje menu podręczne. Zawiera ono następujące komendy, które mogą zostać wykorzystane do wstawiania lub usuwania wolnej przestrzeni w pozycji kursora myszy:

Wstaw wiersze: ..... Komenda ta wstawia poziomą wolną przestrzeń, składającą się z 4 wierszy wielkości jednostki siatki każdy, rozpoczynając w miejscu wskazanym kursorem myszy tam, gdzie jest otwarte menu podręczne.
Kasuj wiersze: ..... Komenda ta usuwa niewykorzystaną przestrzeń poziomą (wiersze), rozpoczynając od miejsca wskazanego kursorem

myszy, gdzie jest otwarte menu podręczne. Komenda ta nie może być używana do usuwania elementów FBD.

Kiedy otwarte jest menu podręczne, obszar rysunku FBD, w którym wstawiono wolną przestrzeń lub z którego ją usunięto jest oznaczony szarą linią.

A.7.4 Opcje wyświetlania

Komendy menu "**Opcje**" są używane do nadania odpowiedniego wyglądu rysunkowi diagramu FBD na ekranie.

- **Dopasowywanie widoku**

Komenda "**Opcje / Ustawienia widoku**" otwiera okno dialogowe, w którym zebrane są wszystkie parametry i opcje odnoszące się do przestrzeni roboczej edytora oraz do rysowania diagramu graficznego. Pola wyboru w ramach grupy "Przestrzeń robocza" są używane do wyświetlania lub ukrywania pasków narzędzi i paska statusu edytora. Opcje w ramach grupy "Dokument", pozwalają na pokazanie lub ukrycie punktów siatki edycyjnej.

Opcje grupy "Powiększenie" pozwalają na wybór głównego współczynnika powiększenia. Można również wykorzystać przycisk "powiększenie" na pasku narzędzi edytora i wybrać jeden z kilku domyślnych współczynników powiększenia.

Komendy "**Opcje / Czcionka**" są stosowane do wyboru nazwy czcionki znakowej, która będzie używana we wszystkich dokumentach graficznych ISaGRAF. Przy wybieraniu czcionki, jej styl i rozmiar nie mają znaczenia i nie muszą być określane. Edytory graficzne ISaGRAF zawsze obliczają rozmiar czcionki w zależności od wybranego współczynnika powiększenia.

A.7.5 Style i śledzenie modyfikacji

Edytor LD/FBD ISaGRAF pozwala na wyznaczenie stylu graficznego dowolnego elementu diagramu LD/FBD. Styl jest określany głównie w formie specjalnych kolorów diagramu. Jednak ISaGRAF wykorzystuje również style do umożliwienia śledzenia modyfikacji diagramów i kontroli ich wersji.

Należy zauważyć, iż style nie są widoczne podczas symulacji lub debagowania, ponieważ kolory (czerwony i niebieski) są w tym trybie wykorzystywane do podświetlania stanu PRAWDA / FAŁSZ śledzonych zmiennych.

- ***Style wstępnie zdefiniowane***

Następujące style są wstępnie zdefiniowane:

Normalny	Rysowanie domyślne (w kolorze czarnym). W śledzeniu modyfikacji styl "normalny" oznacza, iż elementy posiadające ten styl stanowią część diagramu pierwotnego. Elementy stylu "normalnego" są zazwyczaj skanowane podczas wykonywania.
Zmodyfikowany	Elementy oznaczone jako "zmodyfikowane" są rysowane na różowo. W śledzeniu modyfikacji styl "zmodyfikowany" wykorzystuje się do podświetlenia elementów, które zostały dodane lub zmienione po oryginalnym wydaniu diagramu. Elementy stylu "zmodyfikowanego" są zazwyczaj skanowane podczas wykonywania.
Usunięty	Elementy oznaczone jako "usunięte" są rysowane na szaro liniami przerywanymi. Elementy takie nie są uwzględniane podczas wykonywania diagramu. Styl ten jest wykorzystywany do śledzenia elementów usuniętych po oryginalnym wydaniu, jeśli konieczne jest kontrolowanie wersji.
Własny	Poza stylem wstępnie zdefiniowanymi edytor LD/FBD ISaGRAF pozwala na wybór dowolnego koloru do zastosowania do części diagramu. Ta część diagramu jest wtedy wykonana stylem "niestandardowym". Zastosowanie stylu "niestandardowego" nie ma wpływu na wykonywanie diagramu.

Komend podmenu "**Styl**" w menu "**Edycja**" używa się do ręcznego zastosowania stylu do wybranych elementów.

Śledzenie modyfikacji

Wykorzystanie stylów oraz dostępność stylu "usunięty" pozwala na automatyczne śledzenie modyfikacji w istniejącym schemacie. Komendy "**Zaznacz modyfikacje**" w menu "**Edycja/Styl**" używa się do włączenia lub wyłączenia śledzenia modyfikacji. Kiedy włączona jest opcja "Oznacz modyfikacje", wszystkim elementom zmienionym lub dodanym do diagramu jest automatycznie nadawany styl "Zmodyfikowany". Kiedy element zostaje usunięty przy pomocy komend "Kasuj" lub "Wytnij", nie jest on w sposób widoczny usuwany z diagramu, lecz po prostu nadawany jest mu styl "Usunięty". Pozwala to użytkownikowi automatycznie śledzić wszystkie modyfikacje wprowadzone do diagramu.
Komendy "**Edycja/Styl/Skasuj wszystkie usunięte elementy**" używa się do rzeczywistego usunięcie elementów oznaczonych stylem "Usunięty" z diagramu LD/FBD. Komenda ta nie uwzględnia bieżącego zaznaczenia i zawsze odnosi się do całości diagramu.
Aby "odzyskać" jeden element zaznaczony stylem "**Usunięty**", należy wybrać dany element i zastosować do niego styl "**Normalny**" lub "**Zmodyfikowany**" albo dowolny styl " **Własny**". Operacja taka może prowadzić do nieprawidłowych połączeń (więcej niż jedno połączenie prowadzące do tego samego punktu wejściowego), które zostaną wykryte podczas następnej weryfikacji programu.

A.8 Korzystanie z edytora tekstowego

Niniejszy rozdział opisuje jedynie funkcje i komendy edytora tekstowego ISaGRAF szczególnie przy jego wykorzystaniu do wprowadzania kodów źródłowych programów w językach ST i IL.

A.8.1 Komendy edycyjne

Komendy menu "**Edycja**" są wykorzystywane do edytowania tekstu. Większość z tych komend działa na znakach wybranych aktualnie w schemacie lub wykonuje operację w miejscu aktualnego położenia znaku „^".

Wytnij i wklej

Klawisz DEL może być wykorzystany do usunięcia zaznaczonego tekstu. Komendy "**Edycja / Cofnij**" używa się do przywrócenia elementów z kopii zapasowych po użyciu komendy DEL. Komendy "**Wytnij**", "**Kopiuj**", "**Wklej**" w menu "**Edycja**" stosuje się do przemieszczania lub kopiowania tekstu w programie lub do wstawiania fragmentów tekstu skopiowanych do schowka przez inne aplikacje.

Znajdź i zamień

Komendy menu "**Edycja / Znajdź /Zmień**" są stosowane do wyszukiwania i zamiany tekstu w programie. Odnaleźć można dowolny ciąg znaków. Szukanie może odbywać się w górę lub w dół, rozpoczynając od aktualnej pozycji znaku „^". Kiedy wyszukiwanie dochodzi do końca programu, nie jest powtarzane od początku.

Idź do linii

Komenda "**Edycja / Skocz do linii**" jest używana do przemieszczania znaku „^" do linii o konkretnym numerze. Może to być bardzo przydatne przy uzyskiwaniu dostępu do linii programu ST lub IL, w której kompilator ISaGRAF wykrył błąd określony numerem linii.

Wstaw symbol ze słownika

Komendy "**Edycja / Wstaw zmienną**" używa się do wstawienia zmiennej lub obiektu zadeklarowanego w słowniku projektu w pozycji znaku „^". Symbol wybierany jest przy pomocy wspólnego okna wyboru zmiennej, opisanego w rozdziale "Więcej informacji na temat edytorów programów" w tym dokumencie.

Wstaw plik

Komenda "**Edycja / Wstaw plik**" wstawia całą zawartość pliku w miejscu aktualnego położenia znaku „^". Należy zauważyć, iż komenda ta obsługuje jedynie pliki tekstowe w formacie ASCII.

A.8.2 Opcje

Komendy menu "**Opcje**" są używane do wyświetlania i ukrywania pasków narzędzi edytora oraz do wybierania czcionki znakowej. Wybrana czcionka będzie używana do każdej edycji tekstów w pakiecie ISaGRAF.

W przypadku wprowadzania kodu źródłowego programu ST / IL komenda "**Opcje / Pokaż słowa kluczowe**" jest używana do pokazywania i ukrywania okna narzędziowego, grupującego najczęściej używane słowa kluczowe języka ST lub IL. Aby wstawić odpowiednie słowo kluczowe lub operator w aktualnej pozycji znaku „^", należy kliknąć przycisk na pasku narzędzi.

A.9 Więcej informacji o edytorach programowych

Niniejszy rozdział zawiera przydatne informacje dotyczące funkcji edycyjnych, wspólnych dla wszystkich edytorów programów ISaGRAF. Zajmuje się on głównie połączeniami z innymi narzędziami i oknami dialogowymi pakietu ISaGRAF.

A.9.1 Wywoływanie innych narzędzi ISaGRAF

Weryfikacja (kompilacja) programu

Komenda "**Plik / Weryfikuj**" uruchamia generator kodu ISaGRAF weryfikujący składnię programową edytowanego aktualnie programu. W przypadku języka SFC, sprawdzany jest zarówno poziom 1 jak i 2. Po zakończeniu weryfikacji składni, okno generatora kodu musi zostać zamknięte aby umożliwić pracę z programem. Jeśli w aplikacji znajduje się tylko jeden program (edytowany), w razie niewykrycia błędu składni generowany jest kod aplikacji. Komenda "**Opcje / Opcje kompilatora**" jest wykorzystywana do ustawiania parametrów kompilacji i optymalizacji. Dalsze informacje na temat kompilacji i generowania kodu znajdują się w rozdziale "Korzystanie z generatora kodu" w tym dokumencie.

Symulacja lub debagowanie aplikacji

Komendy "**Plik / Symuluj**" i "**Plik / Debaguj**" uruchamiają debager graficzny ISaGRAF w trybie symulacji lub rzeczywistego połączenia oraz ponownie otwierają edytowany program SFC w trybie debagowania. Kiedy program używany jest w trybie debagowania, nie można wprowadzać w nim żadnych modyfikacji.

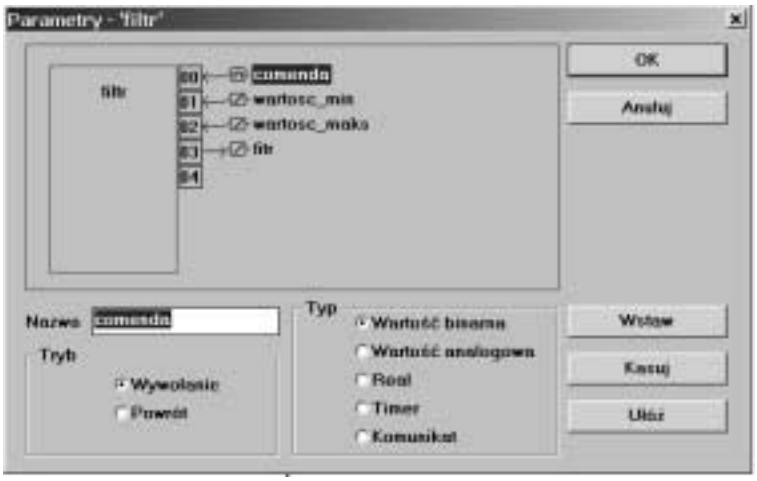
Edycja słownika zmiennych

Komenda "**Plik / Słownik**" jest używana do edycji słownika zmiennych bieżącej aplikacji i bieżącego programu. Zawiera ona również pola wprowadzania danych, służące do edycji słów zdefiniowanych przez użytkownika. **Lokalne** deklaracje lub słowa zdefiniowane odnoszą się do edytowanego aktualnie programu.

A.9.2 Parametry programu

Kiedy edytowany program jest funkcją, blokiem funkcyjnym lub podprogramem, komendy "**Plik / Parametry**" używa się do zdefiniowania jego parametrów wywołania i parametrów zwracanych (powrotu). Komenda ta nie odnosi skutku, jeżeli edytowany program jest programem SFC lub programem najwyższego poziomu z sekcji **Początek** lub **Koniec**.

Podprogramy, funkcje i bloki funkcyjne mogą mieć do **32** parametrów (wejściowych lub wyjściowych). Funkcja lub podprogram posiada zawsze jeden (i tylko jeden) parametr zwracany, który musi posiadać taką samą nazwę jak funkcja dla zachowania zgodności z konwencjami pisania w języku ST. Poniższy rysunek pokazuje parametry podprogramu :



Lista w lewej górnej części okna pokazuje parametry w kolejności ich wywoływania: najpierw parametry wywołania, na końcu parametry zwracane (parametry powrotu). Dolna część okna pokazuje szczegółowy opis parametru aktualnie wybranego z listy. Jako parametr mogą służyć wszystkie typy danych ISaGRAF. Parametry zwracane muszą być umieszczone na liście po parametrach wywołania. Nadawanie parametrom nazw musi być zgodne z następującymi zasadami:

- długość nazwy nie może przekraczać 16 znaków
- pierwszy znak musi być literą
- kolejne znaki muszą być literami, cyframi lub znakami podkreślenia
- nazwy nie uwzględniają wielkości liter

Komendy "**Wstaw**" używa się do wstawienia nowego parametru przed parametrem wybranym. Komendy "**Kasuj**" używa się do usunięcia wybranego parametru. Komenda "**Ułóż**" automatyczne ustawia ponownie (sortuje) parametry, umieszczając parametry zwracane na końcu listy.

A.9.3 Inne komendy menu "Plik"

Następujące komendy dostępne są w menu "**Plik**" wszystkich edytorów programowych:

Otwórz następny program

Pole "**Plik / Otwórz**" pozwala użytkownikowi zamknąć aktualnie edytowany program i rozpocząć edycję innego programu aktualnego projektu w tym samym języku. Funkcja ta nie może zostać wykorzystana do edycji programu napisanego w innym

języku. Nowo wybrany program zajmuje miejsce aktualnego programu w oknie edycyjnym.

Wydruk programu

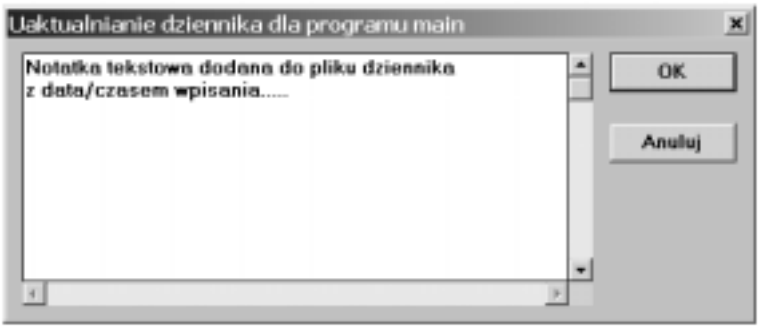
Komenda "**Plik / Drukuj**" wysyła edytowany program do drukarki. Komenda ta automatycznie uruchamia generator dokumentów ISaGRAF w celu wydrukowania edytowanego programu i skojarzonych z nim zmiennych lokalnych.

W przypadku niektórych programów graficznych (SFC, FBD i LD) można użyć również komendy "**Edycja / Kopiuj rysunek**" aby skopiować do schowka rysunek grafu w formacie metapliku tak, aby można było wkleić go w innych aplikacjach, takich jak edytory tekstów. W programach SFC w skopiowanym metapliku pojawiają się tylko informacje poziomu 1 (graf, numerowanie i komentarze poziomu 1).

A.9.4 Aktualizacja dziennika programu

Plik dziennika dołączony do edytowanego programu może być poprawiany ręcznie przy użyciu komendy "**Plik / Dziennik**". Plik dziennika jest aktualizowany automatycznie z wyprowadzeniem komunikatów z kontroli składni przy każdej kompilacji programu. Rezultaty kompilacji są uzupełniane stemplem daty / czasu kompilacji.

Jeśli w menu „**Opcje**" edytorów programu wybrany jest tryb „**Aktualizuj dziennik**", to każdemu zapisaniu programu na dysku towarzyszy otwarcie okna dialogowego "Dziennik".



Jeśli naciśnięty zostanie przycisk OK, wprowadzone uwagi tekstowe są zapisywane na końcu dziennika wraz z aktualnym stemplem daty / czasu. Funkcja ta jest bardzo przydatna do konserwacji kompletnych programów, ponieważ dostarcza użytecznych informacji dotyczących cyklu życia programu.

A.9.5 Wybieranie zmiennej ze słownika

Przy edycji programu tekstowego (ST lub IL) komenda "**Edycja / Wstaw zmienną**" pozwala na wybór nazwy zadeklarowanej zmiennej w celu wstawienia jej w aktualnej pozycji znaku „^". Podczas edycji programów LD lub FBD, wybór zmiennej jest wymagany w przypadku opisu cewek, styków, parametrów We/Wy bloku lub okien ze zmiennymi FBD. W obu przypadkach otwierane jest okno dialogowe "Wybierz" pozwalające wybrać zadeklarowaną zmienną.

Okno wyboru "**Zakres**" służy do wyboru zmiennej globalnej lub lokalnej. Okno wyboru po prawej stronie pozwala na wybór typu danych. Niewielkie ikony obok okna wyboru typu mogą być używane jako skróty ułatwiające wybór najczęściej używanych typów danych:

Binarne

Integer / Real

Timer

Komunikat

Aby wybrać zmienną, należy kliknąć jej nazwę na liście. Nazwa i komentarz zostają wyświetlone w górnej części listy. Naciśnięcie przycisku "**OK**" potwierdza dokonany wybór. Możliwe jest również bezpośrednie wpisanie nazwy zmiennej w sterowaniu edycją bez używania listy.

A.9.6 Okno informacji wyjściowych

Poniższe komendy dostępne są w menu Narzędzia edytorów wszystkich języków. Stosowane są one do wyświetlania informacji w formie wąskiej listy tekstowej w dolnej części okna edycyjnego programu.

"Pokaż okno wynikowe kompilatora" wyświetla w oknie informacji wyjściowych komunikaty o błędach po ostatniej kompilacji edytowanego programu.
"Znajdź w…" znajduje miejsca tekstowe w całym edytowanym programie i wyświetla ich listę w oknie informacji wyjściowych. W przypadku języków SFC i FC komenda ta przeszukuje wszystkie programy poziomu 2.
"Ukryj okno wynikowe" zamyka okno z listą informacji wyjściowych.

Kiedy w oknie informacji wyjściowych pojawią się komunikaty o błędach lub teksty, należy dwukrotnie kliknąć dany wiersz, aby zobaczyć czego dotyczy ta informacja. W przypadku języków SFC i FC komenda ta otwiera odpowiednie okno programowania na poziomie 2.

A.10 Korzystanie z edytora słownika

Słownik ISaGRAF to narzędzie edycyjne, służące do deklarowania zmiennych wewnętrznych, zmiennych We/Wy, instancji bloków funkcyjnych oraz " zdefiniowanych słów " aplikacji. Słownik łączy zadeklarowane zmienne i instancje bloków funkcyjnych aplikacji oraz słowa zdefiniowane jako ciągi stałych.

Zmienne, bloki funkcyjne i słowa zdefiniowane muszą zostać zadeklarowane w słowniku przed ich zastosowaniem w kodzie źródłowym. Zmienne i słowa zdefiniowane mogą być stosowane w dowolnym z języków automatyki: SFC, FBD, LD, ST oraz IL. Bloki funkcyjne zastosowane w języku FBD nie muszą być deklarowane, ponieważ edytory FBD i LD ISaGRAF automatyczne deklarują instancje zastosowanych bloków.

Zmienne

Zmienne są sortowane według zakresu i typu. Jedynie zmienne tego samego typu i o tym samym zakresie mogą być wprowadzane na tej samej siatce wejściowej. Oto podstawowe zakresy zmiennych:

GLOBALNA .....może być stosowana w dowolnym programie bieżącego projektu
LOKALNA.....może być stosowana tylko w jednym programie

Oto podstawowe typy zmiennych:

BINARNA.....wartości logiczne prawda/fałsz
ANALOGOWA .....wartości rzeczywiste lub całkowite
TIMER.....wartości czasowe
KOMUNIKAT .....ciągi znaków

Zmienną identyfikuje się za pomocą nazwy, komentarza, atrybutu, adresu sieciowego i innych pól szczegółowych. Oto podstawowe atrybuty zmiennych:

WEWNĘTRZNA.....zmienna w pamięci
WEJŚCIOWA.....zmienna skojarzona z urządzeniem wejściowym
WYJŚCIOWA.....zmienna skojarzona z urządzeniem wyjściowym
STAŁA zmienna wewnętrzna tylko do odczytu (z wartością początkową)

Uwaga: Zmienne typu **Timer** są zawsze zmiennymi **wewnętrznymi**. Zmienne **wejściowe** i **wyjściowe** posiadają zawsze zasięg **GLOBALNY**.

Słowa zdefiniowane

Słowo zdefiniowane to alias, którego można używać w dowolnym języku w zastępstwie ciągu tekstowego. Zastępowany tekst może być nazwą zmiennej, wyrażeniem stałym lub wyrażeniem złożonym. Słowa zdefiniowane są posortowane według zakresu. Jedynie słowa zdefiniowane tego samego typu i o tym samym zakresie mogą być wprowadzane na jednej siatce wejściowej. Oto podstawowe zakresy:

OGÓLNA.....może być wykorzystana przez dowolny program w dowolnym projekcie

GLOBALNA.....może być wykorzystana przez dowolny program w bieżącym projekcie

LOKALNA.....może być wykorzystana tylko przez jeden program

Słowo zdefiniowane jest identyfikowane za pomocą nazwy, dobrze zdefiniowanego bloku równoważności tekstowych ST oraz dowolnego komentarza.

Instancje bloków funkcyjnych

Instancje bloków funkcyjnych stosowane w językach ST i IL muszą zostać zadeklarowane w słowniku. Każda kopia bloku funkcyjnego musi zostać zidentyfikowana, ponieważ zawiera wewnętrzne dane "ukryte". Poniższy przykład przedstawia zdefiniowany w bibliotece blok funkcyjny "R_TRIG" (wykrywanie zbocza narastającego), stosowany do wykrywania zbocza sygnału dla różnych zmiennych. Każda kopia bloku musi być zidentyfikowana przy pomocy indywidualnej nazwy. Nadawanie nazwy typowi bloku i definiowanie jego parametrów odbywa się przy pomocy Menedżera Biblioteki:

Nazwa bloku:	**R_TRIG**
***Parametry*:**	**Input=CLK**
	Output=Q

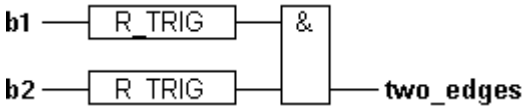
Nadawanie nazw instancjom odbywa się przy pomocy Menedżera Biblioteki:

***Nazwa instancji*:**	**TRIG_B1**	***Nazwa bloku*:**	**R_TRIG**
***Nazwa instancji*:**	**TRIG_B2**	***Nazwa bloku*:**	**R_TRIG**

zadeklarowane instancje mogą zostać wykorzystane w programach ST:

```
TRIG_B1 (b1);
edge_b1 := TRIG_B1.Q;     (* wykrywanie zbocza zmiennej b1 *)
TRIG_B2 (b2);
edge_b2 := TRIG_B2.Q;     (* wykrywanie zbocza zmiennej b2 *)
```

Zadeklarowane instancje bloków funkcyjnych mogą być **GLOBALNE** (znane wszystkim programom projektu) lub **LOKALNE** dla jednego programu. Bloki funkcyjne stosowane w językach FBD lub LD nie muszą być deklarowane, ponieważ edytor FBD ISaGRAF automatycznie deklaruje instancje zastosowanych bloków.



(* nazwa bloku funkcyjnego jest zawsze zdefiniowana w bibliotece. Edytory FBD i LD ISaGRAF automatycznie deklarują instancję za każdym razem, gdy blok zostaje wstawiony do schematu *)

Instancje bloków funkcyjnych zadeklarowane automatycznie przez edytory FBD i LD są zawsze **LOKALNE** dla edytowanego programu.

Adresy sieciowe

Adresy sieciowe są **opcjonalne**. Zmienna z niezerowym adresem sieciowym może być **śledzona** przez system zewnętrzny (na przykład system wizualizacji procesu) podczas wykonywania programu. Uogólniając, adres sieciowy stanowi mechanizm identyfikacji dla każdego wykonywanego systemu komunikacji, który nie obsługuje nazw literowych. Adres sieciowy może być skojarzony z każdą zmienną, podczas pełnego opisywania, kiedy zmienna jest tworzona lub modyfikowana.

A.10.1 Okno główne słownika

Okno edycyjne słownika pokazuje listę zmiennych tego samego typu i zakresu. Typ i zakres edytowanych zmiennych jest zawsze wyświetlony w pasku tytułowym.

Okno edycyjne przedstawia jedynie główne pola opisu zmiennej: nazwę, atrybut oraz adres sieciowy i komentarz tekstowy. Pełny opis wybranej zmiennej wyświetlony jest zawsze na pasku stanu.

Poniższe przyciski paska narzędzi pozwalają wybrać zakres zmiennych do edycji:

OGÓLNA.....może być wykorzystana przez dowolny program w dowolnym projekcie

GLOBALNA .....może być wykorzystana przez dowolny program w bieżącym projekcie

LOKALNA .....może być wykorzystana tylko przez jeden program

Typ obiektu do edycji wybiera się za pomocą wyświetlanej „zakładki” z paskiem tytułowym:

Pole wprowadzania tekstu po lewej stronie paska narzędzi służy do szukania nazwy prefiksu zmiennej. W takim wypadku szukanie odbywa się w całej liście, od początku, na podstawie bieżącego zaznaczenia. Dostępna jest również komenda "**Edycja / Znajdź**", pozwalająca znajdywać ciąg tekstowy w nazwach i komentarzach do zmiennych oraz przemieszczać wybór na tę zmienną. Wyszukiwanie nigdy **nie uwzględnia wielkości liter**.

A.10.2 Zarządzanie zmiennymi

Dostępne komendy menu "**Pliki**" działają na całej wybranej klasie zmiennych, instancji bloków funkcyjnych lub słów zdefiniowanych. Komendę "**Inne**" stosuje się do wyboru typu i zakresu obiektów do edycji.

Drukowanie zmiennych

Komenda "**Pliki / Drukuj**" jest używana do drukowania aktualnie edytowanej listy zmiennych lub zdefiniowanych słów na standardowym urządzeniu drukującym systemu Windows™. Drukowanie odbywa się przy użyciu generatora dokumentów ISaGRAF. Wydruk zawiera pełny opis każdej aktualnie edytowanej zmiennej lub zdefiniowanego słowa.

Tworzenie nowych zmiennych

Komenda "**Edycja / Nowy**" pozwala użytkownikowi tworzyć nowe zmienne, instancje bloków funkcyjnych lub zdefiniowane słowa wybranego zakresu i typu. Nowe zmienne są wstawiane tuż przed zmienną aktualnie wskazywaną przez pasek wyboru. Po wydaniu tej komendy otwierane jest okno wprowadzania danych, pozwalające wpisać opis zmiennej. Po ukończeniu opisu, naciśnięcie przycisku "**Zapisz**" wstawia zmienną na listę. Okno wprowadzania danych jest automatycznie otwierane ponownie tak, że użytkownik może wprowadzać kolejne zmienne przy pomocy tej samej komendy "**Edytuj**". Naciśnięcie przycisku "**Anuluj**" okna dialogowego przerywa proces tworzenia zmiennych.

Modyfikowanie istniejących zmiennych

Komenda "**Edytuj**" menu "**Edycja**" pozwala użytkownikowi na modyfikację opisu zmiennej wskazywanej aktualnie na pasku wyboru. Po wybraniu tej komendy otwierane jest okno wprowadzania danych pozwalające na modyfikację opisu zmiennej. Po ukończeniu opisywania naciśnięcie przycisku "**Zapisz**" wprowadza modyfikację. Użytkownik może również nacisnąć przyciski "**Następny**" i "**Poprzedni**" aby poszerzyć komendę modyfikacji o sąsiednie zmienne. Naciśnięcie przycisku "**Anuluj**" zamyka okno dialogowe bez zachowywania jakiejkolwiek modyfikacji.

Wycinanie i wklejanie

Edytor słownika ISaGRAF umożliwia **zaznaczanie wielu linii**. Dostępnych jest wiele komend pozwalających na pracę nad edytowaną aktualnie listą zmiennych. Poniżej znajdują się dostępne komendy menu "**Edycja**":

KOPIUJ.....Kopiuje wybraną grupę zmiennych do schowka słownika
WYTNIJ .....Kopiuje wybraną grupę zmiennych i usuwa ją z edytowanej listy
WYCZYŚĆ .....Usuwa zaznaczoną grupę zmiennych z edytowanej listy
WKLEJ .....Wstawia zawartość schowka przed wybraną zmienną

Funkcje Kopiuj/Wytnij/Wklej mogą być używane do edycji kilku list ze zmiennymi. Nie mogą one być używane w stosunku do list o różnych typach obiektów.

Sortowanie zmiennych

Komenda "**Narzędzia / Sortuj**" sortuje zmienne lub zdefiniowane słowa aktualnie edytowanej listy. Sortowanie odbywa się według atrybutów zmiennych:

- najpierw zmienne wewnętrzne
- potem zmienne wejściowe
- w końcu zmienne wyjściowe

Zmienne o takim samym atrybucie są sortowane w kolejności alfabetycznej. Zdefiniowane słowa są zawsze sortowane w kolejności alfabetycznej.

Ustawianie adresów sieciowych

Adresy sieciowe są **opcjonalne**. Zmienna z niezerowym adresem sieciowym może być **śledzona** przez system zewnętrzny (na przykład system wizualizacji procesu) podczas wykonywania programu. Adres sieciowy może być skojarzony z każdą zmienną, podczas pełnego opisywania, kiedy zmienna jest tworzona lub modyfikowana. Komenda "**Narzędzia / Przenumeruj adresy**" pozwalam użytkownikowi na ustawienie adresów sieciowych dla całej grupy zmiennych. Po wybraniu tej komendy jest ona stosowana do grupy zmiennych zaznaczonych aktualnie na liście. Wpisanie szesnastkowego **adresu bazowego** (adres pierwszej zmiennej w grupie) prowadzi do przypisania zmiennym z tej grupy **kolejnych adresów** sieciowych. Wpisanie zerowego adresu bazowego powoduje wyzerowanie adresów sieciowych wszystkich wybranych zmiennych.

Importowanie łańcuchów binarnych „prawda/fałsz”

W czasie edycji zdefiniowanych słów, komenda "**Narzędzia / Importuj definicje prawda/fałsz**" pozwala użytkownikowi na automatyczne zdefiniowanie jako słów kluczowych języka ciągów skojarzonych ze zmiennymi binarnymi, reprezentującymi

stany TRUE i FALSE. Ciągi takie są zazwyczaj definiowane dla formatowania do debagowania. Jeśli mają one być użyte w programach, muszą być określone jako słowa zdefiniowane. Komenda ta wyszukuje ciągi binarne prawda/fałsz w deklaracjach takim zakresie, jaki jest aktualnie wybrany do edycji zdefiniowanych słów.

A.10.3 Opis obiektów

Dla każdej zmiennej, instancji bloku funkcyjnym czy zdefiniowanego słowa musi zostać wprowadzony pełny opis. Pola opisu są inne dla każdego typu obiektu. Poniższe pola są wspólne dla każdego typu zmiennej:

Nazwa.....Nazwa zmiennej: pierwszy znak musi być literą, następne muszą być literami cyframi lub znakami '_'.
Adres sieciowy .....Szesnastkowy adres sieciowy (opcjonalny). Kiedy to pole ma wartość niezerową, zmienna może być śledzona przez systemy zewnętrzne w wykonywania programu.
Komentarz.....Dowolny komentarz opisujący zmienną.
Zachowaj.....Opcja ta wskazuje, że zmienną należy zapisać w pamięci nieulotnej.

Oto inne pola opisu dla zmiennej **binarnej**:
Atrybut.....Określa zmienną wewnętrzną, stałą, wejściową lub wyjściową.
Ciąg "Fałsz".....Ciąg oznaczający fałsz w czasie wykonywani debagowania.
Ciąg "Prawda" .....Ciąg oznaczający prawdę w czasie wykonywania debagowania.
Ustaw na pocz. prawda.Jeśli opcja ta jest zaznaczona, wartością początkową jest PRAWDA, jeśli nie, to wartością początkową jest FAŁSZ.

Oto inne pola opisu dla zmiennej typu **integer** lub **real**:
Atrybut.....Określa zmienną wewnętrzną, stałą, wejściową lub wyjściową.
Format.....Określa zmienną typu integer lub real (zmiennoprzecinkową). Można wybrać format wyświetlania używany w czasie debagowania.
Jednostka .....Łańcuch używany do definiowania nazwy jednostki fizycznej i wyświetlany w czasie debagowania.
Konwersja.....Nazwa tabeli konwersji lub funkcji konwersji skojarzonej ze zmienną (tylko dla zmiennych wejściowych i wyjściowych)
Wartość pocz......Początkowa wartość zmiennej (musi mieć taki sam format jak zmienna). Jeśli nie jest podana, to wartością początkową jest 0.

Oto inne pola opisu dla zmiennej typu **timer**:

Atrybuty.....Określa zmienną wewnętrzną lub stałą.
Wartość pocz......Początkowa wartość zmiennej (wartość czasowa). Jeśli nie jest określona, wartością początkową jest time#0s.

Oto inne pola opisu dla zmiennej typu **komunikat**:
Atrybut.....Określa zmienną wewnętrzną, stałą, wejściową lub wyjściową.
Długość maksymalna....Określa maksymalną ilość znaków, które można zapamiętać w komunikacie.
Początkowo .....Początkowa wartość zmiennej (długość nie może przekraczać pojemności komunikatu). Jeśli nie jest określona, wartością początkową jest ciąg pusty.

Oto inne pola opisu dla **zdefiniowanego słowa**:
Nazwa.....Nazwa użyta w plikach źródłowych ST: pierwszy znak musi być literą, kolejne muszą być literami, cyframi lub znakami '_'.
Definicja .....Ciąg zgodny ze składnią ST, który podczas kompilacji zajmuje miejsce zdefiniowanego słowa. Przykład: Nazwa = PI - Równoważność = 3.14159
Komentarz.....Dowolny komentarz do zdefiniowanej równoważności

Oto inne pola opisu dla **instancji bloku funkcyjnego**:
Nazwa.....Nazwa instancji użyta w plikach źródłowych ST: pierwszy znak musi być literą, kolejne muszą być literami, cyframi lub znakami '_'.
Typ.....Nazwa odpowiadającego boku funkcji w bibliotece.
Komentarz.....Dowolny komentarz do instancji bloku funkcyjnego.

A.10.4 Szybkie deklarowanie

Komenda "**Narzędzia / Szybka deklaracja**" pozwala zadeklarować kilka zmiennych jednocześnie. Zmienne utworzone za pomocą szybkiego deklarowania posiadają nazwy zgodne z konwencją numerowania. Potrzebne jest tu określenie:

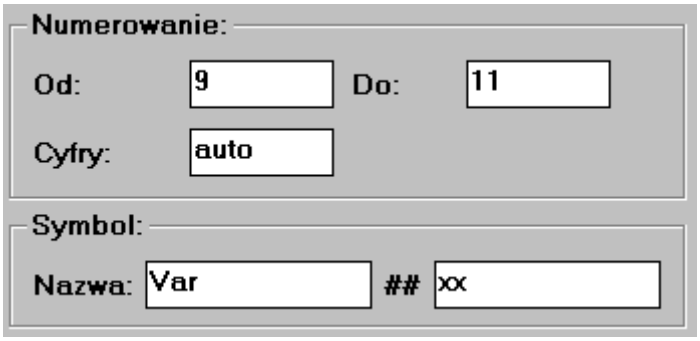
- indeksu (numeru) pierwszej i ostatniej zmiennej,
- tekstu, który należy dodać przed i po numerze w symbolach zmiennych
- ilość cyfr użytych do wyrażenia numeru w symbolach zmiennych.

Ponadto można określić atrybuty tworzonych zmiennych (wewnętrzne, wejściowe lub wyjściowe, itp.) oraz kilka własności zależnie od typu zmiennej (atrybut "Zachowuj", format integer lub real, maksymalna długość komunikatu).

Zawsze konieczne jest zdefiniowanie tekstu, który ma zostać wstawiony przed numerem zmiennej, ponieważ symbol zmiennej nie może rozpoczynać się od cyfry. Kiedy "ilość cyfr" jest ustawiona na "Automatyczna", ISaGRAF formatuje numer zmiennej przy użyciu minimalnie wymaganej ilości cyfr. Kiedy ilość cyfr zostanie określona, ISaGRAF formatuje wszystkie numery zgodnie z określoną długością

dodając na początku znaki '0'. Ustalenie stałej liczby cyfr w numerach zmiennych może być bardzo przydatne do zapobiegania niewłaściwemu sortowaniu leksykograficznemu. Poniżej podano kilka przykładów.

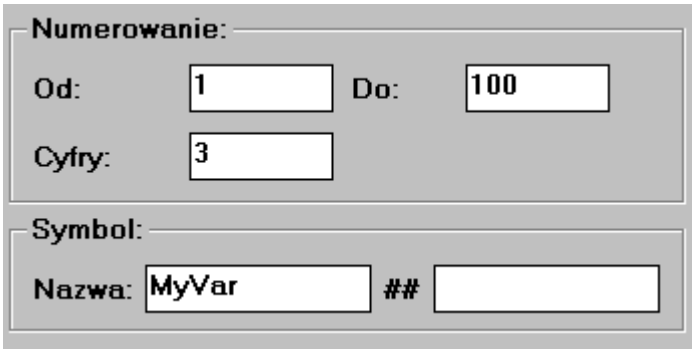
Przykład 1: Takie ustawienie szybkiego deklarowania:



utworzy następujące trzy zmienne:

Var9xx Var10xx Var11xx

Przykład 2: Takie ustawienie szybkiego deklarowania:



utworzy 100 zmiennych o nazwach od MyVar001 do MyVar100

A.10.5 Mapa adresów Modbus SCADA

"Adresy sieciowe" ISaGRAF są często wykorzystywane do utworzenia połączenia pomiędzy systemem ISaGRAF a SCADA, opartym na komunikacji Modbus. W takim przypadku SCADA jest urządzeniem nadrzędnym w stosunku do Modbus, a oprogramowanie wbudowane ISaGRAF działa jako urządzenia podrzędne Modbus. Adresy sieciowe są wykorzystywane do stworzenia wirtualnej mapy Modbus dla wszystkich zmiennych ISaGRAF, które muszą być sterowane za pośrednictwem SCADA. Komenda "**Narzędzia / Mapa adresów Modbus SCADA**" jest narzędziem do szybkiego tworzenia wirtualnej mapy adresów Modbus zmiennych w występujących w aplikacji.

Narzędzie mapujące zawiera dwie listy. Lista górna jest segmentem mapy Modbus (4096 lokalizacji) przedstawiającym mapowane zmienne (te, które posiadają adres sieciowy). Dolna lista przedstawia niemapowane zmienne (bez zdefiniowanego adresu sieciowego). Adres "0" nie może być użyty do mapowania zmiennej.

Stosując komendy "**Mapuj zaznaczoną zmienną**" i "**Usuń zmienną z mapy**" z menu "**Edycja**" można przemieszczać zmienne z jednej listy do drugiej i w ten sposób budować mapę. Te same operacje można przeprowadzić klikając dwukrotnie symbol zmiennej na liście w celu przesłania go do innej listy. W każdej chwili można użyć wysuwanej w dół listy "Segment", aby zobaczyć inny segment mapy.

Komendy menu "**Opcje**" mogą być użyte w dowolnym momencie do wyświetlenia adresów w liczbach dziesiętnych lub szesnastkowych.

Komendy "**Edycja / Znajdź**" są używane do deklarowanych zmiennych, niezależnie czy były mapowane czy nie.

A.10.6 Wymiana informacji z innymi aplikacjami

Narzędzie edycyjne ISaGRAF oferuje funkcje importu/eksportu w celu wymiany informacji z innymi aplikacjami, takimi jak edytory tekstu, arkusze kalkulacyjne, programy zarządzania bazami danych, itp. Komendy te są zgrupowane w menu "**Narzędzia**". Komenda "**Eksportuj tekst**" tworzy opis pól w formie prostego tekstu ASCII opisującego grupę edytowanych obiektów i zapisuje ten tekst w schowku systemu Windows lub w pliku. Informacje takie są zazwyczaj wykorzystywane przez inne aplikacje. Komenda "**Importuj tekst**" importuje pola opisu deklaracji zmiennych w formie prostego tekstu ASCII ze schowka systemu Windows lub z pliku oraz aktualizuje edytowaną aktualnie listę o importowane pola. Informacje takie są zazwyczaj tworzone przez inną aplikację.

Eksport danych

Okno dialogowe "**Eksport zmiennych**" pojawia się po wybraniu komendy "**Eksportuj tekst**". Umożliwia ono użytkownikowi sterowanie mechanizmem eksportu.

Zaznaczenie opcji "**Kompletna lista**" oznacza, iż eksportowana ma być cała lista. W takim wypadku ignorowane są aktualnie zaznaczone elementy. Zaznaczenie opcji "**Zaznaczone zmienne**" oznacza, iż eksportowane mają być jedynie podświetlone zmienne.
Jeśli zaznaczona jest opcja "**Schowek**", eksportowane informacje są zapisywane w formie prostego tekstu ASCII w schowku systemu Windows. Tekst jest wówczas dostępny poprzez komendy "wklej" w innych aplikacjach. Jeśli zaznaczona jest opcja "**Plik**", eksportowany tekst zapisywany jest w pliku ASCII. Należy określić kompletną ścieżkę dostępu dla tego pliku. Można użyć komendy "**Przeglądaj**", aby znaleźć właściwą ścieżkę.
Następnie użytkownik wybiera format eksportowanego tekstu. Dostępne formaty opisano w dalszych sekcjach. Naciśnięcie przycisku "**OK**" uruchamia funkcję eksportu. Naciśnięcie przycisku "**Anuluj**" zamyka okno dialogowe i powoduje rezygnację z wykonywania komendy eksportu.
Wszystkie pola wybranych obiektów są zachowywane w eksportowanym tekście w standardowej kolejności deklarowania. Pierwsza linia eksportowanego tekstu zawiera nazwę pól. Każdy obiekt jest opisany w jednej linii tekstu. Separator "końca linii" to standardowa sekwencja systemu MS-DOS "**0d-0a**". Nazwy używane do identyfikacji pól w pierwszej eksportowanej linii można zmienić naciskając przycisk "**Słowa kluczowe**". Komendę tę opisano w dalszych sekcjach.

Import danych

Okno dialogowe "**Import zmiennych**" pojawia się po wybraniu komendy "**Importuj tekst**". Umożliwia ono użytkownikowi sterowanie mechanizmem importu.

Jeśli zaznaczona jest opcja "**Schowek**", importowane informacje są pobierane w formie prostego tekstu ASCII ze schowka systemu Windows. Jeśli zaznaczona jest opcja "**Plik**", importowany tekst odczytywany jest z pliku ASCII. Należy określić kompletną ścieżkę dostępu dla tego pliku. Można użyć komendy "**Przeglądaj**", aby znaleźć właściwą ścieżkę.

Funkcja importu automatycznie rozpoznaje format (separatory) użyte w importowanym tekście. Dostępne formaty opisano w dalszych sekcjach. Naciśnięcie przycisku "**OK**" uruchamia funkcję importu. Naciśnięcie przycisku "**Anuluj**" zamyka okno dialogowe i powoduje rezygnację z wykonywania komendy importu. Nazwy używane do identyfikacji pól w pierwszej importowanej linii można zmienić naciskając przycisk "**Słowa kluczowe**". Komendę tę opisano w dalszych sekcjach.

Pierwsza linia tekstu musi zawierać nazwę pól zgodnie z kolejnością zastosowaną w następnych liniach. Każdy obiekt musi być opisany w jednej linii tekstu. Separator "końca linii" to standardowa sekwencja systemu MS-DOS "**0d-0a**". Pola mogą występować w dowolnej kolejności. Jeśli jakichś pól brakuje, są one automatycznie wypełniane wartościami domyślnymi przez importowane opisy obiektów. Jeśli importowany obiekt istnieje już na liście edycyjnej, użytkownik musi potwierdzić czy ma on być zastąpiony. Następnie opisy obiektów są uaktualniane importowanymi polami. Jeśli jakichś pól brakuje, nie są one uaktualniane w opisie obiektów.

Dostępne formaty tekstowe

Poniżej znajduje się lista formatów dostępnych w komendzie eksportu. Komenda importu automatycznie rozpoznaje te formaty.

- separatory w postaci znaków tabulacji

Opis*:* Pola są oddzielane znakami tabulacji.

Przykład:

Nazwa	Atrybut	Komentarz
level	internal	wewnętrznie obliczany poziom wody
alrm1	output	wyjście alarmu głównego

- separatory w postaci przecinków

Opis: Pola są oddzielane przecinkami.

Przykład: Nazwa,Atrybut,Komentarz
level,internal,wewnętrznie obliczany poziom wody
alrm1,output, wyjście alarmu głównego

- separatory w formie średników

Opis: Pola są oddzielane średnikami.

Przykład: Nazwa;Atrybut;Komentarz
level;internal; wewnętrznie obliczany poziom wody
alrm1;output; wyjście alarmu głównego

- przecinki i cudzysłowy

Opis: Pola są oddzielane przecinkami.
Każde pole jest ujęte w cudzysłów.

Przykład: "Nazwa","Atrybut","Komentarz"
"level","internal"," wewnętrznie obliczany poziom wody"
"alrm1","output"," wyjście alarmu głównego "

Słowa kluczowe

Nazwy używane do identyfikacji pól w pierwszej eksportowanej lub importowanej linii mogą być zmieniane przez naciśnięcie przycisku "**Słowa kluczowe**".

Okno pokazuje listę pól obiektów oraz powiązanych z nimi słów kluczowych. Aby zmodyfikować słowo kluczowe, użytkownik musi wybrać pole z listy i nacisnąć

przycisk "**Modyfikuj**". Naciśnięcie przycisku "**DomyśInie**" przywraca pierwotną listę słów kluczowych. Nadawanie nazw słowom kluczowym musi być zgodne z następującymi regułami:

- nazwa nie może przekraczać **16** znaków
- pierwszym znakiem musi być **litera**
- kolejnymi znakami mogą być **litery**, **cyfry** lub znak '_'
- różne słowa kluczowe nie może mieć takiej samej nazwy

Poniżej znajdują się standardowe słowa kluczowe wykorzystywane przez ISaGRAF:

Nazwa obiektu	**Name**
Komentarz tekstowy	**Comment**
Adres sieciowy	**Address**
Atrybuty (wewnętrzna, wejściowa, wyjściowa)	**Atribute**
Łańcuch logiczny „fałsz”	**False**
Łańcuch logiczny „prawda”	**True**
Format analogowy (real lub integer)	**Format**
Jednostka (analogowa)	**Unit**
Nazwa konwersji (analogowa)	**Conversion**
Maksymalna długość komunikatu	**MaxLength**
Typ biblioteki bloków funkcyjnych	**Library**
Równoważność słowa zdefiniowanego	**Equivalence**
Atrybut wewnętrzny	**Internal**
Atrybut wejściowy	**Input**
Atrybut wyjściowy	**Output**
Atrybut stałej	**Constant**
Format analogowy real	**Real**
Format analogowy integer	**Integer**

A.11 Korzystanie z edytora konfiguracji We/Wy

Celem operacji konfiguracji We/Wy jest ustalenie logicznego powiązania pomiędzy zmiennymi We/Wy aplikacji a fizycznymi kanałami kart znajdujących się w sterowniku. Aby ustalić to powiązanie użytkownik musi zidentyfikować i skonfigurować wszystkie karty sterownika i powiązać zmienne We/Wy z odpowiadającymi im kanałami We/Wy.

Lista po lewej pokazuje sterownik z **gniazdami na karty**. Gniazdo może być wolne lub zajęte przez jedną kartę We/Wy lub zespół. Każdy gniazdo identyfikowane jest za pomocą **numeru porządkowego**. Sterownik może zawierać do **255** kart. Lista po prawej pokazuje parametry karty i zmienne połączone z wybraną kartą. Karta może posiadać do **128** kanałów We/Wy. Łączna liczba pojedynczych kart We/Wy (łącznie z pojedynczymi urządzeniami i zespołami) nie może przekroczyć **255**.

Ikony

Ikony wyświetlane na pierwszym planie określają typ i atrybuty zmiennych, które mogą być skojarzone z kanałami karty. System ISaGRAF nie pozwala na kojarzenie zmiennych różnych typów z tą samą kartą. Oto znaczenie stosowanych ikon:

..... typu binarnego
..... typu integer/real (mogą zostać powiązane oba typy zmiennych)
..... typu komunikat
..... wejścia - brak podłączonego kanału
..... wyjścia - brak podłączonego kanału
..... wejścia - podłączony co najmniej jeden kanał
..... wyjścia - podłączony co najmniej jeden kanał

Poniżej przedstawiono ikony zastosowane do oznaczenia typu urządzenia We/Wy zainstalowanego w gnieździe:

..... zespół We/Wy
..... rzeczywista karta We/Wy
..... wirtualna karta We/Wy

Poniżej przedstawiono ikony stosowane do kreślenia parametru lub kanału:

..... parametr karty
..... wolny kanał
..... podłączony kanał

Przemieszczanie kart na liście

Aby przemieścić wybraną kartę We/Wy o jedną linię w górę lub w dół na liście głównej, należy użyć tych przycisków na pasku narzędzi lub komend menu "**Edycja / Przemieść karty do góry / w dół** ". Komenda "**Edycja / Wstaw gniazdo**" wstawia w bieżącej pozycji puste gniazdo.

A.11.1 Definiowanie kart We/Wy

Menu "**Edycja**" zawiera podstawowe komendy definiujące wybraną kartę (ustawianie parametrów) i pozwalające skojarzyć zmienne We/Wy z kanałami.

Wybór typu karty We/Wy

Przed skojarzeniem zmiennych We/Wy z kartą, należy wprowadzić dane identyfikacyjne karty. Pakiet ISaGRAF udostępnia bibliotekę wstępnie zdefiniowanych kart. Biblioteka taka może być przygotowana przez jednego lub kilku dostawców urządzeń We/Wy. Komenda "**Edycja / Ustaw kartę/zespół**" jest stosowana do konfiguracji danych identyfikacyjnych karty. Komenda ta może być używana do wyboru pojedynczej karty lub zespołu We/Wy z biblioteki ISaGRAF. Można również dwukrotnie kliknąć dane gniazdo, aby ustawić odpowiadającą mu kartę lub zespół.
Wszystkie kanały na pojedynczej karcie są tego samego typu (binarnne, integer/real lub komunikatowe) i posiadać ten sam kierunek (wejściowe lub wyjściowe). Zmienne typu real i integer nie są rozróżniane przy kojarzeniu z We/Wy. Zespół We/Wy to urządzenie We/Wy posiadające kanały różnych typów i o różnych kierunkach. Zespół We/Wy przedstawiane jest jako lista pojedynczych kart We/Wy. Wykorzystuje on tylko jedno gniazdo z listy.

Usuwanie karty

Komenda "**Edycja / Wyczyść gniazdo**" używana jest do usuwania wybranej aktualnie karty lub zespołu We/Wy. Jeśli zmienne są skojarzone z kanałami, są automatycznie odłączane przy czyszczeniu gniazda.

Karty rzeczywiste i wirtualne

Komenda "**Edycja / Karta rzeczywista/wirtualna** " ustala czy wybrana karta lub zespół We/Wy istnieje. Na liście wyświetlane są następujące ikony wskazujące tryb karty:

..... rzeczywista karta We/Wy
..... wirtualna karta We/Wy

W **trybie rzeczywistym**, zmienne We/Wy są skojarzone bezpośrednio z odpowiadającymi im urządzeniami We/Wy. Operacje wejścia lub wyjścia w

programie aplikacyjnym powiązane są bezpośrednio z odpowiadającymi im wejściami i wyjściami konkretnych urządzeń zewnętrznych We/Wy. W **trybie wirtualnym**, zmienne We/Wy są przetwarzane dokładnie tak, jak zmienne wewnętrzne. Mogą one być odczytywane lub uaktualniane przez debager tak, aby użytkownik mógł zasymulować przetwarzanie We/Wy. Operacje te dzieją się bez powiązania ze światem fizycznym.

Opisy

Komenda "**Narzędzia / Opis**" wyświetla opis przeznaczony dla użytkownika i dotyczący wybranej karty lub zespołu. Opisy techniczne kart sporządzone są przez dostawcę karty We/Wy. Zawierają one wszelkie informacje dotyczące zarządzania kartą We/Wy. Opisują również znaczenie parametrów.

Usuwanie skojarzonych zmiennych

Komenda "**Narzędzia / Zwolnij kanały karty**" odłącza wszystkie zmienne We/Wy wcześniej skojarzone z wybraną kartą.

Komentarze

Tekst komentarza zadeklarowanej zmiennej We/Wy jest wyświetlany wraz z jej nazwą na liście karty. Ponieważ ISaGRAF umożliwia również wykorzystywanie zmiennych o reprezentacji bezpośredniej (zapis z %), komentarze mogą być kojarzone z wolnymi kanałami. Aby wprowadzić komentarz dla wolnego kanału wybranego aktualnie z listy kart, wykorzystaj komendę "**Narzędzia / Ustaw komentarz do kanału**". Komenda ta nie może być wykorzystywana do zmiany komentarza zmiennej We/Wy zadeklarowanej w słowniku projektu

A.11.2 Konfiguracja parametrów karty

Aby ustawić wartość parametru karty, użytkownik musi dwukrotnie kliknąć jej nazwę na liście po prawej stronie. Możliwe jest również jej wybranie (podświetlenie) i wykonanie komendy "**Ustaw kanał/parametr**" z menu "**Edycja**". Parametry są wymienione na początku listy. Do ich oznaczenia na liście stosuje się następującą ikonę:

..... parametr karty

Znaczenie i format wprowadzanego parametru są opracowane przez dostawcę danej karty lub zespołu We/Wy. Aby uzyskać więcej informacji na temat parametrów karty, wybierz komendę "**Narzędzia / Opis**" lub skorzystaj z dokumentacji karty.

A.11.3 Podłączanie kanałów We/Wy

Aby ustanowić połączenie kanału, użytkownik musi dwukrotnie kliknąć jego lokalizację na liście po prawej stronie. Można ją również wybrać (podświetlić) i wykonać komendę "**Edycja / Ustaw kanał/parametr**". Następujące ikony przedstawiają kanały na liście:

 ..... wolny kanał
 ..... podłączony kanał

Lista zawiera wszystkie zmienne, które pasują do kart wybranego typu i kierunku. Wymienione są na niej jedynie nie skojarzone jeszcze zmienne. Przycisk "**Połącz**" łączy wybrane z listy zmienne z wybranym kanałem. Przycisk "**Zwolnij**" usuwa (odłącza) zmienną z wybranego kanału. Przyciski "**Następny**" i "**Poprzedni**" są używane do wybrania następnego kanału karty. Lokalizacja wybranego kanału jest zawsze wyświetlana w tytule okna dialogowego.

A.11.4 Zmienne o reprezentacji bezpośredniej

Wolne kanały to te, które nie są skojarzone z zadeklarowaną zmienną We/Wy. ISaGRAF umożliwia stosowanie w programach źródłowych **zmiennych o reprezentacji bezpośredniej** do przedstawiania kanałów wolnych. Identyfikator zmiennej o reprezentacji bezpośredniej rozpoczyna się zawsze od znaku "**%**".

Poniżej przedstawiono konwencje nazywania zmiennych o reprezentacji bezpośredniej dla kanału lub pojedynczej karty. "***s***" to numer gniazda karty. "***c***" to numer kanału.

%IX*s.c* ..... wolny kanał karty wejść binarnych
%ID*s.c* ..... wolny kanał karty wejść typu integer
%IS*s.c* ..... wolny kanał karty wejść w postaci komunikatu
%QX*s.c* ..... wolny kanał karty wyjść binarnych
%QD*s.c* ..... wolny kanał karty wyjść typu integer
%QS*s.c* ..... wolny kanał karty wyjść w postaci komunikatu

Poniżej przedstawiono konwencje nazywania zmiennych o reprezentacji bezpośredniej dla kanału zespołu "***s***" to numer gniazda zespołu. "***b***" to indeks pojedynczej karty w zespole.
"***c***" to numer kanału.

%IX*s.b.c* ..... wolny kanał karty wejść binarnych
%ID*s.b.c* ..... wolny kanał karty wejść typu integer
%IS*s.b.c* ..... wolny kanał karty wejść typu komunikat
%QX*s.b.c* ..... wolny kanał karty wyjść binarnych
%QD*s.b.c* ..... wolny kanał karty wyjść typu integer
%QS*s.b.c* ..... wolny kanał karty wyjść typu komunikat

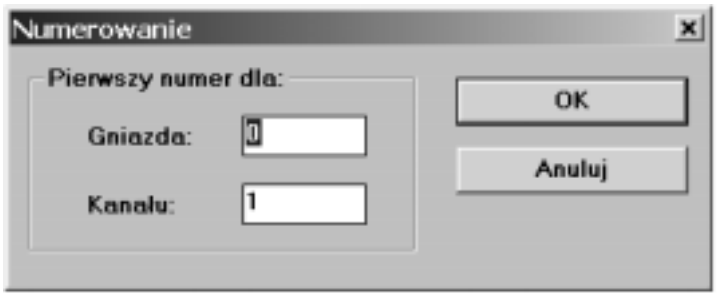
Oto przykłady:

```
%QX1.6  6-ty kanał karty nr 1 (wyjście binarne)
%ID2.1.7        7-my kanał karty nr 1w zespole nr 2 (wejście typu integer)
```

Zmienna o reprezentacji bezpośredniej nie może być typu "**real**".

A.11.5 Numerowanie

Aby ustalić konwencje numerowania, stosuje się komendę "**Opcje / Numerowanie**". W następującym oknie dialogowym można określić numer używany dla pierwszego gniazda oraz numer używany dla pierwszego kanału każdej z kart.



Numerowanie gniazd rozpoczyna się domyślnie od indeksu **0**, a numerowanie kanałów od indeksu **1**.

Uwaga: zmieniając konwencję numerowania należy być bardzo ostrożnym, ponieważ ma to wpływ na symbole stosowane dla zmiennych o reprezentacji bezpośredniej i może prowadzić do błędów w kompilacji jeśli zmienne We/Wy o reprezentacji bezpośredniej są używane w istniejących programach.

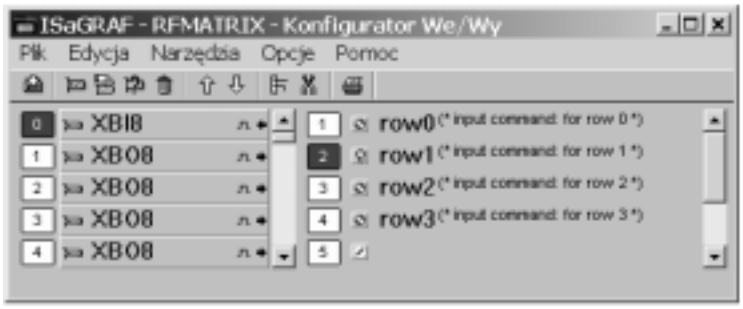
A.11.6 Ustawianie zabezpieczeń indywidualnych

Pakiet ISaGRAF zapewnia pełny system ochrony oparty na hierarchii haseł. Połączenia We/Wy mogą być globalnie chronione przy pomocy hasła. Ponadto ISaGRAF umożliwia ustawianie zabezpieczeń indywidualnych na dowolnym kanale We/Wy, przy założeniu, że:

- hasła zostały już zdefiniowane w systemie definicji haseł (przy użyciu komendy "**Projekt / Ustaw hasło**" w oknie Zarządzanie Projektami) tak, iż dostępne są poziomy ochrony do wykorzystania w zabezpieczeniach indywidualnych.

- poziomy ochrony używane przez użytkownika do zabezpieczeń indywidualnych mają wyższy priorytet niż globalne zabezpieczenie We/Wy.

Kiedy kanał We/Wy posiada zabezpieczenie indywidualne, obok jego nazwy w oknie połączeń pojawia się niewielka ikona:



Do ustawienia lub usunięcia zabezpieczenia indywidualnego dla wybranego kanału służą komendy "**Ustaw zabezpieczenie**" i "**Usuń zabezpieczenie**" w menu "**Edycja**". Obie komendy wymagają podania prawidłowego hasła, umożliwiającego przypisanie poziomu zabezpieczeń do kanału. Za każdym razem przy próbie zmiany połączenia z kanałem posiadającym zabezpieczenie indywidualne trzeba podać hasło o odpowiednim priorytecie.

Uwaga: jeśli kanał jest zabezpieczony na pewnym poziomie i odpowiadające mu hasło zostanie usunięte z systemu ochrony, a nie jest zdefiniowane hasło wyższego poziomu, połączenie z tym kanałem nie będzie już mogło zostać zmienione, do czasu aż zostanie zdefiniowane nowe hasło o wystarczającym priorytecie.

A.12 Tworzenie tablic konwersji

Pakiet IsaGRAF pozwala użytkownikowi na tworzenie tablic konwersji. Tablica konwersji to zestaw punktów wykorzystywanych do zdefiniowania konwersji analogowej. Tablica konwersji może być skojarzona ze zmienną analogową wejściową lub wyjściową. Tablica tworzy proporcjonalne powiązania pomiędzy wartościami elektrycznymi (odczytywanymi na czujniku wejściowym lub wysyłanymi do urządzenia wyjściowego) a wartościami fizycznymi (wykorzystywanymi w programowaniu aplikacji).

Tablice konwersji są edytowane w oparciu o okno dialogowe wywoływane komendą "**Narzędzia / Tablice konwersji**" w oknie słownika ISaGRAF

Zdefiniowana tablica konwersji może być wykorzystywana do filtrowania wartości dowolnej analogowej zmiennej wejściowej lub wyjściowej wybranego projektu. Tablicę konwersji kojarzy się ze zmienną przy użyciu komend edytora deklaracji zmiennych w słowniku ISaGRAF. Następnie należy wybrać analogową zmienną wejściową lub wyjściową i dokonać edycji jej parametrów. Do zmiennej nie można skojarzyć z tablicą konwersji, która nie została wcześniej definiowana.

A.12.1 Komendy podstawowe

Okno dialogowe "**Tablice konwersji**" pokazuje listę zdefiniowanych tablic konwersji i zawiera przyciski komend podstawowych, służących do edycji istniejących tablic (definiowania ich punktów), tworzenia nowych tablic oraz do zmieniania ich nazwy i ich usuwania.

Naciśnij OK aby wyjść z okna dialogowego "**Tablice konwersji**" i zapisać tablice na dysku.

Tworzenie nowej tablicy

Komenda "**Nowa**" pozwala użytkownikowi stworzyć nową tablicę konwersji. Można utworzyć do **127** tablic konwersji dla każdego projektu. Do kodu wykonywalnego aplikacji wstawiane są tylko tablice wykorzystywane (skojarzone ze zmiennymi). Nadawanie tablicom nazw musi być zgodne z następującymi zasadami:

- nazwa nie może przekraczać **16** znaków
- pierwszym znakiem musi być **litera**
- kolejnymi znakami mogą być **litery**, **cyfry** lub znak ('_')
- nazwa tablicy nie uwzględnia wielkości liter

Zmiana zawartości tablicy

Komenda "**Edycja**" jest używana do wprowadzania punktów tablicy wybieranych z listy. Można również kliknąć dwukrotnie nazwę tablicy. Komenda "**Edycja**" wywoływana jest automatycznie, gdy tworzona jest nowa tablica. Do każdej tablicy muszą zostać wprowadzone co najmniej dwa punkty.

A.12.2 Wprowadzanie punktów tablicy

Okno edycyjne pozwala użytkownikowi na zdefiniowanie punktów tablicy. Pole po lewej stronie pokazuje listę zdefiniowanych już punktów. Pole w prawej dolnej części pokazuje zdefiniowaną tablicę w formie graficznej. Punkty wprowadza się przy użyciu komend. Użytkownik musi przestrzegać reguł dotyczących numeracji definiowanych punktów, co opisano pod koniec tego rozdziału. Pole po lewej zawiera zawsze listę istniejących punktów aktualnie edytowanej tablicy. Kolumna po lewej pokazuje wartość elektryczną (zewnętrzną) punktów. Kolumna po prawej pokazuje wartości fizyczne (wewnętrzne). Użytkownik musi wybrać punkt z listy, aby zmodyfikować jego wartości lub aby żeby go usunąć (wyczyścić). Ostatnia pozycja listy ("**... ...**") jest wykorzystywana do definiowania nowego punktu. Pole w lewej dolnej części pokazuje aktualnie edytowaną tablicę w postaci krzywej. Jest to odwzorowanie proporcjonalne krzywej, więc nie zawiera ono osi ani współrzędnych. Takie odwzorowanie jest przydatne do szybkiego sprawdzenia czy krzywa jest zdefiniowana prawidłowo.

- ***Definiowanie nowego punktu***

Definiując nowy punkt należy wybrać ostatnią pozycję ("**... ...**") z listy punktów. Jest to również domyślny tryb przy rozpoczynaniu definiowania nowej tablicy. Użytkownik musi wprowadzić wartości elektryczne (zewnętrzne) i fizyczne (wewnętrzne) dla każdego punktu. Wartości przechowywane są jako liczby zmiennoprzecinkowe pojedynczej precyzji. Należy pamiętać, że wprowadzone muszą zostać co najmniej **dwa punkty** definiujące krzywą. Po zdefiniowaniu obu wartości naciśnięcie przycisku "**Zeskładuj**" dodaje punk do tablicy. W każdej tablicy konwersji można zdefiniować maksymalnie **32** punkty.

- ***Modyfikacja punktu***

Aby zmodyfikować wartości istniejącego punktu należy najpierw wybrać go z listy. Można wprowadzić nowe wartości elektryczne (zewnętrzne) i fizyczne (wewnętrzne) punktu. Wartości są przechowywane jako liczby zmiennoprzecinkowe pojedynczej precyzji. Po wprowadzeniu obu wartości naciśnięcie przycisku " **Zeskładuj**" aktualizuje punkt w tablicy.

- ***Usuwanie punktu***

Istniejący punk czyści się wybierając go z listy i naciskając przycisk "Czyść". Proszę pamiętać, że aby zdefiniować tablicę należy wprowadzić co najmniej dwa punkty.

A.12.3 Zasady i limity

Poniższe reguły muszą być przestrzegane przy definiowaniu tablicy konwersji. Tablica może być wykorzystywana do konwersji zarówno analogowych zmiennych wejściowych jak i wyjściowych:

- Dwa punkty nie mogą być zdefiniowane tą samą wartością elektryczną
- Krzywa musi rosnąć lub opadać w sposób ciągły
- Dwa punkty nie mogą być zdefiniowane tą samą wartością fizyczną

Definiowanie tablicy konwersji dla projektu podlega następującym ograniczeniom:

- W jednym projekcie nie może być zdefiniowanych więcej niż **127** tablic konwersji
- W jednej tablicy konwersji nie może byś zdefiniowanych więcej niż **32** punkty.

A.13 Korzystanie z generatora kodu

Okno generowania kodu jest otwierane automatycznie przez komendy "Weryfikuj" i "Utwórz kod programu" w innych oknach pakietu ISaGRAF. Okno generowania kodu nie jest zamykane automatycznie, kiedy zakończona zostaje żądana operacja generowania kodu, a więc użytkownik ma nadal dostęp do komend i opcji generowania kodu z menu tego okna.

A.13.1 Komendy główne

Menu **"Pliki"** zawiera programy do kontroli składni i generowania kodu programu.

Tworzenie kodu aplikacji

Komenda "**Utwórz kod programu**" buduje całość kodu dla projektu. Przed wygenerowaniem czegokolwiek, komenda ta sprawdza składnię deklaracji i programów. Każdy błąd, którego nie można wykryć podczas kompilacji pojedynczego programu zostanie wykryty przy generowaniu kodu. Odnosi się to do tablic konwersji, połączeń zmiennych We/Wy oraz powiązań z bibliotekami. Po wykryciu błędów generator kodów zatrzymuje kompilowanie programu. Program ten musi zostać poprawiony przed kontynuowaniem generowania kodu. Programy, które zostały już sprawdzone (w których nie wykryto błędów) i które nie zostały zmodyfikowane od ostatniego przeprowadzenia operacji "**Weryfikuj**" nie są ponownie kompilowane. Zawsze przetwarzana jest weryfikacja deklaracji zmiennych i kontrola spójności aplikacji. Podczas kontroli programu operacja "**Utwórz kod programu**" może zostać przerwana naciśnięciem klawisza **ESC**.

Uwaga: Jeśli deklaracja zmiennej lokalnej programu została zmodyfikowana, program ten podlega weryfikacji. Jeśli zmodyfikowana została zmienna globalna, weryfikacji podlegają wszystkie programy.

Kontrola składni programu

Komenda "**Weryfikuj program**" pozwala użytkownikowi na weryfikację tylko jednego programu. Wybrany program jest kompilowany, nawet jeśli nie został zmodyfikowany od ostatniej weryfikacji. Komenda "**Weryfikuj słownik**" pozwala użytkownikowi zweryfikować deklaracje wszystkich zmiennych w projekcie.
"**Weryfikuj wszystkie programy**" sprawdza składnię we wszystkich programach projektu, nawet jeśli niektóre z nich nie zostały zmodyfikowane. Komenda ta **nie** zatrzymuje się, kiedy w programie zostanie wykryty błąd. Można ją wykorzystywać do tworzenia pełnej listy wszystkich błędów, które pozostały w programach projektu. Komendę tę można przerwać naciskając klawisz **ESC**.

Symulowanie modyfikacji

Komenda "**Uznaj źródła za zmodyfikowane**" symuluje wprowadzenie modyfikacji do wszystkich programów projektu tak, aby wszystkie zostały zweryfikowane podczas następnej operacji "**Utwórz kod programu**". Komenda "**Otwórz**" używana jest do otwierania weryfikowanego ostatnio programu. Komenda ta jest wielce przydatna do uzyskania bezpośredniego dostępu do programu, w którym wykryto błędy składni.

A.13.2 Opcje kompilatora

Komenda "**Opcje kompilatora**" używana jest do konfiguracji głównych parametrów wykorzystywanych przez Generator Kodu ISaGRAF w celu zbudowania i zoptymalizowania kodu oprogramowania wbudowanego. Celem tej komendy jest wybranie typu kodu, który musi zostać wygenerowany, zgodnie z odpowiednim oprogramowaniem wbudowanym ISaGRAF oraz konfiguracja optymalizatora parametrów zgodnie z oczekiwanym czasem wykonywania i wymogami wykonawczymi aplikacji.
Przycisk "**Pobierz...**" otwiera drugie okno dialogowe z innymi opcjami, które umożliwiają osadzanie spakowanego kodu źródłowego w kodzie załadowywanym, aby uaktywnić funkcję "Pobierz". Dalsze wyjaśnienia znajdują się w rozdziale "Pobieranie".

Wybór oprogramowania wbudowanego

Górna lista to lista dostępnych kodów oprogramowania wbudowanego, które mogą zostać stworzone. Znak "**>>**" używany jest do oznaczania wybranego oprogramowania wbudowanego. Generator Kodu ISaGRAF może utworzyć do **3** różnych kodów dla jednej operacji kompilacji. Przycisków "**Zaznacz**" i "**Usuń**" używa się do wyznaczenia wymaganych kodów oprogramowania wbudowanego, zgodnych z parametrami sprzętu, na którym pracuje oprogramowanie wbudowane. Poniżej przedstawiono standardowe typy oprogramowania wbudowanego ISaGRAF:

SIMULATE: ..... Kod ten jest przyporządkowany do symulatora ISaGRAF w pakiecie. Symulator nie może zostać uruchomiony jeśli do tworzenia kodu aplikacji nie wybrano tego oprogramowania wbudowanego.

ISA86M: ..... Jest to kod TIC (Target Independent Code - kod niezależny od oprogramowania wbudowanego) przyporządkowany do jąder ISaGRAF zainstalowanych w procesorach firmy Intel. Typ procesora wpływa jedynie na organizację bajtów w wygenerowanym kodzie.

ISA68M: ..... Jest to kod TIC (Target Independent Code - kod niezależny od oprogramowania wbudowanego) przyporządkowany do jąder ISaGRAF zainstalowanych w procesorach firmy Motorola. Typ

procesora wpływa jedynie na organizację bajtów w wygenerowanym kodzie.

SCC: ..... Wybór tego oprogramowania wbudowanego prowadzi do stworzenia przez kompilator ISaGRAF strukturalizowanego kodu źródłowego w języku "C" do skompilowania i połączenia z bibliotekami jądra oprogramowania wbudowanego ISaGRAF w celu stworzenia osadzonego kodu wykonywalnego.

CC86M: ..... Wybór tego oprogramowania wbudowanego prowadzi do stworzenia przez kompilator ISaGRAF nie strukturalizowanego kodu źródłowego w języku "C" do skompilowania i połączenia z bibliotekami jądra oprogramowania wbudowanego ISaGRAF w celu stworzenia osadzonego kodu wykonywalnego. Wybór ten jest zapewniany w celu zachowania kompatybilności z ISaGRAF w wersjach wcześniejszych niż V3.23, w których generowanie i integrowanie strukturalizowanego kodu "C" nie było obsługiwane.

Aby dowiedzieć się jaki typ jądra oprogramowania wbudowanego ISaGRAF zainstalowany jest w sterowniku PLC, należy zapoznać się z instrukcją sprzętową. Obsługa innych typów oprogramowania wbudowanego (kod maszynowy, kod źródłowy C, itp.) mogą zostać dodane w przyszłych wersjach pakietu ISaGRAF.

Przetwarzanie SFC

Aby włączyć używanie mechanizmu SFC ISaGRAF należy zaznaczyć okienko "**Użyj osadzonego mechanizmu SFC**". ***Powinien to być preferowany tryb pracy, gdyż prowadzi on do wyższej efektywności działania***. Jednakże mechanizm oprogramowania wbudowanego może nie występować w pewnych szczególnych implementacjach oprogramowania wbudowanego ISaGRAF, czy też częściej, w indywidualnie dostosowanych systemach oprogramowania wbudowanego opartych na przetwarzaniu końcowym kodu ISaGRAF. W takich przypadkach konieczne może być wyłączenie tej opcji i zezwolenie kompilatorowi ISaGRAF na przełożenie grafów SFC na instrukcje niskiego poziomu. Więcej informacji na temat wykorzystania tej opcji znajduje się w dokumentacji dotyczącej sprzętu.

Opcje optymalizatora

Poniżej podano parametry wykorzystywane przez Generator Kodu ISaGRAF do optymalizacji kodu oprogramowania wbudowanego, którą można skonfigurować za pośrednictwem okna dialogowego "**Opcje kompilatora**". Przycisk "**DomyślnIe**" używany jest do wyłączenia wszelkich opcji optymalizacyjnych w celu skrócenia czasu kompilacji.

Kiedy włączona jest opcja "**Optymalizacja dwuprzebiegowa**", Optymalizator Kodu ISaGRAF jest uruchamiany dwa razy. Optymalizacje dokonane podczas drugiego przejścia są na ogół mniej znaczące niż te dokonane podczas pierwszego przejścia.

- Kiedy włączona jest opcja "**Przelicz stałe wyrażenia**", kompilator ocenia wyrażenia stałe. Na przykład, wyrażenie liczbowe "**2 + 3**" zamieniane jest w kodzie oprogramowania wbudowanego na "**5**". Kiedy opcja ta nie jest włączona, wyrażenia stałe obliczane są w czasie wykonania.

- Kiedy włączona jest opcja "**Pomiń nieużywane etykiety**", Optymalizator upraszcza system skoków i etykiet programów w celu pominięcia używanych etykiet docelowych czy pustych skoków.

- Kiedy włączona jest opcja "**Optymalizuj kopiowanie zmiennych**", wykorzystanie zmiennych tymczasowych (używanych do przechowywania wyników pośrednich) zostaje zoptymalizowane. Opcja ta jest zazwyczaj używana wspólnie z opcją "**Optymalizuj wyrażenia**". Kiedy opcja ta jest włączona Optymalizator ponownie wykorzystuje wyniki wyrażeń lub wyrażeń podrzędnych, które są wykorzystywane w programie więcej niż raz.

- Kiedy włączona jest opcja "**Pomiń nieużywany kod**", Optymalizator pomija kod, który nie jest znaczący. Na przykład, jeśli zaprogramowane zostaną następujące instrukcje: "**var := 1; var := X;**", odpowiadający im wygenerowany kod to tylko: "**var := X;**".

- Kiedy włączona jest opcja "**Optymalizuj operacje arytmetyczne**", Optymalizator upraszcza operacje arytmetyczne zgodnie ze specjalnymi argumentami. Na przykład, wyrażenie "**A + 0**" zostanie zamienione na "**A**". Kiedy włączona jest opcja "**Optymalizuj operacje binarne**", Optymalizator upraszcza operacje binarne zgodnie ze specjalnymi argumentami. Na przykład, wyrażenie logiczne "**A & A**" zostanie zamienione na "**A**".

- Kiedy włączona jest opcja "**Buduj drzewo binarne**", Optymalizator zamienia równania logiczne (łączące operatory **I**, **LUB** , **LUB WYŁĄCZAJĄCE** oraz **NIE**), na ograniczoną listę warunkowych operacji skoku. Taka translacja jest wykonywana jedynie wtedy, gdy spodziewany czas wykonania sekwencji skoków jest krótszy niż spodziewany czas wykonywania wyrażenia pierwotnego.

  Poniższa tabela przedstawia zestawienie spodziewanej optymalizacji i żądanego czasu kompilacji, odpowiadających każdemu z parametrów:

	zysk (wydajność)	czas kompilacji
Wykonaj 2 przejścia	**xxxx**	(*)
Optymalizuj wyrażenia stałe	**xxxxxxxxxx**	**xxxx**
Ukryj nieużywane etykiety	**xxxx**	**xxxxxxxxxx**
Optymalizuj kopiowanie zmiennych	**xxxx**	**xxxxxxxxxx**
Optymalizuj wyrażenia	**xxxx**	**xxxxxxxxxx**
Ukryj nieużywany kod	**xxxx**	**xxxxxxxxxx**
Optymalizuj operacje arytmetyczne	**xxxx**	**xxxxxxxxxx**

Optymalizuj operacje binarne **xxxxxxxxxx** ..... **xxxx**
Buduj binarne diagramy decyzyjne **xxxxxxxxxxxxxx** ....**xxxxxxxxxxxxxx**

(*) czas kompilacji jest również mnożony razy 2.

A.13.3 Tworzenie kodu źródłowego C

Pakiet ISaGRAF umożliwia tworzenie kodu źródłowego w języku "C". W takim przypadku cała zawartość aplikacji, łącznie z opisem grafu SFC, definicją bazy danych i sekwencjami kodów generowana jest w formacie kodu źródłowego "C". Istnieją dwie możliwości, proponowane jako dwa style generowanego kodu:

CC86M (kod źródłowy C - V3.04) tworzy nie strukturalizowany kod źródłowy "C". Styl ten należy wybrać wtedy, gdy oprogramowanie wbudowane oparte jest na wersji ISaGRAF wcześniejszej niż 3.23.

SCC (strukturalizowany kod źródłowy C) tworzy strukturalizowany kod źródłowy "C". Styl ten powinien być preferowany wtedy, gdy oprogramowanie wbudowane oparte jest na ISaGRAF w wersji 3.23 lub późniejszej.

W katalogu projektu tworzone są następujące dwa pliki:

APPLI.C **..... wspólny kod źródłowy aplikacji**
APPLI.H **..... wspólne definicje języka "C"**

W przypadku generowania strukturalizowanego kodu "C", dodatkowo poza wspólnymi plikami "**APPLI.C**" i "**APPLI.H**" dla każdego programu aplikacji tworzony jest plik źródłowy "**.C**" oraz plik definicji "**.H**".

Pliki te muszą być skompilowane i połączone z bibliotekami oprogramowania wbudowanego ISaGRAF w celu stworzenia gotowego kodu wykonywalnego. Dalsze informacje dotyczące zalecanych technik implementacji znajdują się w "Poradniku Użytkownika pakietu narzędziowego We/Wy ISaGRAF".

Uwaga: Niektóre funkcje debagera takie, jak ładowanie aplikacji, modyfikacja natychmiastowa i pułapki nie są dostępne po skompilowaniu aplikacji ISaGRAF w "C".

A.13.4 Przeglądanie informacji

Menu "**Edycja**" zawiera komendy pozwalające przeglądać różne pliki tekstowe, tworzone podczas operacji generowania kodu lub kontroli składni w oknie generowania kodu. Okno generowania kodu to obszar tekstowy, zawierający komunikaty pojawiające się podczas operacji generowania kodu i kontroli składni. Wszystkie informacje przechowywane są na dysku, wiec można je sprawdzać przy użyciu komend menu "**Edycja**".

- ***Komendy edycyjne***

  Komenda "**Wyczyść ekran**" używana jest do czyszczenia obszaru tekstowego okna. Okno jest czyszczone automatycznie przed każdą operacją generowania kodu lub kontroli składni. Komenda "**Kopiuj**" jest używana do kopiowania wyświetlanego tekstu do schowka systemu Windows tak, aby można było wykorzystać go w innych aplikacjach, jak edytory tekstu ISaGRAF.

- ***Przeglądanie komunikatów wyjściowych kompilatora***

  Komenda "**komunikaty wykonania**" pokazuje wszystkie komunikaty wyświetlone podczas wykonania ostatniej operacji "**Utwórz kod programu**" lub "**Weryfikuj**" w obszarze tekstowym okna. Odnosi się to do wszystkich komunikatów o błędach.

  Inne opcje menu "**Edycja**" pozwalają użytkownikowi na monitorowanie pomocniczych plików tekstowych tworzonych podczas weryfikacji składni i generowania kodu. Pliki te nie są zazwyczaj wykorzystywane w standardowym projekcie ISaGRAF.

A.13.5 Definiowanie zasobów

Komenda "**Zasoby**" menu "**Opcje**" pozwala użytkownikowi zdefiniować zasoby. Zasoby, to wszelkie dane zdefiniowane przez użytkownika (konfiguracja sieciowa, ustawienia sprzętowe, itp.) w dowolnym formacie (plik, lista wartości) jakie muszą zostać włączone do generowanego kodu w celu ich załadowania wraz z tym kodem do sterownika PLC z oprogramowaniem wbudowanym. Dane takie nie są obsługiwane bezpośrednio przez jądro ISaGRAF i są zazwyczaj dedykowane pod inne oprogramowanie zainstalowane w PLC z oprogramowaniem wbudowanym. Dalsze informacje dotyczące dostępnych zasobów znajdują się w instrukcji sprzętowej.

- Plik definicji zasobów
  Zasoby są definiowane w "**Pliku definicji zasobów**" przechowywanym wraz z innymi plikami projektu ISaGRAF. Jest to plik tekstowy w formacie ASCII, przetwarzany przez Kompilator Zasobów ISaGRAF. Kompilator ten jest uruchamiany automatycznie, kiedy budowany jest kod aplikacji. Niniejsza sekcja wyjaśnia składnię tego pliku. Plik definicji zasobów wykorzystuje zasady leksykalne języka ST. Komentarze, rozpoczynające się od znaków "**(***" i kończące się znakami "***)**", można wstawiać w dowolnym miejscu tekstu. Ciągi są zawarte pomiędzy pojedynczymi apostrofami. Dalsze wyjaśnienia dotyczące formatów leksykalnych wykorzystywanych do wprowadzania wartości liczbowych znajdują się w drugiej części niniejszej instrukcji.

- Informacje o językach
  Poniżej przedstawiono listę słów źródłowych i instrukcji wykorzystywanych w pliku definicji zasobów

  ULONGDATA

Znaczenie: Określa zasoby stanowiące listę wartości typu integer. Wartości są przechowywane w kodzie oprogramowania wbudowanego w formie 32-bitowych liczb integer bez znaku. Wartości są składowane w kolejności określonej w pliku definicji zasobów. Wartości muszą być oddzielone przecinkami. Nazwa zasobu nie może być dłuższa niż 15 znaków.

Składnia:
```
ULONGDATA '<nazwa_zasobu>'
  BEGIN
        ...wybór oprogramowania wbudowanego...
        ...lista wartości...
  END
```

Przykład:
```
ULongData 'MYDATA'
Begin
      ...
      0, -1, 100_000,       (* dziesiętna *)
      16#A0B1, 2#1011_0101  (*       szesnastkowa,
binarna *)
End
```

VARLIST

Znaczenie: Określa zasoby stanowiące listę adresów zmiennych. Zmienne są identyfikowane przy pomocy ich nazwy w pliku definicji zasobów. Adresy zmiennych są przechowywane w kodzie oprogramowania wbudowanego w fomie16-bitowych liczb integer bez znaku. Adresy są składowane w kolejności określonej w pliku definicji zasobów. Zmienne muszą być oddzielone przecinkami. Nazwa zasobu nie może być dłuższa niż 15 znaków.

Składnia:
```
VARLIST '<nazwa_zasobu>'
  BEGIN
     ... wybór oprogramowania wbudowanego...
     ...lista nazw zmiennych...
  END
```

Przykład:
```
VarList 'LIST'
  Begin
     ...
     Var100, MyParameter, Command, Alarm
  End
```

BINARYFILE

Znaczenie: Określa zasób będący plikiem binarnym. Dane źródłowe są przechowywane w pliku systemu MS-DOS. Definicja zasobu oprogramowania wbudowanego jest uzupełniona ścieżką do tego oprogramowania. Znaki końca linii nie są przetwarzane przez Kompilator Zasobów ISaGRAF. Nazwa zasobu nie może być dłuższa niż **15** znaków.

Składnia:
```
BINARYFILE '<nazwa_zasobu>'
```

```
BEGIN
     ... wybór oprogramowania wbudowanego...
     FROM '<ścieżka_źródłowa>'
     TO '<ścieżka_docelowa>'
END
```

Przykład:
```
BinaryFile 'MYFILE'
 Begin
     ...
     From 'c:\user\config.bin'
     To '/dd/user/appl/config.dat'
 End
```

TEXTFILE

Znaczenie: Określa zasób będący plikiem tekstowym. Dane źródłowe przechowywane są w pliku ASCII. Definicja zasobu oprogramowania wbudowanego jest uzupełniona ścieżką do tego oprogramowania. Znaki końca linii są przetwarzane przez Kompilator Zasobów ISaGRAF zgodnie z konwencjami systemu oprogramowania wbudowanego hosta. Nazwa zasobu nie może być dłuższa niż **15** znaków.

Składnia:
```
TEXTFILE '<nazwa_zasobu>'
     BEGIN
     ... wybór oprogramowania wbudowanego...
     FROM '<ścieżka_źródłowa>'
     TO '<ścieżka_docelowa>'
     END
```

Przykład:
```
TextFile 'MYFILE'
     Begin
     ...
     From 'c:\user\config.bin'
     To '/dd/user/appl/config.dat'
     End
```

TARGET

Znaczenie: Określa nazwę kodu oprogramowania wbudowanego, która musi zostać zawarta w zasobie. Dalsze informacje na temat obsługiwanych oprogramowania wbudowanego znajdują się w poprzedniej sekcji (opcje kompilatora). Instrukcja "**Sterownik**" może pojawiać się więcej niż jeden raz w tym samym bloku zasobów, pozwalając wybrać kilka realizacji oprogramowania wbudowanego. Instrukcja ta nie może być użyta, jeśli zastosowano instrukcję "**Dowolne oprogramowanie sterownika**".

Składnia:
```
TARGET '<nazwa_opr_wbud>'
```

Przykład:
```
BinaryFile 'MYFILE'
 Begin
     Target 'ISA86M'
```

```
     Target 'ISA68M'
     ...
  End
```

ANYTARGET

Znaczenie: Określa, iż zasób musi zostać połączony z wszystkimi kodami oprogramowania wbudowanego utworzonego przy pomocy Generatora Kodu. Generator Kodu ISaGRAF może tworzyć kilka kodów oprogramowania wbudowanego w ramach jednej komendy "**Utwórz kod programu**". Instrukcja ta nie może zostać użyta jeśli zastosowano jedną lub więcej instrukcji "**Sterownik**".

Składnia: **ANYTARGET**

Przykład:
```
ULongData 'MYDATA'
  Begin
     AnyTarget
     ...
  End
```

FROM

Znaczenie: Określa nazwę ścieżki źródłowej (w komputerze PC, na którym zainstalowany jest pakiet ISaGRAF) zasobu **BinaryFile** lub **TextFile**. Znaki zastosowane do oddzielenia komponentów nazwy ścieżki (napęd, katalog, prefiks, sufiks) musza być zgodne z konwencjami systemu MS-DOS.

Składnia: **FROM** '<ścieżka docelowa>'

Przykład:
```
BinaryFile 'MYFILE'
  Begin
     ...
     From 'c:\user\config.dat'
     To '/dd/user/appl/config.dat'
  End
```

TO

Znaczenie: Określa nazwę ścieżki docelowej (w systemie oprogramowania wbudowanego) zasobu **BinaryFile** lub **TextFile**. Znaki zastosowane do oddzielenia komponentów nazwy ścieżki (napęd, katalog, prefiks, sufiks) musza być zgodne z konwencjami systemu MS-DOS.

Składnia: **TO** '<ścieżka docelowa>'

Przykład:
```
TextFile 'MYFILE'
  Begin
     ...
     From 'c:\user\config.dat'
     To '/dd/user/appl/config.dat'
  End
```

- Przykład

Poniżej przedstawiono przykład pełnego pliku definicji zasobów:

```
(* plik definicji zasobów *)

ULongData 'DATA1'          (* lista wartości *)
Begin
   Target 'ISA86M'         (*  tylko  dla  tego  oprogramowania
wbudowanego *)
   1, 0, 16#1A2B3C4D, +1, -1   (* wartości liczbowe *)
End

VarList 'VLIST1'           (* lista zmiennych *)
Begin
   Target 'ISA86M'         (*  tylko  dla  tego  oprogramowania
wbudowanego *)
   Valve1, StateX, Command, Alrm1    (* nazwy zmiennych *)
End

BinaryFile 'FILE1'         (* zasoby w pliku binarnym *)
Begin
   AnyTarget               (*     dedykowane     dla     każdego
oprogramowania wbudowanego *)
   From 'c:\user\updatef.bin'  (* plik źródłowy na komputerze
PC *)
   To 'updatef.cfg'        (* plik docelowy w PLC *)
End

TextFile 'FILE2'           (* zasoby w pliku tekstowym *)
Begin
   Target 'ISA68M'
   From 'c:\nw\nwbd.txt'   (*  plik  źródłowy  na  komputerze  PC
*)
   To '/nw/dat/nwbd'       (* plik docelowy w PLC *)
End
```

- Kompilowanie zasobów

Kiedy zasoby zostaną umieszczone w pliku definicji zasobów, pod koniec generowania kodu ISaGRAF pojawia się okno dialogowe. Naciśnięcie przycisku "**Rozpocznij kompilację**" uruchamia kompilator zasobów. Komunikaty i błędy wyjściowe zostaną wyświetlone w głównym oknie kontrolnym. Naciskając przycisk "**Wyjście**" unika się kompilacji zasobów. W takim przypadku zasoby nie zostaną dodane do kodu ISaGRAF.

- Implementacja

Ilość zasobów, rozmiar rzędów danych i plików nie są ograniczane przez ISaGRAF. Zasoby są przechowywane na końcu wygenerowanego kodu wraz z katalogiem zasobów. Poniżej podano (stosując notację C) format katalogu zasobów:

```
__RESOURCE:
{
```

```
  long nbres;             /* ilość zdefiniowanych zasobów */
  {
    char name[16];        /* nazwa zasobu */
    long type;            /* typ danych zasobu */
    long size;            /* dokładny rozmiar bloku danych */
    void *data;
    char *path_offset;    /* wskazuje na ciąg */
  } /*il rekordów */
}
```

Poniżej znajdują się możliwe wartości pola typu („type”):
1 = plik binarny
2 = plik tekstowy
3 = dane „długie” (pole path_offset nie jest w takim przypadku używane)
4 = lista zmiennych (pole path_offset nie jest w takim przypadku używane)

W przypadku plików tekstowych znaki końca linii zostają przełożone przez kompilator zasobów, zgodnie z konwencjami systemu oprogramowania wbudowanego. Wszystkie wskaźniki to 32-bitowe przesunięcia adresów odpowiadających im struktur. Wszystkie nazwy i ścieżki zasobów są ciągami zakończonymi znakiem NULL (pustym). Nazwy ścieżki i dane są zgodne z katalogiem zasobów.

A.14 Odsyłacze

Pakiet ISaGRAF zawiera edytor odsyłaczy, który zapewnia użytkownikowi globalny podgląd wszystkich zmiennych zadeklarowanych w programach projektu oraz miejsc, w których zostały one użyte. Celem odsyłacza jest sporządzenie listy wszystkich zmiennych zadeklarowanych w projekcie oraz lokalizacja, w kodach źródłowych każdego programu, tych części kodu źródłowego, w których zmienne te zostały użyte. Odsyłacze są bardzo przydatne do globalnego obrazu cyklu życia jednej zmiennej. Pomagają one zlokalizować efekty uboczne i zredukować czas zapoznawania się z projektem podczas jego konserwacji. Odsyłacze mogą być również użyte do uzyskania globalnego obrazu całości słownika projektu, aby łatwo można było odnaleźć nieużywane zmienne oraz ocenić stopień złożoności projektu.
Lista po lewej stronie przedstawia zadeklarowane obiekty projektu (programy, zmienne i słowa zdefiniowane) oraz elementy biblioteki (funkcje i bloki funkcyjne), do których odwołuje się projekt. Lista po prawej stronie przedstawia przypadki wystąpienia w programach obiektu wybranego aktualnie z pierwszej listy.
Opis wystąpienia zawiera nazwę programu, numer etapu FC lub SFC, przejście lub test oraz dodatkowo numer linii dla języków tekstowych lub współrzędne dla schematów LD czy FBD. W przypadku schematów w LD, opis uzupełniany jest numerem szczebla. Jeżeli zmienna jest używana jako wyjście (na cewce), po numerze szczebla występuje znak gwiazdki ("*****").
Aby wyświetlić również zmienne z listy głównej, które nie są używane w programach aplikacji, należy włączyć opcje "**Pokaż nieużywane zmienne**" z menu "**Opcje**".

Wybór typu obiektu

Ze względu na fakt, iż projekt może grupować olbrzymią liczbę zadeklarowanych obiektów, pole zbiorcze na pasku narzędzi jest stosowane do wyboru typu obiektów, które mają zostać wymienione na liście w oknie. Pozwala to użytkownikowi na uzyskanie dostępu do wybranych informacji.

Za każdym razem, gdy odsyłacze są ponownie wyliczane, wybierana jest opcja "**Wszystkie obiekty**", aby przedstawić kompletną ich listę.

Ponowne wyliczanie odsyłaczy

Komenda "**Plik / Przelicz**" może być zastosowana w dowolnym momencie do zaktualizowania odsyłacze zgodnie z modyfikacjami wprowadzonymi w innych oknach edycyjnych ISaGRAF.

Eksportowanie odsyłaczy

Komenda "**Narzędzia / Eksportuj**" używana jest do zapisania kompletnej listy odsyłaczy w pliku tekstowym w formacie ASCII. Plik ten może być następnie otwierany przy pomocy innych aplikacji, takich jak Notatnik Windows czy edytory tekstu.

Błędy słownika

Komenda "**Edycja / Błędy słownika**" wyświetla okno dialogowe z listą błędów wykrytych podczas ładowania słownika projektu.

Statystyka

Komenda "**Narzędzia / Statystyki**" wyświetla okno dialogowe z ilością obiektów i zmiennych zadeklarowanych w projekcie według typów i atrybutów zmiennych. Szczególnym zastosowaniem tej komendy jest odnalezienie ilości zmiennych We/Wy zadeklarowanych w projekcie w celu upewnienia się, czy może on zostać skompilowany, jeżeli używana jest ograniczona wersja pakietu ISaGRAF.

Wyszukiwanie na liście obiektów

Komenda "**Edycja / Znajdź**" pozwala użytkownikowi bezpośrednio wybrać obiekt na liście edytora. Poszukiwany obiekt nie może zostać znaleziony, jeśli nie znajduje się na liście (przy zastosowaniu wybranego rodzaju wyświetlania). Zaleca się, aby przed przystąpieniem do wyszukiwania obiektu wybrać na pasku narzędzi opcję "**Wszystkie**".

Otwieranie programu

Lista po prawej zawiera wystąpienia wybranego obiektu w plikach źródłowych oraz połączeniach We/Wy otwartego projektu. Komenda "**Edycja / Otwórz program**" umożliwia użytkownikowi bezpośrednie otwarcie programu, w którym występuje dany obiekt. Aby otworzyć program można dwukrotnie kliknąć myszą miejsce jego wystąpienia (na liście przypadków wystąpienia).

A.15 Korzystanie z debagera graficznego

Pakiet ISaGRAF zawiera kompletny debager graficzny i symboliczny. Komenda "Debaguj" w oknie ‘Programy’ uruchamia debager kontrolujący aplikację załadowaną do sterownika PLC. W tym trybie debager komunikuje się z systemem oprogramowania sterownika za pośrednictwem fizycznego łącza. Komenda "Symuluj" w oknie ‘Programy’ uruchamia jednocześnie debager i kompletny symulator sterownika. Pozwala to użytkownikowi przetestować aplikację, kiedy system sterownika z wejściami i wyjściami nie jest jeszcze ukończony. Okno debagera zawiera komendy kontrolujące całość aplikacji.

Jeśli debager zostaje uruchomiony i jeśli aplikacja w sterowniku PLC jest taka sama jak w pakiecie ISaGRAF, **okno Programy** automatycznie zostaje otwarte w trybie debagowania. Komendy w tym oknie mogą być używane do otwierania innych okien ISaGRAF (edytorów graficznych i tekstowych, słownika, list zmiennych, konfiguracji We/Wy, itd.). Wszystkie okna otwierane w sesji debagera działają w "**trybie debagowania**", tzn. komenda edycji jest niedostępna. Wyświetlane składniki programu (kroki, przejścia, zmienne, itd.) są pokazane wraz z ich aktualną wartością lub statusem. Dwukrotne kliknięcie obiektu zmienia jego status lub wartość w aplikacji sterownika.

Przy uruchomieniu debagera w **trybie symulacji**, komunikacja z oprogramowaniem wbudowanym ISaGRAF sterownika zostaje zatrzymana. Debager komunikuje się jedynie z oknem symulatora. Ze względu na fakt, iż nie ma sterownika (w tym trybie), komendy "**załaduj**", "**zatrzymaj**" czy "**uruchom**" w menu debagera nie są dostępne.

A.15.1 Okno debagera

Okno debagera zawiera jedynie informacje dotyczące generalnego statusu aplikacji. Jest ono połączone z innymi oknami ISaGRAF, tworząc kompletny, interaktywny system debagowania. Wykryte błędy wykonania wyświetlane są w dolnej części okna debagera. Komendy z menu "**Opcje**" służą do ukrywania, pokazywania lub czyszczenia listy błędów.

Panel sterowania (obszar poniżej menu debagera) pokazuje generalny status aplikacji sterownika oraz informacje na temat parametrów czasowych cyklu wykonywania. Lista możliwych statusów sterownika przedstawia się następująco:

Łączenie:..... Debager ustanawia połączenie ze sterownikiem.
Rozłączony: ..... Debager nie może połączyć się ze sterownikiem. Upewnij się, czy przewód połączeniowy i parametry komunikacji są prawidłowe.

Brak aplikacji: ..... Połączenie jest dobre, lecz w sterowniku ISaGRAF nie znajduje się aktualnie żadna aplikacja. Załaduj aplikację.
Aplikacja aktywna: ..... Połączenie jest dobre i w sterowniku znajduje się aplikacja. Debager ustanawia teraz połączenie z tą aplikacją, jeśli jest ona taka sama jak ta, która znajduj się w pakiecie ISaGRAF.
URUCHOMIONY: ..... Aplikacja sterownika pracuje w trybie rzeczywistym.
ZATRZYMANY: ..... Aplikacja sterownika pracuje w trybie "Cykl po cyklu".
Pułapka: ..... Aplikacja sterownika pracuje w trybie " Cykl po cyklu" i napotkała na pułapkę.
Błąd krytyczny: ..... Aplikacja sterownika uległa awarii z powodu wystąpienia poważnego błędu.

Informacja dotycząca parametrów czasowych wykonywanego cyklu przedstawia się następująco:
Dopuszczalny: ..... zaprogramowane parametry czasowe.
Bieżący: ..... dokładne parametry czasowe ostatniego pełnego wykonywanego cyklu.
Maksymalny: ..... maksymalne parametry czasowe wykryte od momentu uruchomienia aplikacji.
Przekroczenie: ..... ilość wykonanych cykli o parametrach czasowych większych od dopuszczalnych.

Wszystkie wartości czasowe podane są w milisekundach. Wartości czasowe nie są wyświetlane, kiedy debager jest używany w trybie symulacji.

A.15.2 Kontrolowanie pracy aplikacji

Menu "**Plik**" oraz "**Sterowanie**" zawierają komendy instalacji i sterowania aplikacji ISaGRAF aktualnie przetwarzanej w oprogramowaniu sterownika ISaGRAF.

Uwaga: Niektóre z tych komend nie są dostępne podczas symulacji, ponieważ symulator z pakietu ISaGRAF wykona je automatycznie dla przetwarzanej aplikacji.

Zatrzymywanie aplikacji sterownika

Komenda "**Plik / Zatrzymaj aplikację**" zatrzymuje wykonywanie aktywnej aplikacji w sterowniku ISaGRAF.

Uruchomianie aplikacji w sterowniku

Komenda "**Plik / Uruchom aplikację**" uruchamia aplikację znajdującą się w sterowniku. Po załadowaniu aplikacja zostaje automatycznie uruchomiona, nie trzeba więc używać

komendy "**Uruchom**" w tym wypadku. Komenda " **Uruchom** " jest zazwyczaj używana po komendzie "**Zatrzymaj**".

Uwaga: załadowanie nowej aplikacji możliwe jest tylko wtedy, gdy aplikacja sterownika została zatrzymana (jest nieaktywna).

Załadowanie aplikacji

Komenda "**Plik / Załaduj**" jest stosowana do załadowania kodu aplikacji do systemu oprogramowania wbudowanego. Wybierz do załadowania typ kodu zgodny z procesorem systemu oprogramowania wbudowanego i opcjami aplikacji.

Podawanie numeru wersji

Komenda "**Plik / Podaj numer wersji**" jest stosowana do wyświetlenia kompletnych danych identyfikacyjnych aplikacji otwartej w pakiecie ISaGRAF i aplikacji w sterowniku. Przez aplikację sterownika rozumie się aplikację znajdującą się aktualnie w sterowniku, a wykonywaną pod kontrolą oprogramowania sterownika ISaGRAF. Wyświetlane są następujące informacje:

WERSJA:..... Jest to numer wersji kodu aplikacji. Numer ten został wyliczony przez generator kodu.
DATA: ..... Ten element pokazuje datę i czas budowy kodu.
CRC: ..... Jest to suma kontrolna wyliczona z zawartości tablicy symboli. Numer ten jest obliczany przez generator kodu. Wartość ta uzależniona jest od zawartości słownika zmiennych.

Uwaga: Komenda "**Pokaż numer wersji**" jest również dostępna podczas symulacji. W trybie rzeczywistym debagowania komendy tej nie można używać, jeśli nie jest podłączony sterownik PLC z oprogramowaniem wbudowanym.

Modyfikacja natychmiastowa

Komenda "**Plik / Uaktualnij aplikację**" pozwala użytkownikowi wprowadzać "modyfikacje natychmiastowe" działającej aplikacji sterownika. Komenda ta jest szczegółowo opisana w dalszych sekcjach tego rozdziału. Nie jest ona dostępna gdy debager pracuje w trybie symulacji.

Tryb czasu rzeczywistego

Komenda "**Sterowanie / Czas rzeczywisty**" nie jest dostępna, kiedy żadna z aplikacji nie jest aktywna. Przełącza ona aplikację sterownika w normalny tryb pracy - tryb "czasu rzeczywistego": W trybie normalnym cykle wyzwalane są zgodnie z zaprogramowanymi parametrami czasowymi cyklu.

Tryb „cykl po cyklu”

Komenda "**Sterowanie / Cykl po cyklu**" nie jest dostępna, kiedy żadna z aplikacji nie jest aktywna. Przełącza ona aplikację sterownika w tryb "cykl po cyklu ": W trybie tym cykle są wykonywane jeden za drugim, zgodnie z komendami "**Wykonaj jeden cykl**", wydawanymi przez użytkownika z menu debagera.

Wykonywanie jednego cyklu

Gdy sterownik działa w trybie „cykl po cyklu”, komenda "**Sterowanie / Wykonaj jeden cykl**" uruchamia wykonanie jednego cyklu.

Parametry czasowe cyklu

Komenda "**Sterowanie / Zmień czas cyklu**" umożliwia użytkownikowi modyfikację zaprogramowanych parametrów czasowych cyklu. Parametry takie są oznaczone jako "**Dopuszczalny**" w oknie paska sterowania debagera. Tryb "**Cykl po cyklu**" powinien zostać włączony przed modyfikacją parametrów czasowych cyklu. Czas cyklu jest liczbą typu integer. Określa czas w milisekundach.

Kasowanie wszystkich pułapek

Komenda "**Sterowanie / Skasuj wszystkie pułapki**" usuwa wszystkie aktualnie zainstalowane w całej aplikacji pułapki (napotkane lub nadal aktywne).
Ustawione pułapki nie są automatyczne usuwane przy zamknięciu okna debagera.

Odblokowanie zmiennych We/Wy

Komenda "**Sterowanie / Odblokuj wszystkie zmienne We/Wy**" odblokowuje wszystkie zmienne We/Wy aktualnie zablokowane w aplikacji. Kiedy zmienna We/Wy jest zablokowana, nie dokonywane są żadne zmiany statusu wejścia ani wyjścia odpowiadającego im urządzenia We/Wy. Zmienne skojarzone z We/Wy mogą być nadal zapisywane przez aplikację bądź debager. Zablokowane aktualnie zmienne We/Wy nie są odblokowywane automatycznie, kiedy zamykane jest okno debagera.

A.15.3 Opcje

Menu "**Opcje**" zawiera opcje sterujące informacjami wyświetlanymi w oknie debagera.

Parametry komunikacyjne

Parametry czasowe transmisji mogą być ustawiane, gdy aktywne jest okno debagera. Można tu ustawić jedynie **limit czasu** dla komunikacji. Inne parametry komunikacyjne (jak prędkość transmisji, parzystość) należy ustawiać z menu "**Debaguj**" okna Programy.
"**Limit czasu"** komunikacji to czas, w którym system sterownika powinien rozpocząć odpowiadanie na żądanie wysłane z pakietu ISaGRAF. "**Okres odświeżania**" to okres czasu, co który debager ma wysyłać żądania "**czytaj**" w celu odświeżenia danych w otwartych oknach.
Wszystkie wartości czasowe są wyświetlane i wprowadzane jako liczby typu integer i wyrażone w **milisekundach**. Parametry czasowe komunikacji nie mogą być ustawiane, gdy debager pracuje w trybie symulacji.

Opcje wyświetlania

Opcja "**Pokaż czas cyklu**" umożliwia użytkownikowi ukrycie lub pokazanie wartości **czasu cyklu** na pasku sterowania debagera. Kiedy opcja ta jest ustawiona, wyświetlane i odświeżane są wszystkie składniki parametrów czasowych cyklu (dopuszczalny, bieżący, maksymalny, przekroczenia). Wyłączenie tej opcji obniża obciążenie transmisji.
Kiedy ustawiona jest opcja "**Pokaż błędy**", wykryte błędy wykonania zostaną pokazane w dolnej części okna debagera. Kiedy opcja ta jest wyłączona, pole listy błędów jest zamknięte. Wyłączenie tej opcji obniża obciążenie debagera transmisją i wyświetlaniem. Komenda "**Opcje / Skasuj błędy**" czyści listę błędów wykonania wyświetlaną aktualnie w oknie debagera.
Komenda "**Opcje / Minimalizuj okno**" zmniejsza okno debagera tak, iż jest ono widoczne jako niewielki panel, znajdujący się zawsze na wierzchu, zawierający jedynie status aplikacji oraz przyciski graficzne najczęściej używanych komend.

A.15.4 Komendy „Wpisz”

Debager symboli ISaGRAF oferuje wiele komend pozwalających zmienić **wartość** lub **status** składników aplikacji. Wyboru składnika, który ma zostać zmieniony dokonuje się przez **dwukrotne kliknięcie** jego nazwy lub rysunku w oknie edycyjnym, kiedy otwarte jest okno debagera.

Zmienne

Status zmiennej zmienia się przez dwukrotne kliknięcie jej nazwy w jednym z poniższych okien:
- Słownik
- Listy zmiennych lub wykresy czasowe
- Programy LD lub FBD
- Konfigurator We/Wy

W oknie dialogowym debagowania dostępne są następujące funkcje:

- Zapisywanie nowej wartości zmiennej
- **Blokowanie** zmiennej (tylko zmienne We/Wy)
- **Odblokowanie** zmiennej (tylko zablokowane zmienne We/Wy)
- **Uruchomienie** lub **zatrzymanie** zmiennej typu timer (ustaw tryb odświeżania automatycznego)

Wartości symboli stosowane do reprezentowania wartości binarnych **FAŁSZ** i **PRAWDA** to ciągi tekstowe przyporządkowane tej konkretnej zmiennej binarnej w słowniku. Wartość analogowa określana w komendzie "**Modyfikuj**" musi być wprowadzona w formacie integer lub real, zgodnie z definicją zmiennej w słowniku. Ciąg tekstowy określany w komendzie "**Modyfikuj**" dla komunikatu nie może być dłuższy niż wielkość komunikatu, przypisana do tej konkretnej zmiennej w słowniku.

Obiekty SFC

W celu obserwacji operacji sterowania pracą **programu SFC** podczas debagowania aplikacji, w oknie Program stosowane są komendy menu "**Plik**". Program SFC musi zostać wybrany z listy programów. Dostępne są następujące komendy:

Uruchom program SFC:..... Włącza wybrany program przez wstawienie znacznika do każdego z jego kroków początkowych.
Zakończ program SFC: ..... Zamyka wybrany program przez usunięcie wszystkich istniejących znaczników.
Zamroź program SFC:..... Usuwa wszystkie istniejące znaczniki wybranych programów i zapamiętuje ich położenie.
Uruchom ponownie p. SFC:... Ponownie uruchamia wstrzymany program poprzez ponowne wstawienie znaczników, które zostały usunięte komendą "Wstrzymaj".

W przypadku programów potomnych, komendy te odpowiadają funkcjom "**GSTART**", "**GKILL**", "**GFREEZE**" oraz "**GRST**" w języku programowania.

Operację sterowania można zobaczyć w **kroku SFC** podczas debagowania aplikacji przez dwukrotne kliknięcie jej reprezentacji graficznej w oknie edycji SFC. W oknie dialogowym debagera dostępne są następujące komendy:
- Instaluj pułapkę **aktywizacji** kroku
- Instaluj pułapkę **dezaktywizacji** kroku
- **Usuń** pułapkę dodaną do kroku

Uwaga: Pułapki aktywizacji i dezaktywizacji nie mogą być dodawane do tego samego kroku.

Operację sterowania można zobaczyć w **przejściu SFC** podczas debagowania aplikacji przez dwukrotne kliknięcie jej reprezentacji graficznej w oknie edycji SFC. W oknie dialogowym debagera proponowane są następujące komendy:
- Dodaj **pułapkę** do czyszczenia przejścia
- **Usuń** pułapkę dodaną do przejścia
- Ręcznie **usuń** przejście (przesuń lub dodaj znaczniki)

Kasowanie warunkowe: w krokach następujących po przejściu tworzony jest znacznik. Znaczniki istniejące w poprzednich krokach zostają usunięte. **Kasowanie bezwarunkowe**: w krokach następujących po przejściu tworzony jest znacznik. Znaczniki istniejące w poprzednich krokach nie zostają usunięte.

A.15.5 Modyfikacja natychmiastowa

Funkcja "Modyfikacja natychmiastowa" pozwala użytkownikowi na modyfikowanie aplikacji w czasie pracy procesu. Czasami jest to konieczne w procesach chemicznych, których przerwanie mogłoby zagrozić produkcji lub bezpieczeństwu. Funkcji tej należy używać **bardzo ostrożnie**. ISaGRAF może nie być w stanie wykryć wszystkich możliwych konfliktów powodowanych operacjami wykonywanymi przez użytkowników w wyniku wprowadzania takich zmian.

- ***Sekwencje kodów***

Ponieważ ISaGRAF oferuje wiele możliwości dostępu do zmiennych, programów czy kart We/Wy z poziomu debagera, to funkcja "modyfikacji natychmiastowej" odnosi się jedynie do modyfikacji sekwencji kodu. Sekwencja kodu, to pełny zestaw instrukcji ST, IL, LD lub FBD wykonywanych po kolei. W programie "rozpoczynającym cykl" lub "kończącym cykl" sekwencja kodu jest całą listą instrukcji wpisanych w program. W programie SFC sekwencją kodu jest programowanie jednego kroku lub przejścia na poziomie 2. "Modyfikacja natychmiastowa" polega na zamianie jednej lub kilku sekwencji kodu, bez zatrzymywania wykonywanego cyklu PLC. Jako, że kontrola znaczników SFC jest wyjątkowo istotna, **nie jest możliwa modyfikacja struktury SFC, dodawanie, ponowne numerowanie czy usuwanie kroku, przejścia lub programu SFC**.

- ***Zmienne***

Ponieważ baza danych dotyczących zmiennych stanowi niesłychanie istotną część aplikacji, można do niej uzyskać dostęp w dowolnym momencie przez inne procesy (w wielozadaniowym PLC). Możliwa jest także modyfikacja wartości zmiennych z debagera. Dlatego też **ISaGRAF nie pozwala użytkownikowi na natychmiastowe dodawanie, zmienianie nazwy czy usuwanie zmiennej**. Tym niemniej możliwa jest zmiana sposobu, w jaki zmienna jest używana w aplikacji. Możliwe jest również zarezerwowanie "wolnych" zmiennych wewnętrznych lub We/Wy w pierwszej wersji aplikacji tak, aby można je było wykorzystać w przyszłych modyfikacjach.

W bazie danych sterownika ISaGRAF istnieje kilka rodzajów zmiennych. Ograniczenia odnoszą się do każdej z nich:

- Zmienne deklarowane

Są one deklarowane przy użyciu słownika ISaGRAF. Nie mogą być zmieniane ani nie może być zmieniana ich nazwa w sposób natychmiastowy. Zaleca się zadeklarowanie i inicjalizację w aplikacji kilku zmiennych dodatkowych, nawet jeśli nie będą one używane od razu. Takie dodatkowe zmienne pozwolą na

wprowadzanie w przyszłości modyfikacji, z którymi można będzie pracować bez zmiany sumy kontrolnej aplikacji.

- Instancje bloków funkcyjnych

Każda instancja napisanego bloku funkcyjnego "C" lub IEC odpowiada danym przechowywanym w bazie danych czasu rzeczywistego sterownika ISaGRAF. Kiedy instancje bloków funkcyjnych zostaną dodane lub usunięte, nie jest już możliwa modyfikacja natychmiastowa. Tak więc lepiej jest pracować w ST z instancjami FB zadeklarowanymi w słowniku, niż dodawać bloki (które będą odpowiadać nowym, deklarowanym automatycznie instancjom) w schematach FBD czy LD. Ponadto wszelkie modyfikacje definicji dostępnych bloków funkcyjnych w bibliotece ISaGRAF prowadzić będą do uniemożliwienia zmian natychmiastowych.

- Kroki

Każdy krok SFC odpowiada elementowi danych, w którym przechowywane są atrybutu dynamiczne kroku SFC (czas jego działania i flaga). Dodanie lub usunięcie kroków SFC zmienia bazę danych aplikacji i niedozwolona jest ich zmiana natychmiastowa.

- Zmienne ukryte przydzielane przez kompilator

Kompilator ISaGRAF generuje tymczasowe zmienne "ukryte", służące do rozwiązywania wyrażeń złożonych. W niektórych przypadkach zmiana wyrażenia może prowadzić do stworzenia innego zestawu niewidocznych zmiennych tymczasowych, co powoduje uniemożliwienie wprowadzania zmian natychmiastowych. Aby uniknąć takiej sytuacji można dodać poniższe zapisy do pliku ISA.INI, wymuszając minimalną ilość zmiennych tymczasowych, które może przydzielić każdemu programowi nawet, jeśli nie są one wykorzystywane w kompilacji pierwszej wersji aplikacji. Podane tutaj wartości są przykładowe:

```
[DEBUG]
MNTVboo=8       ; dla wartości binarnych
MNTVana=4       ; dla wartości typu integer i real
MNTVtmr=4       ; dla wartości typu timer
MNTVmsg=2       ; dla komunikatów
```

Kiedy ustawienia tego typu zostaną zapisane w pliku ISA.INI, kompilator generuje komunikat ostrzegawczy jeśli nowa kompilacja aplikacji prowadzi do powstania większej liczby przydzielonych zmiennych tymczasowych.

Wejścia i wyjścia

Ponieważ system We/Wy ISaGRAF jest bardzo otwarty, wymagane modyfikacje powinny być implementowane przez producenta sprzętu przy użyciu specyficznych funkcji odpowiedniego sprzętu. System ISaGRAF **nie pozwala użytkownikowi na dodawanie, łączenie czy usuwanie zmiennej We/Wy, ani na modyfikację opisu karty We/Wy** w trybie natychmiastowym. Operacje takie, jak modyfikacja parametrów karty i blokowanie kanałów We/Wy są dostępne przy użyciu standardowych funkcji OEM oraz funkcji "**OPERATE**".

Wykonywane operacje

Modyfikowanie działającej aplikacji polega na wykonaniu następujących operacji:
- modyfikacja kodu źródłowego aplikacji w pakiecie ISaGRAF
- wygenerowanie nowego kodu aplikacji
- załadowanie nowego kodu aplikacji przy użyciu komendy "**uaktualnij**" zamiast "**załaduj**"
- przełączenie się ze starej aplikacji na nową pomiędzy wykonywanymi cyklami PLC przy użyciu komendy "**Wykonaj uaktualnienie**".

Procedura ta gwarantuje, iż sterownik PLC będzie zawsze posiadał pełną i niezawodnie działającą aplikację oraz będzie umożliwiał użytkownikowi kontrolowanie parametrów czasowych operacji w bardzo bezpieczny i wydajny sposób. Pozwala ona również użytkownikowi na modyfikowanie projektu tak często, jak jest to możliwe. Niezależnie od samego procesu "modyfikacja natychmiastowa" w zasadzie nie różni się od normalnego zestawu komend "**zatrzymaj**, **uruchom** i **załaduj**". Jedyne różnice polegają na tym, że nie traci się żadnego stanu zmiennej, a czas przełączenia jest bardzo krótki (trwa zazwyczaj 1 lub 2 cykle). Podczas przełączenia nie jest modyfikowana żadna zmienna i **wszystkie zmienne wewnętrzne, wejściowe czy wyjściowe utrzymują tę samą wartość** przed i po modyfikacji aplikacji. Podczas przełączenia nie jest wykonywane żadne działanie, a **znaczniki SFC nie zostają przemieszczone**.

Wymagana pamięć

Aby możliwa była obsługa funkcji "Modyfikacja natychmiastowa", sterownik PLC musi posiadać wolny obszar pamięci, umożliwiający przechowywanie zmodyfikowanej wersji kodu aplikacji. Obie wersje kodu aplikacji muszą być przechowywane w pamięci PLC podczas operacji przełączenia.

Ograniczenia

Jak opisano wcześniej, dopuszczalne są jedynie modyfikacje sekwencji kodu. Definicje zmiennych, parametry aplikacji oraz konfiguracja We/Wy nie mogą być zmieniane. Przy załadowywaniu zmodyfikowanej wersji aplikacji, ISaGRAF porównuje zmodyfikowaną aplikację z aplikacją działającą, aby wykryć wszelkie niebezpieczne zmiany. Jeśli zmiana wydaje się niebezpieczna lub niemożliwa, generowany jest błąd załadowania. Jednym z zabezpieczeń stosowanych przez ISaGRAF jest porównanie sumy kontrolnej tablicy symboli, co ma na celu wykrycie każdej zmiany nazwy zmiennej, programu czy elementu SFC. Jeśli dany krok jest aktywny w momencie wystąpienia przełączenia, jego nie zapisywane (N) działania zostają utracone. Nie są wykonywane działania aktywizujące nowy krok. Działania wykonywane przy deaktywowaniu kroku zostają przejęte przez nowy kod aplikacji. Jeśli w momencie przełączenia występuje poprawne przejście, jego równanie odbiorcze zostaje uaktualnione. W PLC nie jest wykonywana kopia zapasowa nowego załadowanego kodu aplikacji. Kopię zapasową stanowi wersja, którą załadowano poprzednio przy pomocy standardowych komend.

Operacje

Aby uaktualnić kod działającej aplikacji należy wykonać następujące operacje:

- Przed wykonaniem jakiejkolwiek zmiany w działającej aplikacji, wysoce zalecane jest sporządzenie kopii aktualnego projektu pod inną nazwą. Modyfikacje można wykonywać na kopiach.
- Przed edycją jakiegokolwiek programu użytkownik powinien sprawdzić, czy opcja "**uaktualnij dziennik**" w narzędziach edycyjnych jest włączona, w celu ułatwienia późniejszej konserwacji programu.
- Kiedy zmodyfikowano jedną lub kilka sekwencji (bez zmiany struktur SFC i hierarchii programu), kod nowej aplikacji musi zostać wygenerowany w pakiecie ISaGRAF przed jego załadowaniem.
- Używając debagera pod starym projektem, użytkownik musi przyłączyć sterownik PLC i wykonać dowolną operację, co może sprawić, iż aplikacja uaktualni się szybciej i bezpieczniej.
- Używając debagera pod nowym projektem, użytkownik musi przyłączyć sterownik PLC. Jeśli nazwa aplikacji zmieni się, nie można będzie uzyskać dostępu do bazy danych sterownika. Użytkownik musi wybrać komendę "**Plik / Uaktualnij**".
- Zmodyfikowana aplikacja jest ładowana przez wybranie opcji "**uaktualnij później**". Może to spowolnić nieco pracę PLC podczas przesyłania.
- Kiedy ładowanie zostanie ukończone, użytkownik może uruchomić komendę "**Plik / Wykonaj uaktualnienie**", aby umożliwić przełączenie w najodpowiedniejszym momencie. Przełączenie będzie trwało 1-2 cykle.
- Po prawidłowym wykonaniu przełączenia, wyświetlane są programy zmodyfikowanej, działającej aplikacji. Jeśli tak się nie dzieje, istniejąca działająca aplikacja pozostaje bez zmiany.

A.15.6 Mechanizm wymiany DDE

Debager ISaGRAF zawiera serwer DDE (Dynamic Data Exchange - Dynamicznej Wymiany Danych). Możliwe jest zainstalowanie pętli zawiadamiania pomiędzy debagerem ISaGRAF i innymi aplikacjami w celu dynamicznego wyświetlania aktualnej wartości zmiennych w aplikacjach nie należących do oprogramowania ISaGRAF.

Jedynie transakcje "zawiadamiania" (ang. advised) oraz "wstawiania" (ang. poke) są obsługiwane przez serwer DDE debagera ISaGRAF. Transakcji „żądanie” można używać tylko dla zmiennych śledzonych w pętli zawiadamiania. Inne usługi DDE, takie jak "wykonaj" nie są dostępne. Gdy w pętli zawiadamiania znajduje się zmienna, to jej wartość jest uaktualniona w aplikacji-kliencie za każdym razem, kiedy zmienna ulegnie zmianie. Śledzone mogą być zmienne dowolnego typu. Identyfikacja łącza dynamicznego obejmuje następujące nazwy:

Nazwa usługi:	"ISaGRAF"
Nazwa tematu:	Nazwa projektu ISaGRAF
Nazwa elementu:	Nazwa zmiennej

Jeśli zmienna jest zmienną lokalną programu, to po jej nazwie musi występować nazwa programu-rodzica zapisana w nawiasach i z użyciem następującej składni:

nazwa_zmiennej(nazwa_programu)

Serwer DDE debagera ISaGRAF jest dedykowany aplikacji ISaGRAF śledzonej aktualnie przez debager. Serwer ISaGRAF może śledzić do **256** zmiennych. Serwer DDE może być używany zarówno, gdy debager ISaGRAF działa w trybie ze sterownikiem, jak i w trybie symulacji. Czas odświeżania jest taki, jak ustalony do komunikacji pomiędzy debagerem a systemem sterownika ISaGRAF lub symulatorem.

A.16 Zmienne śledzone

Komenda "**Listy śledzeń**" w menu "**Śledzenie**" okna Debagera pozwala użytkownikowi na budowanie nieciągłych list zmiennych, których wartości odświeżane są na bieżąco. Listy budowane są podczas debagowania aplikacji. Mogą one być przechowywane na dysku i otwierane ponownie podczas innych sesji debagera. Lista może zawierać do **32** zmiennych. Zmienne różnych typów można umieszczać na tej samej liście. Do listy można dodawać zmienne globalne oraz lokalne. Lista zmiennych jest przyporządkowana jednemu konkretnemu projektowi. Listy zmiennych są wielce użyteczne do funkcjonalnego testowania aplikacji. Pozwalają one użytkownikowi obserwować zmiany wyznaczonej części nadzorowanego procesu, niezależnie od odpowiadającego mu kodu źródłowego programów aplikacji. Listy zmiennych są również przydatne przy debagowaniu programów tekstowych ST i IL. Użytkownik może z łatwością pogrupować na liście zestaw zmiennych wykorzystywanych przez program, w celu nadzorowania lub monitorowania wykonywania zaprogramowanych instrukcji.
ISaGRAF wyświetla nazwę, bieżącą wartość i tekst komentarza dla każdej zmiennej na liście. Rozmiar kolumn może być zmieniany poprzez przeciągnięcie myszą oddzielającej je linii na pasku tytułowym listy.

Zapisywanie list na twardym dysku

Komendy menu "**Plik**" używane są do tworzenia, otwierania i zapisywania list zmiennych. Ilość list dla jednego projektu nie jest ograniczana przez ISaGRAF. Przy nadawaniu nazw listom zmiennych zapisywanym na dysku, należy przestrzegać poniższych reguł:

- nazwa nie może być dłuższa niż **8** znaków
- pierwszym znakiem musi być **litera**
- kolejne znaki mogą być **literami**, **cyframi** lub znakiem podkreślenia
- nadawanie nazw listom nie uwzględnia wielkości liter

Edytor list nie może wyświetlić więcej niż jedną listę zmiennych naraz w jednym oknie. Jednakże edytor list można uruchomić więcej niż jeden raz w celu równoczesnego śledzenia różnych list.

Wstawianie zmiennych na listę

Komenda "**Edycja / Wstaw**" wstawia kolejną zmienną na listę. Nazwa zmiennej wybierana jest na liście obiektów zdefiniowanych w słowniku projektu. W ten sposób użytkownik nie musi wprowadzać identyfikatora ręcznie. Zmienna jest wstawiana przed zmienną aktualnie wybraną na liście. Lista nie może zawierać więcej niż **32** zmiennych. Ta sama zmienna nie może pojawiać się więcej niż jeden raz na tej samej liście.

Zmiana wybranej zmiennej

Komenda "**Edycja / Modyfikuj**" zmienia wybraną zmienną na inną zmienną. Można również wykorzystać komendę "**Wytnij**", aby usunąć wybraną zmienną z listy.

Wyświetlanie zrzutu

W dowolnym momencie można zmienić tryb widoku na "Zrzut". Aby zmienić tryb widoku należy nacisnąć przycisk "powiększenia" na pasku narzędzi lub użyć komendy "**Opcje / Zrzucać**".

W trybie "Zrzucania", wyświetlana jest tylko jedna zmienna. Jej wartość wyświetlana jest w formacie numerycznym/symbolicznym w górnej części okna oraz w binarnym formacie "zrzutu". Tryb ten pozwala na śledzenie wartości szesnastkowej każdego bajtu wartości zmiennej.

Wyświetlanie "Zrzutu" jest bardzo przydatne do śledzenia i zrozumienia ciągów komunikatów zawierających znaki nie drukowane.

A.17 Debagowanie programów ST i IL

Podczas symulacji lub debagowania natychmiastowego programów ST i IL, do tekstu programu nie mogą być wprowadzane żadne modyfikacje.

IL W przypadku programów IL, instrukcje są formatowane w widoku listy. Bieżąca wartość zmiennej użytej w instrukcji jest wyświetlana w tej samej linii. Dwukrotne kliknięcie instrukcji pozwala zmienić wartość odpowiadającej jej zmiennej.

ST W przypadku programów ST, okno Listy Śledzenia jest osadzone w oknie edytora. Rozmiar widoków można zmieniać, przeciągając linię podziału między nimi przy pomocy myszy.

ISaGRAF wyświetla nazwę, bieżącą wartość i tekst komentarza dla każdej zmiennej na liście. Wielkość kolumn może być zmieniana poprzez przeciągnięcie myszą oddzielającej je linii na pasku tytułowym listy.

Zapisywanie listy na twardym dysku

Komenda "**Plik / Zapisz listę**" jest używana do zapisu list zmiennych na dysku, pod taką samą nazwą jak edytowany program. Lista ta będzie ponownie ładowana przy każdym otwarciu program ST lub IL w trybie debagowania. Lista ta może również być swobodnie otwierana i modyfikowana przy użyciu narzędzia Śledzenie List, uruchamianego komendą "**Śledź / Śledź listę**" w oknie debagera.

Wstawianie zmiennych na listę

Komenda "**Edycja / Wstaw zmienną**" wstawia kolejną zmienną na listę. Nazwa zmiennej wybierana jest na liście obiektów zdefiniowanych w słowniku projektu. W ten sposób użytkownik nie musi wprowadzać identyfikatora ręcznie. Zmienna jest wstawiana przed zmienną aktualnie wybraną na liście. Lista nie może zawierać więcej niż **32** zmiennych. Ta sama zmienna może się pojawić na liście tylko jeden raz.

Kiedy nazwa zmiennej jest podświetlona w tekście ST, naciśnięcie tego przycisku na pasku narzędzi lub wybranie komendy "**Edycja / Śledź zaznaczenie**" przesyła zmienną bezpośrednio do osadzonej listy śledzenia.

Zmiana wybranej zmiennej

Komenda "**Edycja / Zmień zmienną**" zmienia wybraną zmienną na inną zmienną. Można również wykorzystać komendę "**Wytnij zmienną**" aby usunąć z listy wybraną zmienną.

A.18 Synoptyka

Narzędzie Synoptyka pakietu ISaGRAF pozwala użytkownikowi zdefiniować listy kontrolne, które mogą być wyświetlane albo w formie graficznej, albo jako listy podczas debagowania. Elementy graficzne muszą być połączone ze zmiennymi projektu ISaGRAF. Rysunek graficzny jest definiowany i animowany w czasie rzeczywistym.
Aby wymusić wartość zmiennej, należy dwukrotnie kliknąć odpowiadający tej zmiennej element układu graficznego lub listy lub po jej wybraniu nacisnąć ENTER.
Można również zablokować dokument (uniemożliwić wszelkie modyfikacje) używając komendy "**Plik / Zablokuj**". Kiedy dokument jest zablokowany, można nadal wymuszać zmienne dwukrotnie klikając ich symbol.

A.18.1 Budowanie zobrazowania

Graf jest zbudowany z rysunków umieszczonych w tle (map bitowych lub metaplików) oraz zestawu elementów graficznych, które będą animowane podczas debagowania. Aby wprowadzić graf, należy wykonać następujące operacje: wstawić rysunki tła, wstawić elementy graficzne, połączyć obiekty ze zmiennymi projektu

Rysunki tła

Rysunki umieszczone w tle to "mapy bitowe" (.BMP) lub “metapliki” (.WMF). Ilość plików zawartych w układzie graficznym jest nieograniczona. W układzie graficznym rysunki można przemieszczać lub można zmieniać ich rozmiary. Nie pojawiają się one w układzie listy. Rysunki tworzy się przy pomocy innych narzędzi. Synoptyka nie zawiera narzędzia do rysowania. Komenda "**Opcje / Kolor tła**" jest używana do wyboru koloru wypełnienia pustego obszaru w zobrazowaniu.

Uwaga: Mapy bitowe zajmują dużą ilość pamięci. Zalecane jest odpowiednie ustalenie rozmiarów rysunków oraz ograniczenie nieużywanego obszaru w obrębie prostokąta mapy bitowej.

Wyświetlanie pojedynczego tekstu

Elementy typu “Pojedynczy tekst” to tekst wpisany w prostokącie. Tekst wyświetlany jest wartością powiązanej zmiennej. Dlatego też takie elementy mogą być powiązane ze zmienną typu komunikat.

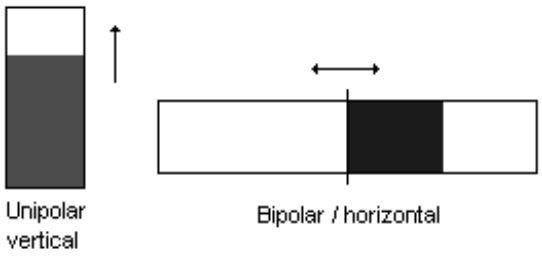
Prostokąt, w którym wyświetlany jest tekst może być wypełniony kolorem lub może być przezroczysty. Podczas zmiany rozmiarów elementu następuje dopasowanie czcionki używanej do wyświetlania tekstu do wysokości prostokąta.

Jednobiegunowe i dwubiegunowe wykresy słupkowe

Wykresem słupkowym jest prostokąt z częściowo wypełnionym wnętrzem, który reprezentuje wartość liczbową skojarzonej zmiennej. Pozostała część prostokąta może być opcjonalnie wypełniona kolorem. Wykres słupkowy może być pionowy lub poziomy.

Jednobiegunowe wykresy słupkowe mogą rozrastać się w dowolnym kierunku: w górę, w dół, w lewo lub w prawo.

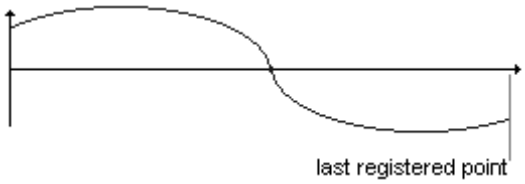
Dwubiegunowe wykresy słupkowe mogą rozrastać się w kierunku dodatnim albo ujemnym, zgodnie z wartością skojarzonej zmiennej. W przypadku dwubiegunowego wykresu słupkowego, maksymalna dopuszczalna wartość jest taka sama dla skali dodatniej i ujemnej.



Krzywe

Możliwe jest wstawienie do zobrazowania krzywej. Krzywa ilustruje historię skojarzonej zmiennej. Choć nie jest to precyzyjne narzędzie pomiarowe, to może dostarczać informacji o synchronizacji różnych zmiennych, przydatnych w procesie debagowania.

Krzywa przechowuje 200 ostatnich wartości zmiennej. Ilość próbek nie zmienia się podczas zmiany rozmiarów elementu krzywej w układzie graficznym.

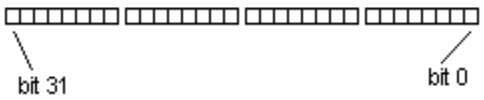


Ikony binarne

Element typu “ikona binarna” jest wykorzystywany do wyświetlania stanu binarnego. Jeden plik definiuje ikonę (.ICO) dla wartości FAŁSZ lub 0. Inna ikona jest zdefiniowana dla wszystkich pozostałych wartości niezerowych. Ponieważ Synoptyka nie zawiera edytora ikon, pliki ikon należy przygotować przy użyciu narzędzia zewnętrznego.

Pola bitowe

Element typu “pole bitowe” prezentuje w formie graficznej 32 bity wartości typu integer. Najmniej znaczący bit jest wyświetlany zawsze po prawej stronie. Nie zaleca się stosowania pól bitowych do innych typów danych, takich jak wartości analogowe typu real, ponieważ wyświetlane informacje mogą prowadzić do nieporozumień.



Wybór, przemieszczanie i zmiana rozmiarów elementów

Wybieranie elementów graficznych jest wymagane w przypadku większości komend edycyjnych. Synoptyka umożliwia wybieranie jednego lub większej ilości obiektów na obszarze grafu. Aby wybrać kilka obiektów, należy włączyć opcję "**wybór**" (przycisk ze strzałką) na pasku narzędzi edytora. Aby wybrać jeden obiekt, użytkownik musi jedynie kliknąć jego symbol. Aby wybrać listę obiektów, należy przeciągnąć myszą po rysunku zaznaczając prostokątny obszar. Wszystkie obiekty graficzne, które otacza ten prostokąt zostaną zaznaczone jako "**wybrane**". Symbol graficzny wybranego obiektu jest otoczony niewielkimi czarnymi kwadratami.

Dokonanie nowego wyboru spowoduje to, ze poprzednio wybrane obiekty przestaną być oznaczone jako wybrane. Aby usunąć bieżące zaznaczenie(a), wystarczy kliknąć myszą pusty obszarze poza prostokątem otaczającym wybrane obiekty.

Aby przemieścić obiekty, należy je najpierw wybrać. Następnie należy umieścić kursor na krawędzi wybranego elementu i przeciągnąć go w inne miejsce.

Aby zmienić rozmiar obiektu, należy go najpierw wybrać. Następnie należy umieścić kursor na jednym z niewielkich kwadratów wyświetlanych na krawędzi zaznaczenia i przeciągnąć je w odpowiednim kierunku, zmieniając rozmiary elementu. Można również zmienić rozmiar rysunków. W takim przypadku, odpowiednia mapa bitowa lub metaplik są rozciągane tak , aby pasowały do nowego określonego prostokąta.

Grupowanie elementów / rozgrupowanie

Elementy można grupować razem, aby można było zarządzać nimi jak jednym elementem. Aby utworzyć grupę, należy wybrać elementy w układzie graficznym i wykonać komendę "**Edycja / Grupuj**". Komenda "**Edycja / Rozgrupuj**" jest używana do rozbicia grupy na pojedyncze elementy.

Grupa może zawierać rysunek. Grupa może również zawierać inną grupę.

Kiedy elementy są zgrupowane, nie można już zmienić ich stylu. Elementy grupy są nadal wyświetlane, ale nie mogą być użyte (przez dwukrotne kliknięcie) do modyfikowania wartości skojarzonych z nimi zmiennych.

Grupa pojawia się w układzie listy jako tylko jedna linia.

A.18.2 Widok

Używając tego przycisku, można w dowolnym momencie przejść z układu graficznego do układu listy lub odwrotnie. Można również wykorzystać komendę "**Opcje / Widok - lista/grafika**".

W układzie listy elementy prezentowane są w klasycznym oknie w formie listy. Wysokość elementu jest obliczana w zależności od stylu jego rysowania. Rysunki (mapy bitowe i metapliki) nie są widoczne na układzie listy. Na układzie listy jest dostępne wybieranie. Powinno ono być wykorzystywane do ustawiania stylu elementu lub do zmiany wartości zmiennej. W trybie tym nie jest dostępne wybieranie wielokrotne i wykorzystujące je komendy.

Kolejność elementów na liście można zmienić używając komend "**Edycja / Przenieś na liście**". Element, który ma zostać przeniesiony musi zostać na liście wybrany.

A.18.3 Definiowanie stylu elementu

Styl graficzny i ustawienia istniejącego elementu mogą być modyfikowane przez dwukrotne kliknięcie jego symbolu w polu graficznym lub przez wybranie komendy "**Edycja / Ustaw styl pozycji**" po wybraniu elementu w układzie graficznym lub w układzie listy. Okno dialogowe "Styl" jest otwierane również wtedy, kiedy do dokumentu jest dodawany nowy element. Okno to zawiera następujące informacje, które mogą zostać wybrane przez użytkownika:

Styl i ustawienia grafiki:
Styl wyświetlania (pojedynczy tekst, wykres słupkowy, krzywa, itp.) elementu może być zmieniany dynamicznie. Kiedy stosowane są kolory tła i pierwszego planu,

można je dostosować indywidualnie w odpowiednich oknach. Kiedy stylem jest "ikona binarna", to należy określić nazwę ścieżki do odpowiednich plików .ICO. Przyciski "**...**" obok tych regulatorów służą do szukania istniejących na dysku plików z ikonami.

Skala:
Jest to maksymalna wartość, która może zostać wyświetlona na wykresach słupkowych lub na krzywej. W przypadku dwuosiowych wykresów słupkowych i krzywych, ta sama wartość bezwzględna jest wykorzystywana zarówno dla osi dodatniej jak i ujemnej.

Nazwa zmiennej:
Kiedy pole "**Nazwa**" jest polem aktywnym, naciśnięcie przycisku "**...**" obok regulatorów pozwala użytkownikowi odnaleźć nazwy zmiennych już zadeklarowanych w słowniku projektu.

Podpis:
W pobliżu elementu graficznego na układzie graficznym może być wyświetlany podpis. Położenie (góra, dół, lewo lub prawo) i treść tekstu podpisu mogą być indywidualnie dostosowane. Podpis może być dowolną kombinacją nazwy zmiennej i jej wartości, sformatowanych w postaci tekstu. Dopasowywanie podpisu nie ma wpływu na układ listy.

Uwzględnianie dla zmiennej:
Jeśli ustawiona jest opcja "**Uwzględniaj dla zmiennej** ", użytkownik może modyfikować wartość skojarzonej zmiennej podczas debagowania przez dwukrotne kliknięcie symbolu graficznego elementu.

A.18.4 Komendy menu "Plik"

Menu "**Plik**" zawiera komendy pozwalające użytkownikowi na zarządzanie kompletnym dokumentem.

Komenda "**Nowy**" w menu "**Plik**" uruchamia edycję nowego dokumentu. Ilość dokumentów zdefiniowanych dla projektu nie jest ograniczona przez ISaGRAF. Przed edycją nowego grafu (obrazu synoptycznego) jest zamykany poprzednio otwarty graf. Synoptyka nie może być używana do edycji kilku grafów równocześnie. Jednakże można otwierać równocześnie wiele okien w Synoptyce, z których każde może służyć do edycji innego dokumentu.

Komenda "**Otwórz**" menu "**Plik**" pozwala użytkownikowi na zamknięcie edytowanego aktualnie dokumentu i rozpoczęcie edycji kolejnego dokumentu bieżącego projektu. Nowo wybrany dokument zastępuje dokument bieżący w oknie edycyjnym. Wybierając nowy dokument, można użyć przycisku "**Usuń**" aby usunąć istniejący plik i wyczyścić katalog projektu. Kiedy usuwany jest graf, to pliki ikon i mapy bitowe, do których on się odwołuje nie są kasowane.

Komenda "**Zapisz**" menu "**Plik**" zachowuje edytowany aktualnie dokument na dysku. Jeżeli jest to nowy dokument bez tytułu, przed zapisem użytkownik musi nadać mu nazwę. Nadawanie nazwy dokumentowi musi być zgodne z następującymi zasadami:

- Długość nazwy nie może przekraczać **8** znaków
- Pierwszy znak musi być **literą**
- Kolejne muszą być **literami**, **cyframi** lub znakiem **podkreślenia**
- Nadawanie nazw uwzględnia wielkość liter

Komenda "**Zapisz jako**" menu "**Plik**" pozwala użytkownikowi zachować aktualnie edytowany dokument pod inną nazwą.

A.18.5 Uwagi dla użytkowników ISaGRAF V3.2

Synoptyka może odczytywać grafikę i listy wykresów czasowych zbudowane przy pomocy narzędzi ISaGRAF w3.0 lub w3.2. Pliki takie pojawiają się w oknie dialogowym "**Otwórz**" wraz z opisem ich pochodzenia. Pliki te mogą być odczytywane i swobodnie modyfikowane przy użyciu Synoptyki.

Przy otwieraniu grafiki ISaGRAF w3.2, dokument zostaje automatycznie oznaczony jako "Zablokowany". Wprowadzenie zmian w tej grafice wymaga wyłączenia z menu "**Plik**" opcji "Zablokowane".

Kiedy otwierana jest grafika lub lista wykresu czasowego ISaGRAF w3.2, Synoptyka zawsze proponuje ich zapisanie w formacie własnym Synoptyki. Okno dialogowe "**Zapisz jako**" jest otwierane automatycznie przy zamykaniu takiego dokumentu.

A.19 Pobieranie aplikacji

ISaGRAF obsługuje pobieranie aplikacji przechowywanych w oprogramowaniu wbudowanym. Procedura pobierania komunikuje się z oprogramowaniem wbudowanym w celu załadowania osadzonego spakowanego kodu źródłowego (EZS), a następnie odtworzenia załadowanego projektu w środowisku pakietu ISaGRAF.
Projekt działający w dołączonym systemie oprogramowania wbudowanego może zostać pobrany, jeśli stosowana jest wersja oprogramowania wbudowanego V3.22 lub późniejsza oraz jeśli spakowany kod źródłowy został osadzony przy pomocy aplikacji. Osadzanie kodu źródłowego do pobierania jest funkcja opcjonalną. W celu uzyskania dalszych informacji dotyczących konfiguracji projektu dla potrzeb pobierania, należy zapoznać się z następującymi tematami:

A.19.1 Pobieranie projektu

Okno dialogowe "**Pobierz**" jest wywoływane przy pomocy komendy "**Pliki**" Menedżera Projektu ISaGRAF. Pobieranie nie odnosi się do projektu istniejącego w pakiecie ISaGRAF. Projekt wybrany aktualnie z listy modułu Zarządzanie Projektem nie ma związku z mechanizmem pobierania. Aby pobrać aplikację działającą w oprogramowaniu wbudowanym, należy:

1- upewnić się, czy oprogramowanie wbudowane jest właściwie podłączone
2- skonfigurować parametry komunikacyjne zgodnie z parametrami łącza
3- nacisnąć przycisk "**Uruchom**"

Pobieranie osadzonego spakowanego kodu źródłowego (EZS) i jego dekompresja może trwać kilka sekund. Komunikaty w oknach dialogowych informują o zakończeniu pobierania lub o wystąpieniu błędu.

Nazwą użytą do utworzeniu projektu ISaGRAF jest ta nazwa, która jest wczytywana do oprogramowania wbudowanego poprzez łącze. Jeżeli nazwa taka jest już wykorzystywana przez projekt istniejący w pakiecie ISaGRAF, pojawi się zapytanie czy zastąpić ją, czy też wybrać nazwę nieużywaną. Po zakończeniu pobierania nie można anulować rejestracji załadowanych kodów źródłowych jako projektu. Pobrany projekt jest teraz gotów i może zostać otwarty.

Możliwe błędy

Podczas pobierania projektu pojawić się mogą następujące błędy. Informacja o błędach pojawia się w oknie dialogowym "Pobierz".

- nie można nawiązać połączenia z oprogramowaniem wbudowanym
- podłączone oprogramowanie wbudowane jest systemem ISaGRAF w wersji wcześniejszej niż 3.22
- w oprogramowaniu wbudowanym nie pracuje żadna aplikacja

- w oprogramowaniu wbudowanym nie jest osadzone żaden kod EZS

A.19.2 Ustawienia parametrów komunikacji

Naciśnięcie przycisku "**Ustawienia**" pozwala użytkownikowi zdefiniować parametry łącza używanego do komunikacji pomiędzy pakietem ISaGRAF a oprogramowaniem wbudowanym. Przed uruchomieniem pobierania należy upewnić się, czy ustawione parametry są zgodne z używanym oprogramowaniem wbudowanym.

A.19.3 Przygotowywanie projektu do pobrania

Generator Kodu ISaGRAF musi otrzymać informację, że spakowany kod źródłowy ma zostać osadzony w kodzie aplikacji, aby istniała możliwość późniejszego wykonania operacji pobrania. W tym celu należy nacisnąć przycisk "**Pobierz**" w oknie dialogowym "**Opcje kompilatora**". Inne okno dialogowe pozwala na opcjonalną kontrolę osadzania spakowanego kodu źródłowego. W tym przypadku osadzone zostanie jedynie wymagane minimum plików kod źródłowego. Zaznaczając inne pola kontrolne można osadzić również pliki opcjonalne.

Ważna uwaga: Biblioteki nie są ładowane wraz z osadzonym kodem źródłowym. Dotyczy to także funkcji, bloków funkcyjnych oraz kart i zespołów We/Wy.

Pliki opcjonalne

Poza minimalnym wymaganym kodem źródłowym osadzane mogą być również następujące pliki. Są one opcjonalne, ponieważ ich wybranie prowadzi do większych wymagań w stosunku do pamięci oprogramowania wbudowanego.

Opis projektu: Jeśli nie zostanie osadzony, opis projektu po pobraniu będzie pokazywał jedynie datę pobrania.

Zabezpieczenie hasłem: Funkcja pobierania nie jest zabezpieczona hasłem. Aby zabezpieczyć pobrany projekt, należy wraz z kodem źródłowym osadzić system ochrony hasłem.

Komentarze do nie podłączonych kanałów We/Wy: ISaGRAF umożliwia wprowadzanie tekst opisu dla nie podłączonych kanałów We/Wy. Nie należy zaznaczać tej opcji, jeśli praca odbywa się tylko z podłączonymi kanałami We/Wy.

Historia modyfikacji: Jest to globalna historia modyfikacji projektu.

Pliki dziennik : Plik dziennika każdego programu zawiera notatki zapisane przez użytkownika oraz historię komunikatów wyprowadzany przez kompilator odnośnie

programu. Osadzanie plików dziennika może zająć dużą część pamięci oprogramowania wbudowanego.

Listy zmiennych i wykresy czasowe: Są to pliki tworzone podczas debagowania. Zawierają listy nazw zmiennych dla potrzeb monitorowania list lub wykresów czasowych.

Grafika, ikony i mapy bitowe: Plik ten obejmuje grafikę ISaGRAF oraz wszystkie załączone pliki z ikonami i mapami bitowymi, jeśli znajdują się one w katalogu projektu. Uwaga: sadzenie plików dziennika może zająć dużą część pamięci oprogramowania wbudowanego.

A.19.4 Przechowywanie spakowanego kodu źródłowego w oprogramowaniu wbudowanym

Osadzony spakowany kod źródłowy (EZS) jest zapamiętywany w wygenerowanym kodzie z zasobami. Wygenerowane zasoby zwane są "**EZS**". Jeśli wybrano osadzanie kodu źródłowego, to nie można nadać jego nazwy innemu zasobowi. Osadzenie kodu źródłowego nie oznacza jakiegokolwiek ograniczenia definicji zasobów. Osadzanie źródła nie ma wpływu na plik definicji zasobów utworzony przez użytkownika.
Dalsze szczegóły i informacje na temat zasobów podano w opisie Generatora Kodu w dokumentacji ISaGRAF.

A.19.5 Wymagania dotyczące pamięci oprogramowania wbudowanego

Osadzony spakowany kod źródłowy (EZS) potrzebuje w systemie oprogramowania wbudowanego dodatkowej pamięci, w której jest przechowywany wraz z kodem aplikacji. Ogólnie szacuje się, że minimalny EZS (bez dodatkowych opcji wybranych do osadzania kodu źródłowego) jest 1,5 razy większy od kodu wykonywalnego. Oznacza to, iż osadzanie EZS powiększy wielkość załadowanego kodu 2,5 razy.

W niektórych systemach oprogramowania wbudowanego, opartych na pamięci segmentowej mogą wystąpić pewne ograniczenia. Ponieważ kody EZS są przechowywane w wygenerowanym kodzie jako zasoby, muszą one być przechowywane w tym samym segmencie danych, co kod aplikacji.

A.19.6 Informacje dotyczące pobieranego projektu

Pobierany projekt zawiera wszystkie pliki i dane wymagane do ponownej kompilacji. W zależności od opcji wybranych podczas poprzedniej kompilacji może on również zawierać pliki pomocnicze, takie jak pliki opisu projektu i dziennika programu.
Przed debagowaniem lub monitorowaniem projektu, najeży go skompilować (utworzyć). Uwaga: ze względu na używanie przez ISaGRAF stempla z datą kompilacji, podczas otwierania debagera pojawi się informacja, iż aplikacje pakietu ISaGRAF i systemu oprogramowania wbudowanego mają różne kody wersji.

Ważna uwaga: Biblioteki nie są ładowane wraz z osadzonym kodem źródłowym. Przed ponowną kompilacją pobranej aplikacji należy się upewnić, czy wersja pakietu ISaGRAF posiada zainstalowane odpowiednie bloki funkcyjne i funkcje biblioteki.

A.19.7 Problemy kompatybilności

Pobieranie jest obsługiwane przez pakiet i oprogramowanie wbudowane ISaGRAF w wersji 3.22 i późniejszych. Do obsługi pobierania dokonano rozszerzenia protokołu komunikacyjnego.
Osadzanie spakowanego kodu źródłowego (EZS) w systemach oprogramowania wbudowanego ISaGRAF w wersjach 3.03 do 3.21 nie podlega żadnym ograniczeniom, ponieważ kody EZS są przechowywane w kodzie aplikacji jako zasoby standardowe. Jednak w takim przypadku nie można pobrać osadzonych informacji, ponieważ tego rodzaju oprogramowanie wbudowane nie obsługuje wymaganych usług komunikacyjnych.

A.20 Korzystanie z narzędzia diagnostycznego

"**Narzędzie diagnostyczne**" jest podzbiorem debagera ISaGRAF. Umożliwia on użytkownikowi końcowemu pracę na wcześniej zdefiniowanym zbiorze zmiennych w celu zbadania i skontrolowania procesu. Debager ISaGRAF jest efektywnym narzędziem zawierającym zawansowane funkcje. Narzędzie diagnostyczne zapewnia bezpieczny sposób kontrolowania aplikacji oprogramowania wbudowanego pod kątem końcowych operacji uruchomienia i utrzymania w sprawności. Narzędzie diagnostyczne ISaGRAF uruchamiane jest bezpośrednio z grupy ISaGRAF w Menedżerze Programów, poprzez dwukrotne kliknięcie ikony:

Lista istniejących projektów wyświetlana jest w oknie dialogowym. Pozwala ona użytkownikowi uruchomić ograniczone debagowanie istniejącej, już załadowanej aplikacji ISaGRAF. Naciśnięcie przycisku "**OK**" uruchamia ograniczone debagowanie wybranego projektu. Naciśnięcie przycisku "**Anuluj**" zamyka okno dialogowe. Przycisk "**Ustaw**" stosowana jest do ustawienia łącza komunikacyjnego pomiędzy pakietem ISaGRAF a oprogramowaniem sterownika PLC. Więcej informacji o tej komendzie podano w rozdziale "**Zarządzanie programami**" w tym podręczniku.

Uwaga: Narzędzie diagnostyczne ISaGRAF (ograniczony debager) nie może być wykorzystane do załadowania, zatrzymania lub uaktualnienia aplikacji uruchomionej w sterowniku PLC z oprogramowaniem wbudowanym. Jeśli projekt wybrany w oknie dialogowym Narzędzie diagnostyczne nie jest taki sam, jak ten zainstalowany i pracujący w sterowniku PLC, to wykonanie jakiejkolwiek operacji nie będzie możliwe.

Kiedy ograniczony debager ISaGRAF jest uruchomiony i prawidłowo podłączony do aplikacji w PLC z oprogramowaniem wbudowanym, dostępne są następujące komendy:

- Śledź listę zmiennych
- Śledź dokumenty graficzne przy pomocy Synoptyki

A.21 Korzystanie z symulatora ISaGRAF

Symulator jądra ISaGRAF jest uruchamiany wraz z debagerem po wybraniu komendy "Symuluj" menu "Debaguj" w oknie Zarządzanie Programem. Symulator to kompletny system oprogramowania wbudowanego ISaGRAF obsługujący standardowe funkcje ISaGRAF oraz bloki funkcyjne i funkcje "C" standardowej biblioteki dostarczonej przez CJ International. Karty We/Wy są symulowane graficznie w oknie. Symulowany może być dowolny typ karty We/Wy. Karty zdefiniowane jako "Karty wirtualne" podczas połączenia We/Wy pojawiają się również w oknie symulacji.

A.21.1 Powiązania z debagerem

Symulator jądra obsługuje pełną komunikację z debagerem ISaGRAF, więc wszystkie możliwości debagera mogą również zostać wykorzystane podczas symulacji. Symulator jądra działa zawsze na bieżącej aplikacji ISaGRAF. Podczas symulacji, komendy debagera "**Uruchom**", "**Zatrzymaj**", "**Załaduj**" czy "**Uaktualnij**" nie są już dostępne. Symulator nie może być używany, jeśli w opcjach kompilacji przed zbudowaniem kodu oprogramowania wbudowanego nie wybrano "**SYMULACJI**" oprogramowania wbudowanego. Zamknięcie okna symulatora pociąga za sobą również zamknięcie okna debagera (i każdego okna ISaGRAF otwartego podczas sesji debagera).

A.21.2 Symulacja We/Wy

Karty We/Wy pojawiają się w oknie symulatora, posiadając w tytule nazwę i numer gniazda. Obsługiwane są wszystkie standardowe typy We/Wy ISaGRAF (binarne, analogowe lub komunikaty). Kanały kart wejściowych są wyświetlane ze specjalnymi klawiszami i polami. Kanały kart wyjściowych wyświetlane są z lampkami graficznymi i polami danych.

Wejścia binarne: Wejście binarne jest reprezentowane przez kwadratowy, zielony przycisk. Numer kanału wyświetlany jest wraz z przyciskiem We/Wy. Wartością wejściową jest PRAWDA, jeśli przycisk jest wciśnięty. Kliknięcie przycisku zmienia odpowiadającą mu wartość We/Wy. Przy pomocy prawego przycisku myszy ustawia się wejście tylko wtedy, gdy przycisk jest naciśnięty.

Wyjścia binarne: Wyjście binarne jest reprezentowane przez niewielkie kółko. Wraz z We/Wy jest wyświetlany numer kanału. Wartością wyjściową jest PRAWDA gdy symbol graficzny jest podświetlony.

Wejścia analogowe: Kanał wejścia analogowego to proste pole numeryczne, w które można wprowadzić wartość odpowiadającego mu wejścia. Kliknięcie okna

wyświetla znak „^". Można wtedy wprowadzić dla kanału nową wartość. Po wprowadzeniu nie trzeba naciskać klawisza **ENTER**. Wejścia analogowe można wprowadzać z podstawą dziesiętną jak i szesnastkową. Przycisków ze strzałkami "w górę/w dół" używa się do zwiększenia lub zmniejszenia bieżącej wartości.

Wyjścia analogowe: Analogowe kanały wyjściowe to wyjściowe pola numeryczne. Wartość wyjściowa może być wyświetlana zarówno jako liczba dziesiętna jak i szesnastkowa. Użytkownik nie może przeprowadzić żadnej operacji na kanale wyjściowym.

Wejścia komunikatu: Kanał wejściowy komunikatu to proste pole tekstowe, w które wprowadza się odpowiadającą mu wartość wejściową. Kliknięcie okna wyświetla znak „^". Można wtedy wprowadzić dla kanału nową wartość. Po wprowadzeniu nie trzeba naciskać klawisza **ENTER**.

Wyjścia komunikatu: Kanał wyjściowy komunikatu to tekstowe pole wyjściowe. Użytkownik nie może przeprowadzić żadnej operacji na kanale wyjściowym.

A.21.3 Komponenty biblioteki

Symulator ISaGRAF w pełni obsługuje konwersje standardowe, funkcje i bloki funkcyjne dostarczone przez CJ International. Poniżej znajduje się lista obsługiwanych obiektów:

- ***Funkcje konwersji:***

  bcd, scale

- ***Funkcje:***

  abs, acos, ArCreate, ArRead, ArWrite, ascii, asin, atan, char, cos, delete, expt, find, insert, left, limit, log, max, mid, min, mlen, mod, mux4, mux8, odd, rand, replace, right, rol, ror, sel, shl, shr, sin, sqrt, tan, trunc

- ***Bloki funkcyjne:***

  average, blink, cmp, ctd, ctu, ctud, derivate, f_trig, hyster, integral, lim_alrm, r_trig, rs, sema, sr, stackint, tof, ton, tp

Konwersje zdefiniowane przez użytkownika, bloki funkcyjne i funkcje "C" zwykle nie są integrowane z Symulatorem ISaGRAF. Obiekty takie są zazwyczaj zaprojektowane do wykorzystania zasobów sprzętowych i programowych systemu oprogramowania wbudowanego. Zasoby takie nie są na ogół dostępne w systemie Windows. Symulator ISaGRAF zapewnia następujące standardowe zachowania w odniesieniu do każdej konwersji użytkownika, funkcji czy bloku funkcyjnego:

- Kiedy nowa konwersja jest przetwarzana przez symulator, zastępuje ją konwersja "null" (pusta). Oznacz to, iż wartość fizyczna zmiennych analogowych jest zawsze równa wartości elektrycznej (wprowadzanej lub wyświetlanej na panelu Symulatora).
- Kiedy symulator uruchamia nowy blok funkcyjny lub funkcję "C", nie jest przetwarzana żadna operacja. Wartość wynikowa nie jest ustawiana.

A.21.4 Opcje

Komendy menu "**Opcje**" umożliwiają użytkownikowi kontrolowanie wyświetlania wartości We/Wy na panelu symulatora. Użytkownik może ustawić lub wyłączyć te opcje w dowolnym momencie debagowania.

- Kiedy ustawiona jest opcja "**Kolory**", kanały We/Wy są wyświetlane jako kolorowe mapy bitowe. Jeżeli na niektórych ekranach LCD nie można rozróżnić kolorów, użytkownik powinien wyłączyć tę opcję, aby otrzymać wyraźną, czarno-białą grafikę wejść i wyjść dla kanałów We/Wy.

- Kiedy ustawiona jest opcja "**Nazwy zmiennych**", obok każdego kanału We/Wy wyświetlana jest etykietka zawierająca nazwę powiązanej z nim zmiennej We/Wy. Wyłączenie tej opcji pozwala użytkownikowi zmniejszyć rozmiary panelu symulatora.

- Kiedy ustawiona jest opcja "**Wartości szesnastkowe**", wartość każdego analogowego kanału wejściowego jest wyświetlana i wprowadzana w formacie szesnastkowym.

- Kiedy ustawiona jest opcja "**Zawsze na górze**", okno symulatora jest zawsze widoczne, nawet jeśli aktywne wejście znajduje się w innym oknie.

A.21.5 Zapisywanie i przywracanie stanów wejściowych

Przy użyciu symulatora ISaGRAF kanały wejściowe są wymuszane operacjami ręcznymi działającymi na przełączniki i przyciski edycji na panelu symulacji. W dowolnym momencie można wybrać poniższe komendy w menu "**Narzędzia**", aby zapisać bądź przywrócić stany wszystkich kanałów wejściowych:

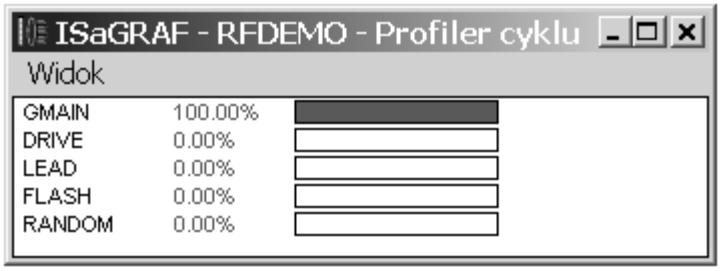
Załaduj stan wejść	Ustawia wartości kanałów wejściowych zgodnie z wartościami przechowywanymi w pliku stworzonym na dysku przy pomocy komendy "Zapisz układ wejść".
Zapisz stan wejść	Zapisuje stan kanałów wejściowych do pliku tak, aby można było przywrócić je później przy użyciu komendy "Załaduj układ wejść". Plik przechowywany jest w katalogu projektu, a co za tym idzie zapisywany z innymi plikami projektu przez narzędzie archiwizacji ISaGRAF.

Uwaga: Na dysku zapisywane są tylko kanały wejściowe z nazwą (te, z którymi powiązana jest zmienna).

A.21.6 Profiler Cyklu

Profiler cyklu systemu ISaGRAF to rozbudowane narzędzie diagnostyczne, pokazujące w jaki sposób czas cyklu jest rozdzielany pomiędzy różne programy, funkcje i bloki funkcyjne aplikacji. Narzędzie to jest bardzo przydatne do przeprowadzenia szybkiej diagnostyki wydajności aplikacji i wskazuje programiście te części kodu, które mogą wymagać optymalizacji.

Profiler Cyklu uruchamiany jest komendą "**Narzędzia / Profiler Cyklu**" menu okna Symulatora ISaGRAF. Wyświetla on procentowo część czasu cyklu, zużywaną na wykonanie każdego programu, funkcji lub bloku funkcyjnego:



Kiedy wybrana zostanie opcja "**Widok / Uśredniaj**", wyświetlone informacje, to średnia procentowa, obliczona od uruchomienia aplikacji lub od ostatniego wybrania komendy "**Widok / Kasuj**".

Kiedy opcja "**Widok / Uśredniaj**" nie jest włączona, wyświetlone informacje pokazują pomiary dokonane podczas wykonywania ostatniego cyklu. Funkcji tej można używać również gdy aplikacja działa w trybie "**Cykl po cyklu**", aby uzyskać zestaw pomiarów zależnych od kontekstu aplikacji.

Komendy "**Widok / Kopiuj**" używa się do skopiowania nazw programów i wartości procentowych do schowka Windows w formacie ASCII. Dane te można później wkleić do dokumentów tekstowych lub popularnych arkuszy kalkulacyjnych.

Ważne uwagi:

Pomiary te nie są precyzyjne. Obliczanie wartości procentowych oparte jest na liczeniu instrukcji TIC (kodu niezależnego od oprogramowania wbudowanego), z uwzględnieniem różnych czasów wykonania instrukcji. Wyliczenie nie uwzględnia czasu spędzonego w blokach funkcyjnych i funkcjach "C".

Wartość wyświetlona dla funkcji i bloków funkcyjnych, to suma wszystkich „czasów wywołania" z programów aplikacji w tym samym cyklu

Obliczanie czasu opiera się na kodzie TIC i nie zapewnia solidnej informacji, jeśli rzeczywisty kod aplikacji jest generowany w języku "C" oraz budowany przy użyciu kompilatora "C".

A.21.7 Skrypty symulacyjne

Symulator ISaGRAF zawiera narzędzie służące do budowania i uruchamiania skryptów symulacyjnych. Skrypty są opisywane w prostym języku tekstowym, podobnym do ST i używane do automatyzacji testów w symulatorze ISaGRAF.

Edytor skryptów symulacyjnych uruchamia się komendą "**Narzędzia / Skrypty symulacyjne**" w oknie Symulator. Poniżej znajduje się ramka edytora skryptów:

Górne okno, to edytor tekstowy, w którym wprowadza się instrukcje. Używa się go tak, jak innych edytorów tekstowych ISaGRAF. Obsługuje on funkcje wysokiego poziomu, takie jak wybieranie symbolu zmiennej przy pomocy myszy. Komend menu "**Opcje**" można użyć do ustawienia szerokości tabulatora i do wyboru czcionki znakowej.
Okno dolne pokazuje wszystkie komunikaty wyjściowe, kiedy skrypt jest uruchomiony. Linia podziału pomiędzy tymi oknami może być swobodnie przeciągana, umożliwiając zmianę ich rozmiarów. Okno wyjściowe może zostać ukryte podczas edycji skryptu, jednak otwiera się automatyczne przy każdorazowym uruchamianiu skryptu.

Edycja skryptów

Komend menu "Plik" używa się do zarządzania plikami skryptów:

Nowy	tworzy nowy skrypt bez nazwy
Otwórz	ładuje istniejący skrypt z pliku
Zapisz	zapisuje tekst skryptu i zawartość okna wyjścia na dysk, w katalogu projektu
Zapisz jako	zapisuje skrypt pod inną nazwą

Dla każdego skrypt w katalogu projektu ISaGRAF tworzone są dwa pliki:

<*nazwaskryptu*>.SCC	tekst skryptu (instrukcje)
< *nazwaskryptu*>.SCO	zawartość okna wyjściowego

gdzie < *nazwaskryptu*>, to nazwa skryptu. Oba pliki to standardowe pliki tekstowe, które można otwierać pod dowolnym innym edytorem tekstów.

Podczas edycji skryptu można użyć komendy "**Edycja / Wstaw symbol**", aby wybrać zadeklarowaną nazwę zmiennej do wstawienia w pozycji znaku „^”.

Uruchamianie skryptów

Skrypt przed uruchomieniem musi zostać sprawdzony i skompilowany. W razie konieczności, kontrola składni odbywa się automatycznie w momencie wydania komendy "Uruchom". Użyj następujących komend menu "**Skrypt**":

Sprawdź sprawdzenie składni i kompilacja skryptu

Uruchom skrypt rozpoczęcie wykonywania edytowanego aktualnie skryptu

W przypadku nowego skryptu bez tytułu, należy go zapisać (i wprowadzić jego nazwę) zanim zostanie rozpoczęte sprawdzanie. W przypadku skryptu posiadającego nazwę, skrypt jest zapisywany automatycznie na dysku przed sprawdzeniem składni.

Kiedy skrypt pracuje, jego zawartość nie może być zmieniana. Kiedy skrypt zostanie zakończony, wyświetlany jest komunikat. Można również przerwać wykonywanie skryptu, używając następującej komendy menu "**Skrypt**":

Przerwij skrypt kończy działanie skryptu

Wykonywanie skryptu odbywa się pomiędzy cyklami oprogramowania wbudowanego. W przypadku pętli nieskończonej zaprogramowanej w cyklu, symulator ISaGRAF zawsze zapewnia przerwanie tej pętli tak, aby cykle ISaGRAF były nadal wykonywane, a inne aplikacje ISaGRAF nie były blokowane. Interpreter skryptów ISaGRAF decyduje o przerwaniu wykonywania skryptu, jeśli taka sama "etykieta" pojawi się więcej niż raz w jednym cyklu oprogramowania wbudowanego. Wykonywanie skryptu może być również przerwane normalnie, przy pomocy instrukcji "Cycle" bądź "Wait".

Język opisu skryptów

Język opisu skryptów jest bardo prostym językiem tekstowym, podobnym do ST, lecz w którym jednak każda instrukcja jest wprowadzana w osobnej linii i nie musi być zakończona średnikiem. Poniższego przycisku na pasku narzędziowym używa się do zapoznania się z listą dostępnych instrukcji i wstawienia słowa kluczowego w miejscu położenia znaku „^":

Wstaw instrukcję (słowo kluczowe i pomoc w formie komentarzy)

Istnieją różne typy instrukcji. Pierwszy, to przypisanie (wymuszenie) zmiennej:

:=	przypisanie

Inne instrukcje pozwalają na wysyłanie komunikatów do okna wyjściowego:

Print	wysyła ciąg tekstowy lub nazwę zmiennej
PrintTime	wysyła stempel z bieżącym czasem

Jeszcze inne, są używane do synchronizacji instrukcji skryptu z cyklem ISaGRAF:

Cycle	pozwala symulatorowi ISaGRAF wykonać jeden cykl
Wait	czeka przez określony czas

Pozostałych używa się do kontroli przepływu instrukcji w skrypcie:

Labels	można wstawić etykiety w dowolnym miejscu skryptu
Goto	bezwarunkowy skok do etykiety
If goto	warunkowy skok do etykiety
End	zakończenie skryptu

Język skryptu nie uwzględnia wielkości liter. Komentarze można wstawiać na końcu każdej linii tekstu. Komentarze można wpisywać zgodnie z konwencją ST (pomiędzy znakami "**(***" i "***)**"), lub poprzedzając je znakiem "**;**".

":=" Przypisanie

Znaczenie: Wymusza wartość zmiennej ISaGRAF. Może to być zmienna wewnętrzna, kanał wejściowy lub kanał wyjściowy.

Składnia: <nazwazm> **:=** <wyrażenie_stałe>
<nazwazm > **=** <wyrażenie_stałe >

Argumenty: <nazwazm> to dopuszczalny symbol zadeklarowanej zmiennej aplikacji lub zmiennej We/Wy o reprezentacji bezpośredniej, wykorzystującej konwencję zapisu z "%".

<wyrażenie_stałe> to dopuszczalne wyrażenie stałe, które pasuje do typu określonej zmiennej. Dla zmiennych binarnych można użyć "0" i "1" zamiast "FAŁSZ" i "PRAWDA". W przypadku zmiennych typu timer, prefiks "T#" lub "TIME#" możne zostać pominięty.

Uwagi: Zmienna wejściowa wymuszona przez skrypt nie musi być blokowana. Rysowanie odpowiednich kanałów wejściowych nie jest aktualizowane, kiedy zmienna wejściowa jest wymuszana przez skrypt.
Uwaga: nie należy wymuszać analogowej zmiennej wejściowej lub wyjściowej powiązanej z konwersją, ponieważ wykonywanie skryptów nie obsługuje funkcji ani tablic konwersji.

Przykład:
```
MyBooVar := 1          (* to samo, co PRAWDA *)
MyIntVar := 1234
MyRealVar := 1.2345
MyMsgVar := 'Hello'
MyTmrVar := t#12s
```

Print

Znaczenie: Wypisuje ciąg lub wartość zmiennej w oknie wyjściowym. Tekst jest wyprowadzany w formie pojedynczej linia na końcu tekstu już wypisanego w oknie wyjściowym.

Składnia: **Print '**<*tekst*>**'**
Print <*nazwazm*>

Argumenty: <tekst> to dowolny ciąg tekstowy wpisany pomiędzy cudzysłowami
<nazwazm> to dopuszczalny symbol zadeklarowanej zmiennej aplikacji lub zmienna We/Wy o reprezentacji bezpośredniej, wykorzystująca konwencję zapisu z "%".

Uwagi: Wyjście wartości zmiennych ma zawsze format zgodny z konwencjami IEC.

Przykład:
```
Print 'Hello'
Print MyBooVar
```

Wyjście:
```
Hello
MyBoovar = TRUE
```

PrintTime

Znaczenie: Wpisuje stempel bieżącego czasu w oknie wyjściowym. Tekst jest wyprowadzany jako pojedyncza linia na końcu tekstu wypisanego już w oknie wyjściowym.

Składnia: **PrintTime**

Uwagi: Oznaczenie czasu ma format zgodny z bieżącymi ustawieniami systemu Windows

Przykład:
```
Print 'Time now is:'
PrintTime
```

Wyjście:
```
Time now is:
15:45:22
```

Cycle

Znaczenie: Zawiesza wykonywanie skryptu do momentu wykonania następnego cyklu ISaGRAF.

Składnia: **Cycle**

Notes: Instrukcje skryptu są wykonywane na początku cyklu ISaGRAF. Jeśli symulator pracuje w trybie "Cykl po cyklu", to bezpośrednio po instrukcji "Cycle" następuje cykl. Kolejne instrukcje skryptu zostaną wykonane po komendzie debagera "Wykonaj jeden cykl".

Przykład:
```
(* program ISaGRAF kopiuje A do B *)
A := 0
Cycle
   Print B
A := 1
   Print B (* nie jest wykonywany żaden cykl / B nie
ustawione na 1 *)
Cycle
Print B
```

Wyjście:
```
B = 0
B = 0
B = 1
```

Wait

Znaczenie: Zawiesza wykonywanie skryptu do momentu upływy ustalonego opóźnienia

Składnia: **Wait** <opóźnienie>

Argumenty: <opóźnienie> opóźnienie wyrażone zgodnie z konwencjami IEC dla wyrażenia stałych czasowych. Prefiks "T#" lub "TIME#" można pominąć. Wartość opóźnienia musi zawierać się pomiędzy 10 milisekundami, a 1 godziną.

Uwagi: Dokładność instrukcji "Wait" nie jest wielka, ponieważ zależy ona od systemu Windows. Ponadto opóźnienie należy rozpatrywać z dokładnością plus-minus jednego cyklu ISaGRAF.

Przy napotkaniu instrukcji "Wait", cykle ISaGRAF są wykonywane do momentu upływu ustalonego opóźnienia i przed wznowieniem wykonywania skryptu.

Przykład:
```
PrintTime
   Wait 2s
PrintTime
```

Wyjście:
```
15:45:27
15:45:29
```

Etykiety

Znaczenie: Etykiety można wstawiać w dowolnym miejscu skryptu. Są one celem instrukcji "Goto" i pozwalają kontrolować przepływ instrukcji skryptu.

Składnia: <nazwaetykiety>**:**

Argumenty: <nazwaetykiety> niepowtarzalna nazwa, zgodna z konwencjami nazywania zmiennych ISaGRAF: ograniczona do 16 znaków, rozpoczynająca się literą, po której następują litery, cyfry lub znaki podkreślenia. Podczas definiowania, nazwa zmiennej powinna kończyć się znakiem "**:**".

Uwagi: W linii, w której umieszczana jest etykieta, nie powinno się umieszczać żadnych instrukcji.
Nazwa etykiety nie powinna być taka sama jak nazwa zadeklarowanego symbolu zmiennej ISaGRAF

Przykład:
```
(* przykład skryptu z pętlą nieskończoną *)
   loop:
PrintTime
Wait 1s
Goto loop
```

Goto

Znaczenie: Bezwarunkowy skok do etykiety

Składnia: **Goto** <nazwaetykiety>

Argumenty: <nazwaetykiety> to nazwa etykiety zdefiniowanej w skrypcie.

Uwagi: Niedozwolone są skoki wstecz. W przypadku pętli nieskończonych, wykonywanie skryptu jest automatycznie przerywane przy każdej pętli, w celu umożliwienia wykonania cyklów ISaGRAF.

Przykład:
```
Print 'Before Jump'
Goto MyLabel
Print 'Within Jump' (* nigdy nie wykonywana instrukcja *)
```

```
MyLabel:
  Print 'After Jump'
```

Wyjście:
```
Before Jump
After Jump
```

If Goto

Znaczenie: Warunkowy skok do etykiety. Warunkiem jest porównanie dwóch zmiennych ISaGRAF lub porównanie zmiennej z wyrażeniem stałym.

Składnia: **If** <zm1> ***test*** <zm2> **Goto** <nazwaetykiety>
If <zm1> ***test*** <*w*yrażenie_stałe> **Goto** <nazwaetykiety>
Dostępne ***testy*** porównawcze to:

=	prawda, jeśli oba składniki mają tę samą wartość
<>	prawda, jeśli składniki mają różne wartości
<	prawda, jeśli pierwszy składnik jest mniejszy od drugiego
<=	prawda, jeśli pierwszy składnik jest mniejszy lub równy drugiemu
>	prawda, jeśli pierwszy składnik jest większy od drugiego
>=	prawda, jeśli pierwszy składnik jest większy lub równy drugiemu

Argumenty: <zm1> <zm2> to dopuszczalne symbole zadeklarowanych zmiennych aplikacji lub zmienne We/Wy o reprezentacji bezpośredniej, wykorzystujące konwencję zapisu z "%".
<wyrażenie_stałe> to dopuszczalne wyrażenie stałe, które pasuje do typu określonej zmiennej. Dla zmiennych binarnych można użyć "0" i "1" zamiast "FAŁSZ" i "PRAWDA". W przypadku zmiennych typu timer, prefiks "T#" lub "TIME#" możne zostać pominięty.
<nazwaetykiety> to nazwa etykiety zdefiniowanej w skrypcie.

Uwagi: Niedozwolone są skoki wstecz. W przypadku pętli nieskończonych, wykonywanie skryptu jest automatycznie przerywane przy każdej pętli, w celu umożliwienia wykonania cyklów ISaGRAF.

Przykład:
```
(* Skrypt ten wykonuje pętlę do momentu, gdy MyVar ma wartość TRUE *)
Loop:
If MyVar = TRUE Goto TheEnd
Print MyVar
Goto Loop
TheEnd:
```

End

Znaczenie: Kończy skrypt

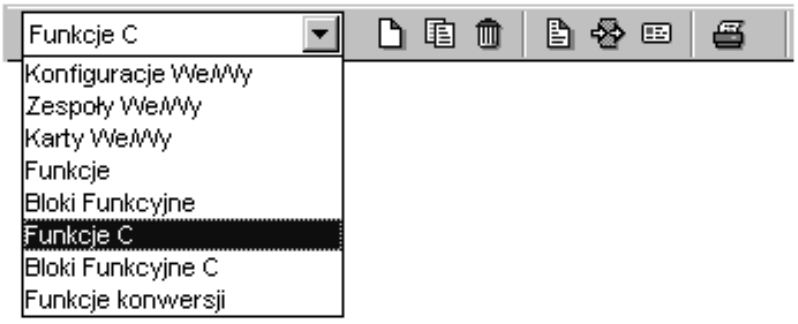
Składnia: **End**

Uwagi: Umieszczanie instrukcji "End" w ostatniej linii skryptu nie jest wymagane.

Przykład:
```
(*Skrypt ten wykonuje pętlę do momentu, gdy MyVar ma
wartość TRUE *)
   Loop:
If MyVar = FALSE Goto Continue
End
Continue:
   Print MyVar
Goto Loop
```

A.22 Korzystanie z Menedżera Biblioteki

Biblioteki ISaGRAF zapewniają standardowy interfejs pomiędzy oprogramowaniem automatyki a możliwościami programowymi czy sprzętowymi systemu oprogramowania wbudowanego ISaGRAF. Dla każdego typu interfejsu istnieje jedna biblioteka. Menedżer Biblioteki Pakietu ISaGRAF jest dedykowany dostawcom sprzętu lub inżynierom oprogramowania. Wykorzystują oni menedżera biblioteki do opisu interfejsu programowego IsaGRAF dla tworzonych przez siebie obiektów.
Menedżer Biblioteki pakietu ISaGRAF przedstawia elementy jednej z bibliotek ISaGRAF. Po lewej stronie okna znajduje się **lista elementów** wybranej biblioteki. Po prawej stronie, **opis** (podręcznik użytkownika) elementu wybranego aktualnie z listy elementów. Menu Menedżera Biblioteki zawiera komendy pozwalające tworzyć, definiować lub modyfikować elementy aktywnej biblioteki. Komenda "**Plik / Inna biblioteka**" pozwala na wybór jednej z bibliotek ISaGRAF. Pole zbiorcze po lewej stronie paska narzędzi może również służyć do wyboru biblioteki:



A.22.1 Zarządzanie elementami biblioteki

Komend menu "**Plik**" używa się do tworzenia elementów i opracowywania tych już istniejących w otwartej bibliotece

Tworzenie nowego elementu

Komenda "**Nowy**" w menu "**Plik**" tworzy w wybranej bibliotece nowy element. Nazwa nowego elementu jest wprowadzana w oparciu o następujące zasady nazewnictwa:

- maksymalna długość nazwy to **8** znaków
- pierwszy znak musi być **literą**

- kolejne znaki muszą być **literami**, **cyframi** lub znakiem '_'
- nadawanie nazw elementom biblioteki nie uwzględnia wielkości liter.

Z każdym elementem biblioteki jest powiązany komentarz tekstowy. Komentarz jest wprowadzany podczas tworzenia elementu. Przy tworzeniu nowego elementu, konieczne jest wprowadzenie:
- jego definicji dla konfiguracji We/Wy,
- jego parametrów dla karty We/Wy,
- jego interfejsu użytkownika dla funkcji lub bloku funkcyjnego.

Kiedy tworzona jest konwersja "C", funkcja "C" czy blok funkcyjny "C", to jest automatycznie generowana pełna struktura ich kodu źródłowego.

Opracowywanie istniejących elementów

Komenda "**Plik / Zmień nazwę**" pozwala użytkownikowi zmienić nazwę lub komentarz dla elementu wybranego z listy elementów. Komenda "**Plik / Kopiuj**" pozwala użytkownikowi skopiować element podświetlony w aktywnej bibliotece do innego elementu w tej samej bibliotece. Jeśli element docelowy już istnieje, cała jego zawartość zostaje zamieniona. Jeśli element docelowy nie istnieje, jest on automatycznie tworzony. Komenda "**Plik / Usuń**" usuwa zaznaczony aktualnie element z aktywnej biblioteki. Komendami "**Zmień nazwę**", "**Kopiuj**" i "**Usuń**" można operować na następujących komponentach elementu:
- opis techniczny
- pełna definicja konfiguracji We/Wy
- parametry dla kart i zespołów We/Wy
- definicje interfejsu dla funkcji i bloku funkcyjnego
- kod źródłowy dla funkcji i bloku funkcyjnego napisanych w języku IEC
- kod źródłowy dla konwersji, funkcji lub bloku funkcyjnego C

⚠ jeśli element jest konwersją "C", funkcją "C" lub blokiem funkcyjnym "C", jego nazwa nie jest automatycznie aktualizowana w dołączonym kodzie źródłowym po komendzie "**Zmień nazwę**" lub "**Kopiuj**".

⚠ jeśli element jest funkcją napisaną w języku IEC, nazwa parametru zwracanego nie zmienia się po komendzie "**Zmień nazwę**" lub "**Kopiuj**".

Ustawianie zabezpieczenia hasłem

Komenda "**Plik / Ustaw hasło**" pozwala użytkownikowi zdefiniować zabezpieczenie hasłem dla wybranych elementów w otwartej bibliotece. Dalsze informacje dotyczące poziomów haseł i zabezpieczenia danych znajdują się w sekcji "**Ochrona hasłem**" na końcu pierwszej części tego podręcznika. Hasła odnoszą się jedynie do zaznaczonych elementów. Nie mają one wpływu na inne elementy bibliotek ISaGRAF.

Kompilowanie funkcji i bloków funkcyjnych

Kiedy wybrana zostanie biblioteka funkcji lub bloków funkcyjnych napisanych w językach IEC, komenda "**Weryfikuj (kompiluj)**" w menu "**Plik**" jest wykorzystywana do sprawdzenia składni wybranych elementów i utworzenia ich kodu pośredniego. Funkcje i bloki napisane w językach IEC muszą zostać skompilowane bez błędów zanim będzie można je użyć w projektach ISaGRAF. Komenda ta nie jest skuteczna, jeśli wybrana zostanie inna biblioteka.

Opisy techniczne

Komenda "**Edycja / Opis**" pozwala użytkownikowi wprowadzić opis techniczny elementu wybranego w aktywnej bibliotece. Opis wprowadzany jest przy pomocy edytora tekstowego ISaGRAF. Opis elementu, to jego **poradnik dla użytkownika**. Użytkownik elementu będzie się do niego odnosił podczas konsolidacji projektu ISaGRAF. Opis dotyczący wykorzystania elementu powinien zawierać opis jego głównych funkcji, szczegółowe wyjaśnienie jego interfejsu programowego i parametrów oraz warunki i ograniczenia jego stosowania.

Komenda "**Narzędzia / Standardowy format**" pozwala użytkownikowi zdefiniować standardowy format tekstu dla wszystkich elementów wybranej aktualnie biblioteki. Podczas edycji opisu dla nowego elementu, format ten jest wykorzystywany jako model. Pozwala to użytkownikowi zoptymalizować edycję opisów.

Parametry

Parametry elementu opisują **interfejs** pomiędzy operacjami komputerowymi zapewnianymi przez element a zastosowaniem elementu w aplikacji ISaGRAF. Parametry posiadają inne znaczenie dla każdego elementu biblioteki.

Parametry konfiguracji We/Wy definiują pełny zestaw kart We/Wy dla konfiguracji oraz domyślne nazwy zmiennych stosowanych w kanałach We/Wy. Parametry karty We/Wy lub zespołu definiują fizyczną i logiczną konfigurację karty. Parametry funkcji lub bloku funkcyjnego definiują interfejs elementu, zgodnie z konwencjami nadawania nazw funkcjom języka ST. Funkcja konwersji nie posiada parametrów, ponieważ wykorzystuje standardowy wstępnie zdefiniowany interfejs.

Kod źródłowy

Pakiet ISaGRAF pozwala programiście zarządzać kodem źródłowym konwersji, funkcji lub bloku funkcyjnego biblioteki. Kod źródłowy funkcji lub bloku napisanego w języku IEC to tekst lub diagram opisany przy pomocy języka powiązanego z funkcją. Kod źródłowy komponentów "C" (funkcji "C", bloków funkcyjnych "C" oraz funkcji konwersji) podzielony jest na dwa odrębne pliki: **nagłówek kodu źródłowego**, który zawiera dokładną definicję **interfejsu**, zgodnie z definicją parametru elementu oraz plik **kodu źródłowego**, który zawiera implementację operacji elementu.

Pakiet ISaGRAF generuje plik kodu źródłowego kiedy tworzony jest nowy element biblioteki. W oparciu o definicję parametrów tworzy on również i aktualizuje nagłówek źródła. Programista może wykorzystać edytor tekstowy ISaGRAF do uzupełniania pliku kodu źródłowego.

Archiwizacja elementów biblioteki

Komendy menu "**Narzędzia / Archiwum**" uruchamia Menedżer Archiwizacji ISaGRAF pozwalający zapisywać i przywracać elementy biblioteki. Przed uruchomieniem "**Komendy archiwizacji**" trzeba najpierw wybrać bibliotekę. Menedżer Archiwizacji pokazuje listę elementów tylko jednej biblioteki w danej chwili.

A.22.2 Konfiguracje We/Wy

Biblioteka konfiguracji We/Wy ISaGRAF zapewnia prosty sposób inicjalizacji nowych projektów ISaGRAF o wstępnie zdefiniowanej konfiguracji We/Wy. Konfiguracja We/Wy definiuje:

- zestaw kart We/Wy
- wartości domyślne parametrów kart We/Wy
- domyślne nazwy kanałów We/Wy

Kiedy tworzony jest nowy projekt ISaGRAF z użyciem bibliotecznej konfiguracją We/Wy, automatycznie są ustawiane odpowiednie połączenia We/Wy, a zmienne We/Wy odpowiadające nazwom kanałów są deklarowane automatycznie w słowniku projektu.

Definicja konfiguracji We/Wy odbywa się za pomocą narzędzia połączeń We/Wy ISaGRAF (to samo narzędzie używane jest w ramach projektu). Dalsze informacje na temat wykorzystania tego narzędzia znajdują się w sekcji "Połączenia We/Wy" tego podręcznika. Podczas wstawiania do konfiguracji nowej karty We/Wy, wszystkie kanały nowej karty zostają zadeklarowane z domyślnymi nazwami. Domyślna nazwa kanału We/Wy posiada następujący format:

<direction („kierunek”)><typ><numer_gniazda>_<numer_kanału>

Pierwszy znak oznacza direction („kierunek”) kanału We/Wy:
"**I**" ..... kanał wejściowy
"**O**" ..... kanał wyjściowy

Drugi znak oznacza typ kanału We/Wy:
"**X**"..... binarny
"**D**"..... analogowy
"**M**" ..... komunikat

poniżej podano nazwy standardowych kanałów We/Wy:

IX0_7	binarny, wejściowy - karta #0 - kanał #7
QD2_4	integer, wyjściowy - karta #2 - kanał #4

Komenda "Podłącz kanały We/Wy" Edytora Połączeń We/Wy jest używana do zmiany domyślnej nazwy powiązanej z kanałem We/Wy.

A.22.3 Zespoły We/Wy

Wszystkie kanały pojedynczej karty posiadają ten sam typ (binarne, analogowe lub komunikat) i direction („kierunek") (wejściowe lub wyjściowe). Zespół We/Wy reprezentuje urządzenie We/Wy z kanałami o różnych typach lub kierunkach. Zespół We/Wy jest utożsamiany z listą pojedynczych kart We/Wy. Wykorzystuje on tylko jedno gniazdo listy połączeń na stojaku.

Aby zdefiniować zespół We/Wy użytkownik musi zdefiniować listę tworzących go pojedynczych kart We/Wy. Musi on również wprowadzić szczegółowe parametry każdej pojedynczej karty. Lista kart We/Wy wprowadzana jest za pośrednictwem okna dialogowego.

Naciśnięcie przycisku "**Dołącz**" pozwala użytkownikowi dodać pojedynczą kartę na końcu bieżącej listy. Przycisku "**Wstaw**" używa się do wstawienia nowej pojedynczej karty przed kartą wybraną aktualnie na liście. Przycisk "**Usuń**" usuwa zaznaczoną pojedynczą kartę z listy. Przyciski "**Zmień nazwę**" i "**Parametry**" używane są do zmiany nazwy i parametrów wybranej pojedynczej karty. Pełne wyjaśnienie parametrów pojedynczych kart znajduje się w następnej sekcji. Zespół We/Wy może zawierać do **16** pojedynczych kart We/Wy. Nazwa pojedynczej karty (w ramach zespołu We/Wy) nie może być dłuższa niż **8** znaków.

A.22.4 Karty We/Wy

Biblioteka kart We/Wy ISaGRAF definiuje standardowy interfejs pomiędzy zmiennymi aplikacji i sprzętem oprogramowania wbudowanego. Podczas opisywania aplikacji, wszystkie zmienne We/Wy są powiązane z kanałami kart We/Wy oprogramowania wbudowanego. Karta We/Wy ISaGRAF jest definiowana przy pomocy **nazwy** oraz "**kodu-klucza OEM**", który identyfikuje jej **dostawcę**. Inne parametry kart We/Wy opisują ich topologię (ilość kanałów, direction („kierunek") i typ kanałów) oraz ich konfigurację programową i sprzętową.

Parametry kart We/Wy

Istnieją dwa różne typy parametrów dla kart We/Wy: parametry wspólne, które definiowane są dla każdej karty z biblioteki ISaGRAF oraz parametry OEM charakterystyczne dla implementacji karty, zapewniane przez dostawcę sprzętu. Parametry wspólne wprowadzane są w górnej części okna definicji parametrów kart

We/Wy. Parametry te (wraz z nazwą karty We/Wy) identyfikują standardowy interfejs karty We/Wy.

"**Klucz-kod OEM**" to prosty numer, określający **dostawcę sprzętu**. Wszystkie karty definiowane przez jednego dostawcę muszą posiadać taki sam klucz-kod OEM. Klucz-kod OEM, to **16-bitowe słowo bez znaku** podawany w formacie **szesnastkowym**. Kodem-kluczem OEM zarezerwowanym dla **CJ International** jest "**1**".

Parametry główne określają topologię karty We/Wy. **Ilość kanałów** określa ilość dostępnych na karcie kanałów. **Typ** karty, to typ zmiennych, które mogą być powiązane z kanałami karty. **Direction („kierunek")** określa, czy zmienne powiązane z kartą, to zmienne **wejściowe** czy **wyjściowe**.

Uwaga: zmienne We/Wy o różnych typach lub kierunkach nie mogą znajdować się na jednej karcie We/Wy ISaGRAF. Funkcja taka wymaga wykorzystania zespołu We/Wy.

Parametry OEM

Parametry OEM wprowadzane są w dolnej części okna definiowania parametrów karty We/Wy. Parametry te określane są przez dostawcę sprzętowego karty We/Wy i są charakterystyczne dla każdej karty. Każda karta może posiadać co najwyżej **16** parametrów OEM. Karta może nie posiadać żadnych parametrów OEM. Menedżer Biblioteki ISaGRAF pozwala dostawcy sprzętu na określenie identyfikacji i formatu każdego z parametrów oraz sposobu, w jaki są one wprowadzane przez programistę automatyki.

Okno po lewej zawiera listę parametrów OEM. Każdy parametr identyfikowany jest przy pomocy **nazwy** i **numeru** logicznego, od **0** do **15**. Obszar po prawej zawiera szczegółowy opis parametru wybranego z listy. Wyboru parametru z listy dokonuje się w celu uzyskania dostępu do jego pełnego opisu. Naciśnięcie przycisku "**Czyść**" kasuje opis parametru i usuwa parametr z listy parametrów. Uwaga: komendy tej nie można "cofnąć".

Nazwa parametru jest stosowana do identyfikacji odpowiadającego mu pola wprowadzania danych podczas łączenia karty We/Wy, jeśli pole to musi zostać określone przez operatora automatyki. Nazwa parametru musi być zgodna z następującymi zasadami:

- maksymalna długość nazwy to **16** znaków
- pierwszy znak nazwy musi być **literą**
- kolejne znaki muszą być **literami**, **cyframi** lub znakiem '_'

Typ parametru określa **format wewnętrzny** parametru oraz **format wprowadzania danych** podczas połączenia We/Wy aplikacji. Oto lista dostępnych formatów wewnętrznych:

word ..... 16-bitowe słowo bez znaku
long ..... 32-bitowe słowo bez znaku
word hexa ..... 16-bitowe szesnastkowe słowo bez znaku
long hexa ..... 32-bitowe szesnastkowe słowo bez znaku

boolean ..... 16-bitowe słowo bez znaku (używany tylko najniższy bit)
character..... 16-bitowe słowo bez znaku (używany tylko najniższy bit)
string..... tablica 16 bajtów, zawierająca ciąg (łańcuch) zakończony znakiem null (pustym)
float ..... 32-bitowa liczba zmiennoprzecinkowa o pojedynczej precyzji

Poniżej przedstawiono dostępne formaty wprowadzania danych:
word ..... dziesiętne słowo bez znaku
long ..... dziesiętne słowo długie
word hexa ..... szesnastkowe słowo bez znaku
long hexa ..... szesnastkowe słowo długie bez znaku
boolean ..... „prawda" lub „fałsz"
character..... pojedynczy znak
string..... ciąg znaków ASCII (maksimum 15 znaków)
float ..... liczba zmiennoprzecinkowa pojedynczej precyzji

Okno "**dostęp**" jest stosowane do określania sposobu, w jaki użytkownik końcowy możne uzyskać dostęp do parametru. Jeśli ustawiona jest opcja "**Zdefiniowany użytkownik**", parametr przedstawiany jest jako pole wprowadzania danych podczas połączenia karty We/Wy. Wartość domyślna parametru OEM jest wykorzystywana domyślnie podczas edycji parametru. Jeśli ustawiona jest opcja "**Ukryty**", parametr jest stałą i nie figuruje w oknie połączenia karty We/Wy. Wartość domyślna parametru OEM określa wartość parametru stałego. Opcja "**Tylko do odczytu**" wskazuje, iż parametr jest widoczny dla użytkownika, lecz nie może być modyfikowany. Wartość domyślna jest używana jako wartość stała.

A.22.5 Funkcje i bloki napisane w językach IEC

ISaGRAF obsługuje bibliotekę funkcji i bloków funkcyjnych napisanych w językach IEC. Języki dostępne do opisu takiej funkcji lub bloku funkcyjnych to **FBD** (Diagram Bloków Funkcyjnych), **LD** (Diagram Drabinkowy), **ST** (Tekst Strukturalny) lub **IL** (Lista Instrukcji). Proszę zauważyć, iż języki LD i FBD można wykorzystywać w tym samym schemacie. Język **SFC** (Sekwencyjny Graf Funkcji) nie może być wykorzystany do opisu funkcji czy bloków funkcyjnych w bibliotece. Język powiązany z elementem biblioteki wybierany jest przy tworzeniu funkcji i nie może później ulec zmianie.

Kompilowanie

Funkcje i bloki definiowane w bibliotece muszą zostać skompilowane (zweryfikowane) zanim będzie można wykorzystać je w projekcie ISaGRAF. Jeśli chodzi o bibliotekę, nie trzeba zmieniać niczego poza tym w funkcjach czy blokach funkcyjnych. Elementy biblioteki pojawią się automatycznie w oknie menu wyboru przy używaniu edytorów graficznych LD/FBD w ramach projektu.

Funkcja zdefiniowana w bibliotece może odwoływać się do innych funkcji biblioteki. Jednak system ISaGRAF **nie obsługuje rekursywności** w wywołaniach funkcji. Blok funkcyjny napisany w języku IEC nie może wywoływać innego bloku funkcyjnego (ani w języku IEC ani "C").

Wprowadzanie kodu źródłowego

Kod źródłowy funkcji lub bloku funkcyjnego biblioteki jest wprowadzany przy użyciu standardowych narzędzi ISaGRAF: edytora graficznego dla programów LD lub FBD, edytora tekstowego dla programów ST lub IL. Dalsze informacje na temat tych narzędzi znajdują się w odpowiednich sekcjach tego podręcznika. Generator Kodu ISaGRAF może być wywoływany bezpośrednio z graficznego lub tekstowego okna edycyjnego w celu kompilacji kodu źródłowego funkcji lub bloku biblioteki.

Słownik ze zmiennymi lokalnymi

Funkcja lub blok funkcyjny biblioteki mogą posiadać zmienne lokalne i lokalne słowa zdefiniowane. Aby uzyskać dostęp do deklaracji zmiennych, użytkownik musi skorzystać z komendy "**Słownik**" w menu "**Plik**" w oknie edytora w czasie edycji kodu źródłowego funkcji.

Funkcja lub blok funkcyjny biblioteki nie może korzystać ze zmiennej globalnej czy instancji bloku funkcyjnego. Zmienne lokalne funkcji należy inicjalizować w treści funkcji.

Zmienne lokalne bloku funkcyjnego napisanego w języku IEC są kopiowane (wywoływane) przy każdym użyciu bloku w projekcie. Zmienne lokalne instancji zachowują swoje wartości z jednego wywołania na drugie.

Określanie interfejsu

Funkcje lub bloki funkcyjne mogą posiadać do **32** parametrów (wejściowych lub wyjściowych). Funkcja posiada zawsze jeden (i tylko jeden) parametr zwracany, który musi posiadać taką samą nazwę jak funkcja, aby być zgodnym z konwencjami zapisu języka ST.

Lista w lewym górnym obszarze okna przedstawia parametry w kolejności ich wywoływania: najpierw parametry wywołania, na końcu parametry zwracane. Część dolna okna przedstawia szczegółowy opis parametru wybranego aktualnie z listy. Każdy z typów danych ISaGRAF może być wykorzystany jako parametr. Parametry zwracane muszą być umieszczone na liście po parametrach wywołania. Nadawanie parametrom nazw musi być zgodne z następującymi zasadami:

- długość nazwy nie może przekraczać 16 znaków
- pierwszy znak musi być literą
- kolejne znaki muszą być literami, cyframi lub znakiem podkreślenia
- nadawanie nazw nie uwzględnia wielkości liter

Komenda "**Wstaw**" jest stosowana do wstawiania nowego parametru przed parametrem wybranym. Komenda "**Usuń**" jest stosowana do usuwania zaznaczonego parametru. Komenda "**Ułóż**" automatycznie układa (sortuje) ponownie parametry tak, iż parametry zwracane są umieszczane na końcu listy.

A.22.6 Funkcje i bloki funkcyjne "C"

Funkcje i bloki funkcyjne "C" to **funkcje komputera**, wywoływane z aplikacji automatyki, zgodnie z interfejsem wywoływania funkcji języka ST.

Funkcje to procesy **synchroniczne**. Aplikacja oprogramowania wbudowanego ISaGRAF jest wstrzymywana podczas wykonywania funkcji. Bloki funkcyjne łączą operacje z ukrytymi danymi statycznymi. Na przykład, blok funkcyjny "licznika" reprezentuje operację liczenia, jak również wynik liczenia. Funkcje i bloki funkcyjne można wykorzystać do uzupełnienia możliwości standardowego języka automatyki lub do uzyskania dostępu do zasobów systemowych.

Okno definicji parametrów jest używane do definiowania nazwy i typu każdego parametru wywołania czy zwracanego funkcji lub bloku funkcyjnego. Komendy menu "**Edycja**" używane są do definiowania parametrów wybranej funkcji lub bloku funkcyjnego. Funkcja może posiadać do **31** parametrów wywołania oraz posiada zawsze **jeden** parametr zwracany. Blok funkcyjny może posiadać do **32** parametrów, będących dowolną kombinacją parametrów wywołania i zwracanych. Poniżej przedstawiono odpowiedniki typów ISaGRAF i typów "C":

BINARNA	długa bez znaku	32-bitowe słowo bez znaku: 1=prawda / 0=fałsz
ANALOGOWA	długa	32-bitowe słowo typu integer ze znakiem
REAL	zmiennoprzecinkowa precyzji	wartość zmiennoprzecinkowa pojedynczej
TIMER	długa bez znaku	32-bitowe słowo integer bez znaku (jednostka to 1 ms)
KOMUNIKAT	znak *	ciąg znaków.

Kiedy wartość komunikatu zostaje przekazana do funkcji lub bloku funkcyjnego "C", nie może on zawierać znaków pustych. Ciąg przekazywany do kodu "C" jest zakończony znakiem null (pustym).

Dalsze informacje o zarządzaniu kodem źródłowym „C” dla funkcji lub bloku funkcyjnego oraz o konsolidacji nowych elementów z systemem oprogramowania wbudowanego ISaGRAF podano w Podręczniku Użytkownika Oprogramowania Wbudowanego ISaGRAF.

A.22.7 Funkcje konwersji

Funkcja konwersji to funkcja "C" wywoływana przez menedżera We/Wy ISaGRAF po każdym wprowadzeni do projektu lub wyprowadzeniu z projektu zmiennych analogowych wykorzystujących tę konwersję.

Funkcja ta tworzy powiązanie pomiędzy **wartością elektryczną** zmiennej (odczytywaną z czujnika wejściowego lub wysyłaną do urządzenia wyjściowego) a jej **wartością fizyczną** (wykorzystywaną w wyrażeniach aplikacji). Dlatego też funkcja ta podzielona jest na dwie części: konwersję wejściową i konwersję wyjściową. Menedżer Biblioteki ISaGRAF umożliwia użytkownikowi kontrolowanie kodu źródłowego "C" funkcji konwersji.

Konwersję można zastosować do zmiennej analogowej typu **integer** lub **real**. Znaczy to, że interfejs funkcji konwersji jest zawsze definiowany przy pomocy wartości zmiennoprzecinkowych. Interfejs jest taki sam dla każdej funkcji konwersji. Definicja "C" takiego interfejsu odbywa się w pliku definicji "**TACN0DEF.H**".

Dalsze informacje o zarządzaniu kodem źródłowym „C” dla funkcji konwersji oraz o konsolidacji nowych elementów w systemie oprogramowania wbudowanego ISaGRAF podano w Podręczniku Użytkownika Oprogramowania Wbudowanego ISaGRAF.

A.23 Używanie programu archiwizacji

Program archiwizacyjny ISaGRAF umożliwia użytkownikowi zapisywanie projektów i bibliotek ISaGRAF na dyskietkach lub w katalogu z kopiami zapasowymi. Menedżer archiwizacji ISaGRAF to okno dialogowe, które można wywołać z programu Zarządzanie Projektami lub Biblioteki ISaGRAF.

⚠ Aby utworzyć i utrzymać niezawodne archiwum, zalecamy stosowanie się do poniższych wskazówek:

- Wpisz nazwę i opis zapisanego obiektu na naklejce dysku
- Nie zapisuj projektów i bibliotek na tej samej dyskietce
- Nie zapisuj różnych projektów na tej samej dyskietce

A.23.1 Wywoływanie menedżera archiwizacji

Okno dialogowe "Archiwum" można wywołać z menu "**Narzędzia / Archiwum**" w oknie Zarządzanie Projektami, w celu zapisania lub odtworzenia projektu oraz wspólnych danych:

Okno dialogowe "Archiwum - " można również uruchomić za pomocą komendy "**Narzędzia / Archiwum**" Menedżera Biblioteki ISaGRAF w celu zapisania lub odtworzenia elementów biblioteki wybranej w danym momencie w oknie Biblioteki:

- ***Projekty***

Projekt jest zawsze zapisywany w całości. Wszystkie komponenty projektu (pliki źródłowe programu, kod pośredni i kod wykonywalny aplikacji) są zapisywane razem w jednym pliku-archiwum. Wybranie opcji "**Kompresja**" redukuje rozmiar archiwum projektu.

- ***Elementy bibliotek***

Elementy bibliotek ISaGRAF mogą być zapisywane pojedynczo. Wszystkie komponenty elementów biblioteki (opis, definicja, interfejs, kod źródłowy, itd.) są zapisywane razem, w jednym pliku-archiwum.

- ***Wspólne dane***

Komenda "**Narzędzia / Archiwum / Wspólne dane**" w oknie Zarządzanie Projektami pozwala użytkownikowi wykonać kopie zapasowe lub odtworzyć dane używane przez różne aplikacje. Komenda ta nie obsługuje bibliotek ISaGRAF.

Poniżej znajduje się lista plików, które mogą zostać przekopiowane dzięki tej komendzie:

common.eqv wspólne zdefiniowane słowa
oem.bat zdefiniowany przez użytkownika plik wsadowy MS-DOS

Pliki te są zapisywane po jednym na dysk-archiwum w formie oryginalnej. Odpowiadające im pliki archiwizacyjne nigdy nie są kompresowane.

A.23.2 Opcje

Ścieżka wykorzystana do katalogu archiwum ISaGRAF wyświetlona jest w dolnej części okna dialogowego. Naciśnij przycisk "Przeglądaj", aby przeszukać dyski i znaleźć inny dysk i katalog archiwum.

Po ustawieniu opcji "**Kompresja**", wszystkie pliki archiwizowane podczas wykonywania procedury "**Kopia zapasowa**" są kompresowane. Opcja ta jest bardzo przydatna do zmniejszenia rozmiarów dużego pliku-archiwum projektu i jego zapisania na jednej dyskietce. Kompresja archiwum nie jest z reguły wymagana dla komponentów biblioteki. Menedżer Archiwizacji ISaGRAF automatycznie rozpoznaje status pliku-archiwum (czy jest skompresowany) przy jego odtwarzaniu. Oznacz to, że opcja "**kompresji**" nie ma wpływu na procedurę "**Przywróć z kopii**".

A.23.3 Kopia zapasowa i przywracanie danych z kopii

Lista "**Projekty pakietu ISaGRAF**" (po lewej) pokazuje obiekty istniejące w pakiecie ISaGRAF zainstalowanym na twardym dysku. Lista "**Archiwum** " (po prawej) pokazuje obiekty zapisane na wskazanym dysku i w katalogu archiwum.

- ***Kopia zapasowa***

Zapisania obiektu w archiwum dokonuje się przez zaznaczenie obiektu z listy po **lewej** (obiekty pakietu ISaGRAF) i naciśnięcie przycisku "**Kopia zapasowa**". Można

zaznaczyć więcej niż jeden obiekt z listy. Przycisk "**Kopia zapasowa**" jest niedostępny, kiedy zaznaczony zastanie element listy po **prawej** (dostępny dla „przywracania z kopii zapasowej").

Przywracanie z kopii zapasowej

Kopiowanie obiektów z archiwum do pakietu ISaGRAF odbywa się poprzez zaznaczenie obiektu z listy po **prawej** (obiekty archiwum) i naciśnięcie przycisku "**Przywróć z kopii**". Można zaznaczyć więcej niż jeden obiekt z listy. Przycisk "**Przywróć z kopii**" jest niedostępny, kiedy zaznaczony zostanie element z listy po **lewej** (dostępny dla tworzenia kopii zapasowej).

A.23.4 Pliki archiwalne

Menedżer Archiwizacji ISaGRAF tworzy osobny plik-archiwum dla każdego zapisanego obiektu. Plik-archiwum posiada taką samą nazwę, jak obiekt. Rozszerzenie pliku wskazuje na jego typ. Poniżej są podane używane rozszerzenia:

.pia ..... projekt
.bia ..... karta We/Wy
.iia ..... funkcja w języku IEC
.aia ..... blok funkcyjny w języku IEC
.uia ..... funkcja C
.fia ..... blok funkcyjny C
.cia ..... funkcja konwersji C
.ria ..... konfiguracja We/Wy
.xia ..... zespół We/Wy

A.24 Drukowanie całej dokumentacji

Generator Dokumentów ISaGRAF pozwala użytkownikowi budować i drukować kompletne dokumenty wybranych projektów. Można go wywołać komendą "**Projekt / Drukuj**" z okna Zarządzanie Projektami lub Programy i wydrukować całą dokumentację. Można uruchomić Generator Dokumentów przy pomocy komendy "**Drukuj**" wszystkich innych edytorów ISaGRAF w celu wydrukowania zawartości pojedynczego dokumentu ISaGRAF. W obu przypadkach Generator Dokumentów posiada takie same funkcje.
Komendy menu "**Edycja**" są wykorzystywane do określania, które z elementów projektu mają zostać wstawione do dokumentu. W ten sposób użytkownik tworzy "spis treści" żądanego dokumentu. Wszelkie informacje dotyczące projektu (programy, zmienne, opcje, konfiguracja We/Wy, itp.) mogą zostać wstawione do dokumentu projektu. W dokumencie tym nie mogą znaleźć się informacje z innego projektu ani z bibliotek ISaGRAF.

Komenda "**Plik / Drukuj**" generuje dokument i wysyła go do drukarki, zgodnie z określoną zawartością. Budowanie i formatowanie dokumentacji do zadania "Drukuj" może zająć kilka minut. Wysoce zalecane jest odczekanie do momentu ukończenia "zadania drukowania" w oknie Generatora Dokumentów ISaGRAF przed wywołaniem innych komend pakietu ISaGRAF. Budowa całości dokumentu może wymagać dużej ilości wolnego miejsca na twardym dysku. Jeśli dysk zostanie zapełniony, zostanie wyświetlony komunikat o błędzie. W takim przypadku użytkownik powinien zwolnić część miejsca na twardym dysku poprzez usunięcie plików, lub zredukować zawartość dokumentacji. Po wybraniu komendy "**Drukuj**" pojawia się okno dialogowe. Pozwala ono użytkownikowi na wprowadzenie opisu samej komendy drukowania. Opisy te przechowywane są w pliku historii i zostaną wydrukowane na pierwszej stronie każdego dokumentu w przyszłości (łącznie z bieżącym).

A.24.1 Dostosowywanie spisu treści

Menu "**Edycja**" zawiera komendy określające "spis treści" dokumentu. Szereg komend pozwala użytkownikowi wybrać spis domyślny (zawierający wszystkie komponenty projektu), opracować spis specjalny (zawierający jedynie niektóre komponenty) lub przemieszczać elementy spisu i modyfikować je.

Lista domyślna

Komenda "**Lista domyślna**" w menu "**Edycja**" definiuje standardowy spis treści dla dokumentu, który zawiera wszystkie komponenty projektu. Spis standardowy składa się z:

- Opisu projektu

- Drzewa hierarchii (połączenia pomiędzy programami)
- Kodu źródłowego dowolnego programu
- Pliku dziennika dowolnego programu
- Definicji wspólnych
- Definicji globalnych
- Lokalnych definicji dowolnego programu
- Zmiennych globalnych
- Zmiennych lokalnych dowolnego programu
- Opcji aplikacji
- Połączenia We/Wy
- Listy zmiennych
- Tablic konwersji
- Odsyłaczy skróconych
- Odsyłaczy szczegółowych
- Podsumowania deklaracji
- Mapy adresów sieciowych
- Historii modyfikacji

Spis treści można zapisać przy użyciu komendy "**Plik / Zapisz**". Komenda ta jest niedostępna kiedy generator dokumentów zostanie uruchomiony z poziomu edytora ISaGRAF w celu wydrukowania jednego dokumentu.

Wycinanie i wklejanie

Komend "**Edycja / Wytnij**" oraz " **Edycja / Wklej**" używa się do przemieszczania elementów na liście w celu dopasowania uporządkowania spisu do określonych potrzeb. Generator Dokumentów pozwala na wielokrotny wybór, więc wycinane i wklejane mogą być również grupy elementów.

Czyszczenie spisu

Komendy "**Edycja / Wyczyść**" używa się do kasowania spisu treści tak, żeby można było ją całkowicie przebudować wstawiając pojedyncze elementy.

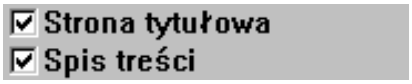
Wstawianie elementów do spisu

Po wywołaniu komendy " **Edycja / Wstaw**", pojawia się okno dialogowe "**Dodaj pozycję**". Pozwala ono użytkownikowi wstawiać do spisu treści elementy (komponenty projektu).

Dla elementów powiązanych z programem należy wywołać okno zbiorcze "**Program**", aby wybrać nazwę programu. Aby wstawić wybrany element do spisu treści, należy nacisnąć przycisk "**Dodaj**". Ten sam element może wystąpić w spisie treści tylko raz.

A.24.2 Opcje

Komendy menu "**Opcje**" są używane do definiowania i dostosowywania formatu generowanego dokumentu. Inne opcje są dostępne bezpośrednio poprzez przyciski okna Generatora Dokumentów:



Jeśli ustawiona jest opcja "**Strona tytułowa**", na początku dokumentu drukowana jest strona nagłówkowa, zawierająca tytuł projektu i historię wydruków. Kiedy opcja ta jest odznaczona, pierwszy element jest drukowany na pierwszej stronie.
Kiedy ustawiona jest opcja "**Spis treści**", to na końcu wygenerowanego dokumentu drukowany jest spis treści.
Przy początkowym uruchomieniu Generatora Dokumentów przy pomocy komendy "**Drukuj**" jednego z edytorów ISaGRAF (programu, słownika, itp.), obie te opcje są odznaczone.

Grafy SFC

Opcja "**Oddziel poziomy SFC**" nakazuje systemowi wydrukowanie najpierw 1 poziomu SFC (graf i komentarze), a następnie kodu poziomu 2 dla każdego programu SFC. Kiedy opcja ta nie jest zaznaczona, poziomy 1 i 2 pojawiają się razem na jednym wydruku.

Format strony

Komenda "**Ustawienia strony**" w menu "**Opcje**" jest używana do zdefiniowana parametrów działania Generatora Dokumentów podczas formatowania strony. Określone mogą zostać następujące parametry:

- **Lewy margines**: (1 lub 2 centymetry albo bez marginesu)
- **Ramka strony**: Kiedy opcja ta jest zaznaczona wokół każdej drukowanej strony rysowane jest obramowanie.

Szablon tytułu strony

Komenda " **Tytuł strony**" w menu "**Opcje**" jest używana do określania zawartości tabeli tytułowej, drukowanej w dolnej części każdej strony. Standardowy format takiej tabelki jest następujący:

Pierwsza linia tytułu głównego (zawierająca nazwę projektu ISaGRAF), aktualna data i numer strony są generowane automatycznie przez Menedżera Dokumentów i nie mogą być zmienione.
Trzy linie tekstu po lewej stronie tabelki (text1, text2, text3) oraz druga linia tytułu głównego są definiowane przez użytkownika. Użytkownik może również zmienić logo drukowane w lewej części tabelki. Aby wprowadzić inne logo, użytkownik musi określić ścieżkę dostępu do pliku mapy bitowej (**.BMP**). Plik ten może posiadać dowolne rozmiary. Zostanie on rozciągnięty lub zmniejszony w zależności od rzeczywistych rozmiarów drukowanej strony. Kliknięcie obszaru logo w oknie dialogowym spowoduje wyświetlenie nowo wybranego obrazu. W chwili uruchomienia komendy "**Drukuj**", plik z obrazem musi znajdować się na dysku (we wskazanym katalogu i posiadać określoną nazwę).

Wybór czcionek

Komendy "**Czcionka tekstu**" i "**Czcionka tytułu**" menu "**Opcje**" służą do określania czcionki znakowej, używanej podczas drukowania tekstu oraz tytułów każdego elementu dokumentu. Wielkość i styl znaków tekstu i tytułów może również zostać określony. Wybór czcionki odbywa się przy pomocy standardowego okna określanego przez system Windows. Każdy tekst (programy tekstowe, nazwy na schematach, itp.) zostanie wydrukowany znakami o wybranej wielkości, stylu i czcionce. Tylko tytuły zostaną wydrukowane czcionką wybraną dla tytułów.
Jeśli typ czcionki znaków nie jest określony, dla każdego tekstu zostaną użyte standardowe czcionki drukarki o następujących stylach:

- Styl "Normalny" dla tekstu i nazw na grafach
- Styl "Pogrubiony" dla tytułów

A.25 Ochrona hasłem

Pakiet ISaGRAF zawiera cały system ochrony danych, pozwalający użytkownikowi na ochronę projektów i elementów bibliotek przy pomocy hasła. Elementem biblioteki może być konfiguracja We/Wy, karta We/Wy lub zespół We/Wy, funkcja lub blok funkcyjny napisany w językach IEC, funkcja, blok funkcyjny lub funkcja konwersji "C". Baza danych ochrony hasłem jest przeznaczona dla jednego projektu lub elementu biblioteki i nie może być udostępniana kilku projektom lub elementom biblioteki.

Poziomy zabezpieczeń

W ramach jednego projektu lub elementu biblioteki, użytkownik może zdefiniować do **16** poziomów dostępu, odpowiadających różnym hasłom. Poziomy dostępu posortowane są według systemu hierarchicznego. Posiadają numery od **0** do **15**. Najwyższy poziom dostępu posiada numer **0**. Jeśli użytkownik zna hasło, może uzyskać dostęp do wszystkich elementów chronionych w ramach odpowiadającemu mu poziomu dostępu oraz do tych, które chronione są niższymi poziomami. Każda podstawowa komenda lub dane projektu lub elementu biblioteki mogą być osobno chronione przy pomocy poziomu dostępu. Na przykład, komenda "Utwórz kod aplikacji" z menu ISaGRAF może być chroniona osobno. Podstawowe dane mogą być programem, listą opcji, uwagą techniczną do elementu biblioteki itp...

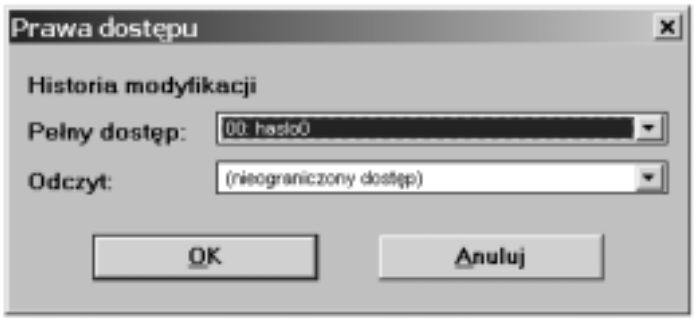
Definiowanie ochrony hasłem

Komenda "**Ustaw hasło**" z menu ISaGRAF wykorzystywana jest do definiowania haseł i poziomów dostępu dla jednego projektu lub jednego elementu biblioteki. Komenda ta wywoływana jest z menu Menedżera Projektów ISaGRAF (w przypadku projektu) lub Menedżera Bibliotek ISaGRAF(w przypadku elementu biblioteki). Przy uruchamianiu tej komendy po raz pierwszy nie jest wymagane żadne hasło. Jeśli hasła zostały już zdefiniowane, to przed uzyskaniem dostępu do tej komendy użytkownik musi wprowadzić hasło najwyższego znanego mu poziomu. Hasła i elementy chronione wyższego poziomu nie mogą być wtedy modyfikowane. Komenda "**Ustaw hasło**" umożliwia użytkownikowi zdefiniowanie haseł dopowiadających różnym poziomom dostępu i chronienie komend i danych podstawowych na zdefiniowanych poziomach.

Hasła (odpowiadające poziomom ochrony) wprowadza cię poprzez dwukrotne kliknięcie linii listy górnej. Do wprowadzania hasła służy następujące okno.

Lista w części dolnej zawiera różne elementy (dane lub funkcje), które mogą być chronione oraz przyporządkowany do nich aktualnie poziom zabezpieczeń: uprawnienie do dostępu: „Odczyt” " albo „Pełne". Wyznaczenie poziomu ochrony jako uprawnienia do "odczytu" pozwala zapobiec nawet otwarciu lub wydrukowaniu dokumentu przez użytkowników nie posiadających wystarczającego uprawnienia.

Dwukrotne kliknięcie linii listy dolnej pozwala ustawić uprawnienia dotyczące wybranego elementu lub danych. Otwiera się następujące okno:



Obydwa uprawnienia mogą zostać ustawione zarówno jako "dostęp nieograniczony", jak i na poziom ochrony zdefiniowany przez hasło. Uprawnienie pełnego dostępu nie może zostać przyporządkowane do poziomu o priorytecie niższym niż poziom przeznaczony do odczytu.

Należy zauważyć, że w pewnych dokumentach takich, jak opis projektu, wyraźnie widocznych w pakiecie ISaGRAF, dostęp do odczytu nie może być zabezpieczony hasłem.

Dostęp do danych chronionych

Do uruchomienia pakietu ISaGRAF nie jest wymagane żadne hasło ani nazwa użytkownika. Za każdym razem, gdy użytkownik chce uzyskać dostęp do chronionych danych lub funkcji, musi wpisać w oknie dialogowym wymagane hasło.

Jeśli użytkownik wprowadzi wymagane hasło (lub hasło przyporządkowane do wyższego poziomu dostępu), może normalnie kontynuować pracę. Za każdym razem, gdy użytkownik wprowadza hasło, zostaje ono zachowywane w pamięci tak, aby później nie musiał wprowadzać go ponownie. Zapamiętane hasła są sprawdzane za każdym razem, gdy narzędzie ISaGRAF jest uruchamiane z innego narzędzia ISaGRAF (na przykład, Menedżer Projektów uruchamia Menedżera Programów). Zapamiętane hasła zostają utracone po zamknięciu ostatniego pozostałego okna ISaGRAF. Hasła wprowadzone podczas edycji projektu albo przy użyciu Menedżera Bibliotek czy też Menedżera archiwizacji nie mogą być udostępniane. Jeśli użytkownik wprowadzi nieprawidłowe hasło, nie będzie mógł uruchomić wybranej funkcji.

Powiązania z Menedżerem archiwizacji

Przy zapisywaniu obiektu (projektu lub elementu biblioteki) na dysku-archiwum, wywoływany jest element zabezpieczenia o nazwie "**Tworzenie kopii zapasowej w archiwum**". Odpowiada on systemowi ochrony danych przyporządkowanemu do obiektu w pakiecie ISaGRAF (dysk twardy). W już istniejącym systemie ochrony danych obiektu na dysku-archiwum nie jest przeprowadzany żaden test. Komenda "**Wykonaj kopię zapasową**" Menedżera Archiwizacji ISaGRAF zapisuje na dysku-archiwum obiekt wraz z informacjami dotyczącymi ochrony danych.

Podczas przywracania z kopii zapasowej obiektu już istniejącego w pakiecie ISaGRAF (dysk twardy), wywoływany jest element ochrony danych o nazwie "**Nadpisz z archiwum**". Odpowiada on systemowi ochrony danych skojarzonemu z obiektem pakiecie (dysk twardy). W systemie ochrony danych obiektu na dysku zawierającym -archiwum nie jest przeprowadzany żaden test. Jeśli sprawdzana jest poprawność tej komendy, odzyskane informacje dotyczące ochrony danych zastąpią te istniejące na dysku twardym.

Ustawianie indywidualnej ochrony dla zmiennych i kanałów We/Wy

Pakiet ISaGRAF zapewnia pełny system ochrony danych oparty na hierarchii haseł. Deklaracja zmiennych i połączeń We/Wy może być chroniona globalnie za pomocą hasła. Ponadto ISaGRAF pozwala ustawiać indywidualne zabezpieczenia dla dowolnej zmiennej lub kanału We/Wy. Zakłada się, iż:

- hasła zostały już zdefiniowane w systemie definicji haseł (zastosuj komendę "**Projekt / Ustaw hasło**" z okna Zarządzanie Projektami) tak, aby poziomy zabezpieczeń były dostępne dla indywidualnej ochrony.

- używa się do ochrony indywidualnej poziomu zabezpieczenia o priorytecie wyższym niż globalne zabezpieczenie używa się zmiennych lub We/Wy.

Gdy zmienna lub kanał We/Wy są objęte ochroną indywidualną, to obok nazwy w słowniku lub w oknie połączeń We/Wy pokazuje się niewielka ikona.

Komendy "**Ustaw zabezpieczenie**" oraz "**Usuń zabezpieczenie**" w menu "**Edycja**" w Słowniku lub oknie Konfiguratora We/Wy służą do ustawiania i usuwania zabezpieczenia indywidualnego wybranej zmiennej lub kanału. Do skojarzenia tej zmiennej lub kanału z poziomem zabezpieczenia obie komendy wymagają wprowadzenia prawidłowego hasła. Następnie każdorazowo przy próbie zmiany zmiennej lub połączenia do kanału objętych zabezpieczeniem indywidualnym, konieczne jest wprowadzenie hasła dostatecznego poziomu.

Uwaga: jeśli zmienna lub kanał są zabezpieczone na pewnym poziomie, a odpowiadające hasło zostanie usunięte z systemu zabezpieczeń i nie zostało zdefiniowane żadne hasło wyższego poziomu, to zmienna lub kanał nie będą mogły być zmieniane, chyba, że zostanie zdefiniowane nowe hasło dostatecznego poziomu.

A.26 Zaawansowane techniki programowania

W rozdziale podano bliższe informacje na temat pakietu ISaGRAF i systemu oprogramowania sterownika. Zaleca się, aby przed przeczytaniem niniejszej sekcji użytkownik zapoznał się z narzędziami i metodami ISaGRAF.

A.26.1 Bliższe informacje na temat narzędzi ISaGRAF

Używając narzędzi edycyjnych ISaGRAF, użytkownik może nacisnąć **prawy przycisk myszy**, aby otworzyć menu podręczne, zawierające główne komendy edycji. Menu to zostaje otwarte w miejscu, w którym w danej chwili znajduje się kursor. Jest ono bardzo przydatne w komendach wycinania i wklejania, zmniejsza bowiem konieczność operowania myszą.

Narzędzia ISaGRAF mogą być **uruchomione wielokrotnie** w danej chwili. Nie można jednak otworzyć danego narzędzia dwa razy do edycji tego samego dokumentu, możliwe jest otwarcie odrębnych okien z tym samym narzędziem i równoległa edycja różnych obiektów.

Dostępne są również inne komendy, pozwalające odnaleźć informacje dotyczące przycisków graficznych na paskach narzędzi. Aby wyświetlić zawartość paska narzędzi w formie menu podręcznego, należy dwukrotnie kliknąć pusty obszar tego paska. Przytrzymanie kursora myszy nad dowolnym przyciskiem graficznym powoduje wyświetlenie odpowiadającej mu komendy tekstowej.

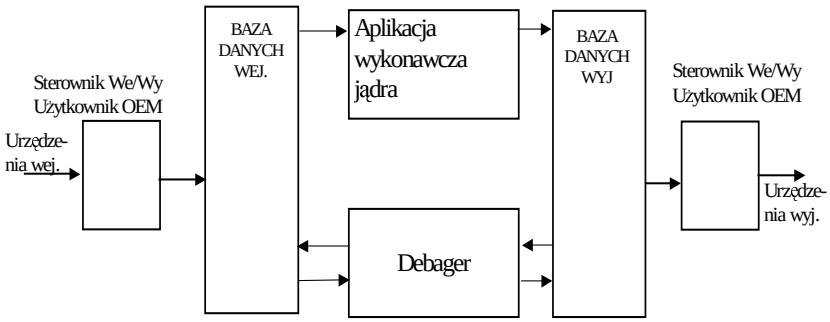
A.26.2 Zablokowane We/Wy i wirtualne We/Wy

Zdefiniowanie karty We/Wy jako **wirtualnej** wyłącza przetwarzanie fizycznych kanałów We/Wy. Kiedy karta jest zdefiniowana jako wirtualna, operacje jądra ISaGRAF nie zmieniają się. Jedyna różnica polega na braku odczytywania informacji wejściowych i uaktualniania stanów wyjść. W trybie tym możliwe jest używanie debagera ISaGRAF do modyfikacji wartości wejściowych. Atrybut **Wirtualna** odnosi się do całości karty. Ustawiany jest on podczas definicji We/Wy karty, **przed** generacją kodu aplikacji. Atrybut **wirtualna** jest cechą **statyczną** i jest zapisywany, gdy aplikacja zostaje zatrzymana i ponownie uruchomiona.

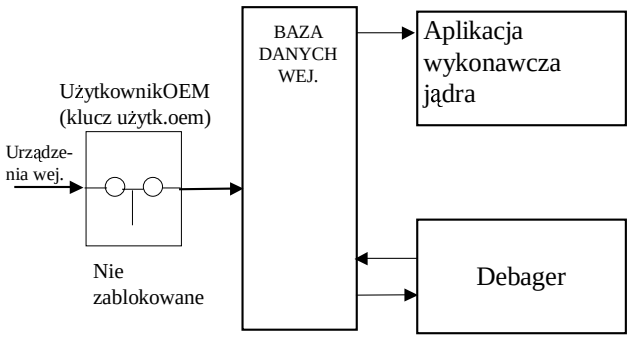
Inną możliwością jest **blokowanie** zmiennej We/Wy. Polega to na rozłączeniu jednego urządzenia fizycznego i odpowiadającej mu zmiennej We/Wy pakietu ISaGRAF. Blokowanie i odblokowywanie zmiennej odbywa się poprzez debager. Blokowanie zmiennej jest operacją **dynamiczną** i nie jest zapamiętane gdy aplikacja jest ponownie uruchamiana. Operacja **blokowania** odnosi się jedynie do **jednej** zmiennej (jeden kanał We/Wy) naraz. Oto podsumowanie głównych funkcji kontrolujących We/Wy:

	Atrybut Wirtualna	Komenda blokowania
narzędzie wyboru	połączenie karty We/Wy	debager
definicja	statyczna	dynamiczna
tryb wyboru	karta	zmienna
aplikacja	kontrola poprawności i testy	konserwacja

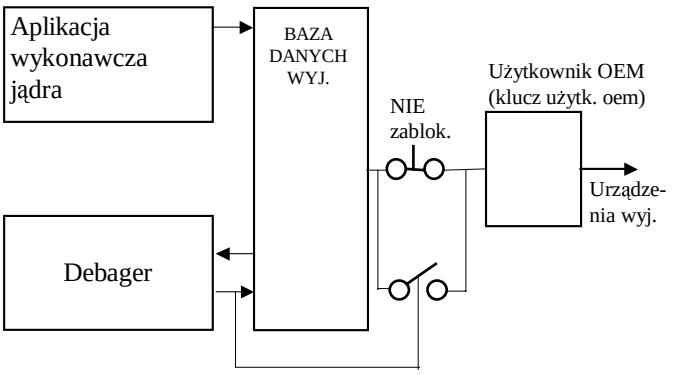
Poniższy schemat ilustruje przepływ danych We/Wy pomiędzy zadaniami ISaGRAF:



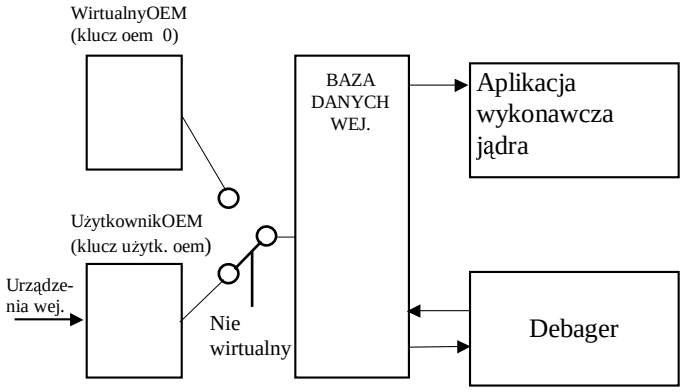
Kiedy zmienna wejściowa jest zablokowana, nie zmienia dostęp do bazy danych, lecz rozłączone zostaje urządzenie wejściowe. Wartości wejściowe mogą być ustalane za pomocą debagera i przetwarzane przez jądro ISaGRAF:



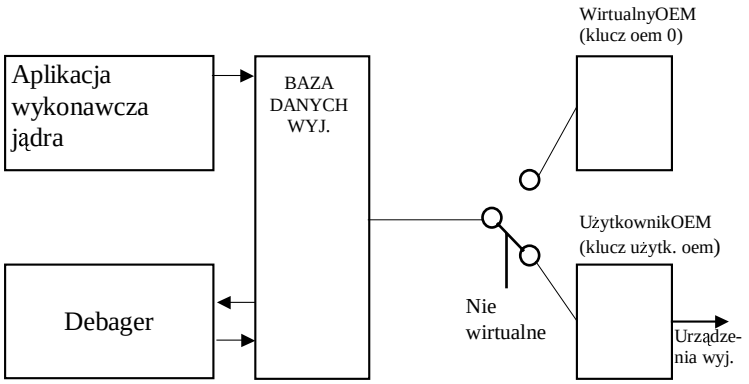
Kiedy zmienna wyjściowa jest zablokowana, jądro wykonawcze i sterownik wyjściowy są rozłączone. W takim przypadku dostęp do urządzenia wyjściowego jest nadal możliwy poprzez sterownik wyjściowy, za pomocą debagera ISaGRAF:



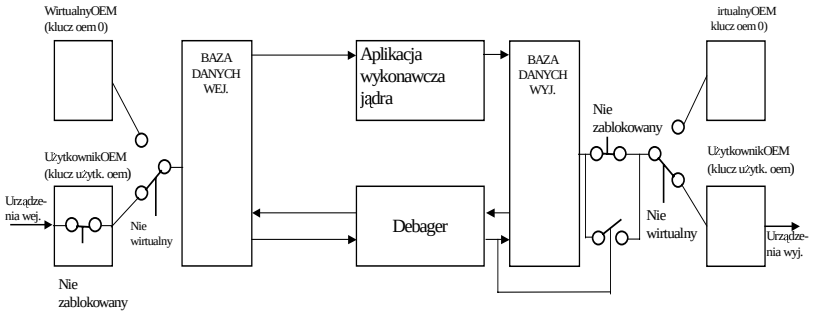
Przy ustawianiu atrybutu wirtualny dla wejścia, baza danych wejściowych i związane z nią urządzenia wejściowe zostają rozłączone. Wirtualny sterownik We/Wy zastępuje sterownik rzeczywisty.



Ustawienie atrybutu Wirtualna dla karty wejściowej i wyjściowej odbywa się według tych samych zasad. W przypadku karty wyjściowej, jądro ISaGRAF aktualizuje bazę danych wyjściowych. Jednak ta baza danych wraz ze skojarzonymi z nią urządzeniami wyjściowymi zostaje rozłączona. Wirtualny sterownik We/Wy zastępuje sterownik rzeczywisty.



Zestawienie wszystkie możliwości:



A.26.3 Kontrola poprawności połączenia PC-PLC

Większość problemów związanych ze złą komunikacją pomiędzy pakietem ISaGRAF, a oprogramowaniem sterownika PLC przedstawiana jest w oknie debagera przy pomocy komunikatu statusu "**rozłączony**". Przed wykonaniem jakichkolwiek testów diagnostycznych należy skontrolować poprawność komunikacji przy **braku aktywnych aplikacji** w oprogramowaniu wbudowanym PLC. W ten sposób można skontrolować poprawność samego szeregowego połączenia komunikacyjnego, niezależnie od skutków związanych z wykonywaniem aplikacji.

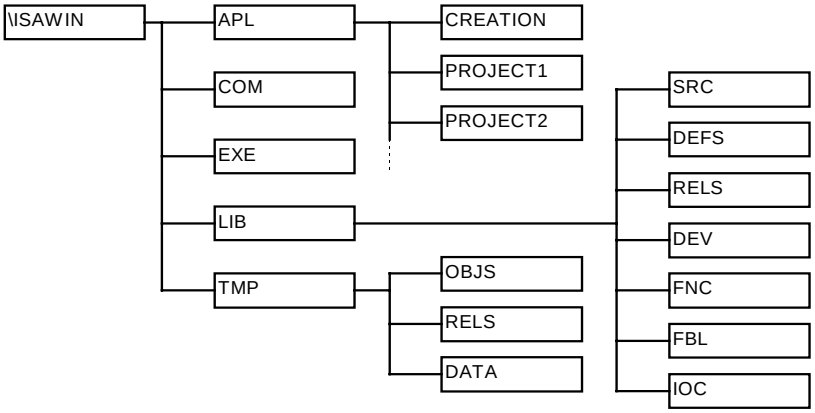
Język "C", stosowany do opisu funkcji konwersji oraz funkcji C, pozwala na bezpośredni dostęp do systemu oprogramowania sterownika. Błąd programistyczny w takim elemencie oprogramowania może generować błędy lub nieprawidłowości działania systemu ISaGRAF. Problemy takie mogą wystąpić, kiedy sterowniki We/Wy są opracowywane za pomocą pakietu narzędziowego **We/Wy** systemu ISaGRAF. Na przykład błędy systemowe mogą wystąpić, jeśli karta We/Wy jest skojarzona z

nieprawidłowym adresem magistrali. Poniższa tabela zawiera syntetyczne podsumowanie diagnostyki błędów:

status	kontekst	Diagnostyka
"rozłączony" (przed pobraniem)		- oprogramowanie wbudowane nie działa - brak przewodu / przewód nieprawidłowy - nieprawidłowe parametry łącza - źle zainstalowane oprogramowanie wbudowane ISaGRAF
"rozłączony" (po pobraniu)	cykl po cyklu tryb uruchom.	- nieprawidłowa konfiguracja We/Wy - zatrzymanie awaryjne systemu
	czas rzeczywisty tryb uruchom.	- nieprawidłowa konfiguracja We/Wy - zatrzymanie awaryjne systemu (z powodu programu "C")
"brak aplikacji"		- nie pobrano aplikacji - nie uruchomiono aplikacji (z powodu programu "C") - niezgodność Intel/Motorola - nieprawidłowa wersja oprogramowania wbudowanego

A.26.4 Katalogi ISaGRAF

Pakiet ISaGRAF pracuje w ramach dedykowanej struktury katalogów dysku. Katalog główny tej architektury wybierany jest przez użytkownika podczas instalacji ISaGRAF. Domyślną nazwą katalogu głównego ISaGRAF jest **ISAWIN**. Oto standardowa architektura dyskowa tworzona przez program instalacyjny:



Oto standardowe podkatalogi ISaGRAF

KATALOG	ZAWARTOŚĆ
APL	Katalog główny projektów ISaGRAF Każdy projekt odpowiada jednemu podkatalogowi, który zawiera wszystkie dane tego projektu Dla różnych grup projektów mogą istnieć różne katalogi. Program instalacyjny ISaGRAF tworzy katalog "**SMP**", gdzie przechowywane są przykłady aplikacji.
COM	Dane o zasięgu "ogólnym" Dane mogą być wykorzystywane przez dowolny projekt
EXE	Programy i pliki pomocy ISaGRAF
LIB	Biblioteki ISaGRAF: - listy elementów - parametry lub interfejs dla każdego z elementów - opisy
LIB\IOC	kod źródłowy do konfiguracji We/Wy
LIB\FNC	kod źródłowy funkcji napisanych w językach IEC
LIB\FBL	kod źródłowy bloków funkcyjnych napisanych w językach IEC
LIB\SRC	kod źródłowy konwersji i funkcji C
LIB\DEFS	nagłówek źródłowy konwersji i funkcji C
LIB\RELS	kod obiektowy konwersji i funkcji C
LIB\DEV	pliki komend do opracowywania bibliotek "C" plików tworzących, list połączeń, itp.
TMP	pliki tymczasowe: podkatalogi TMP są zarezerwowane dla Generatora Kodów ISaGRAF i nie mogą być usuwane.

Podkatalogi można przenosić w inne miejsca dysku. Jeśli użytkownik korzysta z architektury niestandardowej, ścieżki dostępu do podkatalogów należy zadeklarować w sekcji **WS001** pliku inicjalizacyjnego **ISA.ini** w podkatalogu **EXE** ISaGRAF. Oto zapisy w sekcji **WS001**:

Isa	katalog główny architektury ISaGRAF

IsaExe	katalog główny programów i plików pomocy ISaGRAF
IsaApl	katalog główny projektów ISaGRAF
IsaTmp	katalog plików tymczasowych
IsaSrc	katalog kodu źródłowego bibliotek
IsaDefs	katalog nagłówków źródłowych bibliotek

Uwaga: jeśli zamiast IsaTmp zostanie podany na inny katalog, w tym nowym katalogu trzeba stworzyć podkatalogi OBJS, RELS i DATA. Poniższy przykład używa zapisy z sekcji **WS001** do ponownego zdefiniowania standardowej architektury dyskowej ISaGRAF:

```
;plik c:\ISAWIN\EXE\ISA.ini

[WS001]
Isa=c:\isawin
IsaExe=c:\isawin\exe
IsaApl=c:\isawin\apl
IsaTmp=c:\isawin\tmp
IsaSrc=c:\isawin\lib\src
IsaDefs=c:\isawin\lib\defs
```

Jeśli trzeba dodać funkcje lub bloki funkcyjne "C" do oprogramowania sterownika ISaGRAF, to katalog **\ISAWIN\LIB\DEV** jest używany do przechowywania plików konstrukcyjnych: plików komend, plików tworzących, map, itp. Katalog **\ISAWIN\LIB\RELS** jest używany do przechowywania plików obiektów generowanych podczas kompilacji "C" oraz bibliotek ISaGRAF "C" wymaganych w operacjach konsolidacji.

A.26.5 Symbole aplikacji

Odwołanie do każdego obiektu aplikacji ISaGRAF odbywa się za pomocą nazwy (wprowadzanej podczas deklaracji zmiennej) oraz wewnętrznego **adresu wirtualnego** obliczanego przez generator kodu. Adres wirtualny zmiennej nie jest jej adresem **sieciowym** wprowadzanym podczas deklaracji zmiennej. Adresy wirtualne są wykorzystywane do komunikowania się oraz do specjalnych aplikacji "C" wykorzystujących opcję OEM. Po uruchomieniu generator kodu ISaGRAF tworzy plik ASCII zawierający połączenia logiczne pomiędzy nazwami i adresami wirtualnymi wszystkich obiektów (zmiennych, programów, etapów...) projektu. Do pliku tego można z każdej aplikacji użytkownika w prosty sposób wysyłać zapytania o informacje dotyczące statycznej bazy danych ISaGRAF. Plik ten nosi nazwę "**APPLI.TST**" i umieszczony jest w katalogu projektu ISaGRAF: "**\ISAWIN\APL\nazwapro**" (nazwapro oznacza nazwę projektu). Niniejsza sekcja szczegółowo opisuje format pliku "**APPLI.TST**". Stosowane są następujące oznaczenia:

VA	adres wirtualny
ATTR	atrybut zmiennej
USP	funkcja "C"

Poniżej podano możliwe wartości atrybutów zmiennej. Wartości takie występują w polach "**atrybuty**":

+X zmienna wewnętrzna
+C zmienna wewnętrzna tylko do odczytu
+I zmienna wejściowa
+O zmienna wyjściowa

Wszystkie numery poza adresami wirtualnymi wyrażone są dziesiętnymi liczbami całkowitymi. Adresy wirtualne (**VA**) wyrażone są jako 4-cyfrowe liczby szesnastkowe, poprzedzone znakiem "**!**". Na przykład:

`123` to jest liczba dziesiętna
`!A003` to jest szesnastkowy adres wirtualny

Struktura główna pliku "**APPLI.TST**" przedstawiona jest poniżej. Plik ma strukturę listy **bloków**. Blok, to lista **rekordów**. Każdy rekord jest opisany jedną linią tekstu. Każdy blok rozpoczyna się od nagłówka umieszczonego w jednej linii tekstu.

```
Blok początkowy
Bloki opisowe
Blok końca
```

Ogólna struktura jednego bloku przedstawiona jest poniżej:

```
@ <nazwa_bloku> <argumenty>
#rekord...
#rekord...
...
```

Pierwszy blok, zawierający główne informacje dotyczące aplikacji, ma następującą strukturę:

```
@ISA_SYMBOLS,<appli_crc>
#NAME,<appli_name>,<version>
#DATE,<creation_date>
#SIZE,G=<nbprg>,S=<nbstep>,T=<nbtra>,L=0,P=<nbpro>,V=<nbvar>
#COMMENT,cj international
```

appli_crc suma kontrolna symboli aplikacji
appli_name nazwa aplikacji
version numer wersji pakietu ISaGRAF
creation_date data wygenerowania aplikacji
nbprg ilość programów
nbstep ilość etapów SFC
nbtra ilość przejść SFC
nbpro ilość wykorzystanych funkcji "C"
nbvar łączna ilość zmiennych

Struktura ostatniego bloku, który sygnalizuje koniec pliku, jest przedstawiona poniżej:

```
@END_SYMBOLS
```

Blok używany do opisu programów aplikacji ma następującą strukturę:

```
@PROGRAMS,<nbprg>
#<va>,<name>
#...
```

nbprg ilość programów zdefiniowanych w danym bloku
va adres wirtualny programu
name nazwa programu

Strukturę bloku używanego do opisu etapów SFC aplikacji przedstawiono poniżej. Należy zauważyć, że dla każdego programu nie będącego SFC zdefiniowano jeden etap wirtualny:

```
@STEPS,<nbsteps>
#<va>,<name>,<father>
#...
```

nbsteps ilość etapów zdefiniowanych w danym bloku
va adres wirtualny etapu
name nazwa etapu
father adres wirtualny rodzica

Strukturę bloku wykorzystywanego do opisu przejść SFC aplikacji przedstawiono poniżej:

```
@TRANSITIONS,<nbtrans>
#<va>,<name>,<father>
#...
```

nbtrans ilość przejść zdefiniowanych w danym bloku
va adres wirtualny przejścia
name nazwa przejścia
father adres wirtualny rodzica

Strukturę bloku używanego do opisu zmiennych binarnych aplikacji przedstawiono poniżej:

```
@BOOLEANS,<nb_boo>
#<va>,<name>,<attr>,<program>,<eq_false>,<eq_true>
#...
```

a jeśli ilość zmiennych przekracza 4095:

```
X#(1.<varno>),<name>,<attr>,<program>,<eq_false>,<eq_true>
```

nb_boo ilość zmiennych w danym bloku
va adres wirtualny zmiennej
varno zakres adresów (w ramach danych typu binarnego)
name nazwa zmiennej
attr atrybut zmiennej
program adres wirtualny programu-rodzica
lub "**!0000**" dla zmiennej globalnej

eq_false ciąg stosowany dla wartości „fałsz”
eq_true ciąg stosowany dla wartości „prawda”

Strukturę bloku używanego do opisu zmiennych analogowych aplikacji przedstawiono poniżej:

```
@ANALOGS,<nb_ana>
#<va>,<name>,<attr>,<program>,<format>,<unit>
#...
```

a jeśli ilość zmiennych przekracza 4095:
`X#(2.<varno>),<name>,<attr>,<program>,<eq_false>,<eq_true>`

nb_ana ilość zmiennych w danym bloku
va adres wirtualny zmiennej
varno zakres adresów (w ramach danych typu analogowego)
name nazwa zmiennej
attr atrybut zmiennej
program adres wirtualny programu-rodzica
lub "**!0000**" dla zmiennej globalnej
format = "**I**" dla zmiennej będącej liczbą całkowitą
= **"F**" dla zmiennej rzeczywistej
unit ciąg jednostkowy

Strukturę bloku używanego do opisu zmiennych typu Timer aplikacji przedstawiono poniżej:

```
@TIMERS,<nb_tmr>
#<va>,<name>,<attr>,<program>
#...
```

a jeśli ilość zmiennych przekracza 4095:
`X#(3.<varno>),<name>,<attr>,<program>,<eq_false>,<eq_true>`

nb_tmr ilość zmiennych w danym bloku
va adres wirtualny zmiennej
varno zakres adresów (w ramach danych typu Timer)
name nazwa zmiennej
attr atrybut zmiennej (zawsze +X: wewnętrzna)
program adres wirtualny programu-rodzica
lub "**!0000**" dla zmiennej globalnej

Strukturę bloku używanego do opisu aplikacyjnych zmiennych typu komunikat przedstawiono poniżej:

```
@MESSAGES,<nb_msg>
#<va>,<name>,<attr>,<program>,< max_len>
#...
```

a jeśli ilość zmiennych przekracza 4095:
`X#(4.<varno>),<name>,<attr>,<program>,<eq_false>,<eq_true>`

nb_msg	ilość zmiennych w danym bloku
va	adres wirtualny zmiennej
varno	zakres adresów (w ramach danych typu komunikat)
name	nazwa zmiennej
attr	atrybut zmiennej
program	adres wirtualny programu-rodzica lub "**!0000**" dla zmiennej globalnej
max_len	maksymalna długość (zadeklarowana pojemność)

Strukturę bloku używanego do opisu funkcji „C" używanych w aplikacji przedstawiono poniżej:

```
@USP,<nb_usp>
#<va>,<name>
#...
```

nb_usp	ilość funkcji C w danym bloku
va	adres wirtualny funkcji C
name	nazwa funkcji C

Strukturę bloku używanego do opisu wystąpień *bloku funkcyjnego* „C" używanych w aplikacji przedstawiono poniżej:

```
@FBINSTANCES,<nb_fb>
#<va>,<inst_name>,<fb_name>
#...
```

nb_fb	ilość wystąpień bloku funkcyjnego C w danym bloku
va	adres wirtualny instancji bloku funkcyjnego C
inst_name	nazwa instancji bloku funkcyjnego C
fb_name	nazwa odsyłacza do instancji bloku funkcyjnego C

A.26.6 Ograniczenia pakietu ISaGRAF, wersja "LARGE" (WDL)

Obiekty wykorzystywane w pakiecie ISaGRAF posiadają pewne ograniczenia. Rzecz jasna istnieje wiele innych ograniczeń praktycznych, związanych z konfiguracją wykorzystywanego komputera (dostępna pamięć i przestrzeń dyskowa) oraz z możliwościami systemu oprogramowania wbudowanego sterownika ISaGRAF (dostępna pamięć, dostępne zasoby sprzętowe i programowe, itp.). Poniżej wyszczególniono limity absolutne, których nie można przekraczać.

Dla projektu:

Obiekty	*Maksymalnie*	*Uwagi*
Programy	255	grupowanie programów głównych, potomnych i podprogramów
Poziomy hierarchii	20	

Ilość projektów zainstalowanych w pakiecie ISaGRAF ograniczona jest jedynie dostępną przestrzenią dysku twardego.

Dla nazw:

Nazwa:	*Maksymalnie*	*Uwagi*
Projektu	8 znaków	
Programu	8 znaków	
Zmiennej	16 znaków	+ 60 znaków na komentarz
Zdef. etykiety słownej	16 znaków	
Zdef. równoważności	255 znaków	+ 60 znaków na komentarz
Tablica konwersji	16 znaków	
Listy zmiennych	16 znaków	
funkcji / bloku f. (bibl) funkcyjnych C	8 znaków	odnosi się to do funkcji C, bloków lub funkcji napisanych w językach IEC
parametru funkcji (bibl) funkcyjnych C	16 znaków	odnosi się to do funkcji C, bloków lub funkcji napisanych w językach IEC
karty We/Wy	8 znaków	
konfiguracji We/Wy	8 znaków	
Parametru oem karty	16 znaków	
Funkcji konwersji	8 znaków	

Edycja (dla jednego programu):

Obiekty	*Maksymalnie*	*Uwagi*
Wiersze SFC	600	
Kolumny SFC	20	
Etapy SFC grupujących, etapów wstępnych, kończących	4095	dla całego projektu, etapów etapów rozpoczynających i
Przejścia SFC	4095	dla całej aplikacji
Edycja LD/FBD	200 kolumn 2000 wierszy	jest to wielkość obszaru edycyjnego liczona w komórkach.
Edycja LD PC	brak limitu	limity narzuca wydajność komputera
Etykiety IL	251	w tym samym programie IL
Edycja tekstu	40 kilobajtów	

Dla słownika (dla jednego projektu):

Obiekty	*Maksymalnie*	*Uwagi*
Zmienne binarne	65535	
Zmienne analogowe	65535	grupujące zmienne typu integer i real
Liczniki czasu	65535	
Zmienne komunikatów	65535	

Podane ograniczenie maksymalnej liczby zmiennych binarnych, analogowych czy komunikatów łączy zmienne wewnętrzne, wejściowe i wyjściowe. Obejmuje ono również wszelkie zmienne ukryte, tymczasowe lub utworzone przez kompilatory. Ilość zmiennych edytowanych łącznie (ten sam typ, taki sam zakres) w edytorze słownika nie może przekroczyć 16000. W zależności od konfiguracji komputera PC, limit ten może być mniejszy niż 16000. Aplikacja nie może działać na sterowniku ISaGRAF w wersji V1.21 lub wcześniejszej, jeśli łączna ilość zmiennych jednego typu przekracza

4095. Jeśli łączna ilość zmiennych jednego typu przekracza 4095, nie może zostać wykorzystane standardowe łącze "Modbus", korzystające z adresów sieciowych.

Zdefiniowane słowa	4095	na tej samej liście (taki sam zakres)
Zdefiniowane słowa programie	255	wykorzystanych w tym samym
Tabele konwersji	127	wykorzystanych w aplikacji
Punkty w jednej tabeli konwersji	32	zdefiniowanych w tej samej tabeli

Połączenia We/Wy:

Obiekty	*Maksymalnie*	*Uwagi*
Karty We/Wy	256	zdefiniowanych dla tej samej aplikacji (karty lub zespołu)

Ilość płytek We/Wy, łącznie z pojedynczymi kartami i zespołami nie może przekraczać 256.

Kanały We/Wy	128	na jednej płytce

Dla bibliotek:

Obiekty	**Maksymalnie**	**Uwagi**
Funkcje (język IEC)	255	
Bloki funkcyjne (język IEC)	255	
Funkcje C	255	zainstalowane wspólnie w bibliotece
Bloki funkcyjne C	255	zainstalowane wspólnie w bibliotece
Instancje bloków funkcyjnych	4095	zainstalowane wspólnie w bibliotece dla tych samych typów bloków funkcyjnych w tej samej aplikacji
Parametry wej. Funkcji	31	odnosi się to do funkcji C i funkcji napisanych w jęz. IEC
Parametry bloków funkcyjnych	32	dowolnie rozmieszczonych między parametrami wej. i wyj. Wymagany jest przynajmniej 1 parametr wyjściowy.
Funkcje konwersji	128	zainstalowane wspólnie w bibliotece
Konfiguracje We/Wy	255	zainstalowane wspólnie w bibliotece
Karty We/Wy	255	zainstalowane wspólnie w bibliotece
Zespoły We/Wy	255	zainstalowane wspólnie w bibliotece
Parametry oem karty	16	

B. Języki programowania

B.1 Architektura projektu

Projekt ISaGRAF podzielony jest na kilka elementów programistycznych, zwanych **programami**. Programy projektu połączone są ze sobą w architekturze drzewiastej. Programy mogą być opisywane przy użyciu któregokolwiek z następujących języków graficznych lub tekstowych: **SFC, FC (Flow Chart – Graf przepływowy)**, **FBD**, **LD, ST** lub **IL**.

B.1.1 Programy

Program to logiczny element programistyczny, który opisuje operacje na **zmiennych** procesu. Programy opisują operacje **sekwencyjne** lub **cykliczne**. Programy cykliczne są wykonywane w każdym cyklu przetwarzania oprogramowania sterownika. Wykonywanie programów sekwencyjnych jest zgodne z dynamicznymi zasadami języka **SFC** lub języka **FC**.

Programy są połączone ze sobą w formie drzewa hierarchicznego. Programy umieszczone na szczycie hierarchii są uaktywniane przez system. Podprogramy (niższy poziom hierarchii) są uaktywniane przez swoje programy-rodziców. Program może być opisany przy pomocy dowolnego z poniższych języków graficznych lub tekstowych:

Sequential Function Chart (SFC) Graf Funkcji Sekwencyjnych – programowanie wysokiego poziomu
Flow Chart (FC) Graf Przepływowy – programowanie wysokiego poziomu
Function Block Diagram (FBD) Diagram Bloków Funkcyjnych – skomplikowane operacje cykliczne
Ladder Diagram (LD) Diagram Drabinowy (Język stykowy)– tylko operacje logiczne
Structured Text (ST) Tekst Strukturalny – wszelkie operacje cykliczne
Instruction List (IL) Lista Instrukcji - – operacje niskiego poziomu

W jednym programie nie można używać kilku języków, z wyjątkiem programowania w językach LD i FBD, które mogą znajdować się w jednym diagramie.

B.1.2 Operacje cykliczne i sekwencyjne

Hierarchia programów podzielona jest na cztery główne **sekcje** lub grupy:

Początek	programy wykonywane na początku każdego cyklu oprogramowania wbudowanego
Sekwencyjnie	programy działające zgodnie z zasadami dynamicznymi SFC lub FC
Koniec	programy wykonywane na końcu każdego cyklu oprogramowania wbudowanego
Funkcje	zbiór podprogramów niededykowanych

Programy sekcji **'Początek'** lub **'Koniec'** opisują operacje cykliczne i są niezależne od czasu. Programy sekcji '**Sekwencyjnie**' opisują operacje sekwencyjne, w których zmienna czasowa bezpośrednio synchronizuje podstawowe operacje. Główne programy sekcji **'Początek'** są wykonywane systematycznie na początku każdego cyklu wykonawczego. Główne programy sekcji **'Koniec'** są wykonywane systematycznie na końcu każdego cyklu wykonawczego. Główne programy sekcji **'Sekwencyjnie'** są wykonywane zgodnie z zasadami dynamicznymi **SFC** lub **FC**.

Programy sekcji "**Funkcje**" są podprogramami, które mogą zostać wywołane przez dowolny inny program projektu. Programy sekcji "**Funkcje**" mogą wywoływać inne programy tej sekcji. Programy główne i potomne sekcji sekwencyjnej muszą być opisane językiem **SFC** lub **FC**. Programy sekcji cyklicznych (**Początek** i **Koniec**) nie mogą być opisywane w języku **SFC** ani **FC**. Dowolny program dowolnej sekcji może posiadać jeden lub więcej podprogramów. Dowolny program sekcji sekwencyjnej może posiadać jeden lub więcej programów potomnych **SFC** lub **FC** (zgodnych z własnym językiem). Podprogramy nie mogą być opisywane za pomocą języka **SFC** lub **FC**.

Programy sekcji **Początek** są zazwyczaj stosowane do opisu operacji wstępnych na urządzeniach wejściowych do stworzenia zmiennych filtrowanych wysokiego poziomu. Zmienne takie są często wykorzystywane w programach sekcji **Sekwencyjnie**. Programy sekcji **Koniec** są zazwyczaj wykorzystywane do opisu operacji bezpieczeństwa dotyczących zmiennych, który stany wypracowuje sekcja **Sekwencyjnie**, przed wysłaniem wartości do urządzeń wyjściowych.

B.1.3 Programy potomne SFC i FC

Każdy program **SFC** sekcji sekwencyjnej może sterować innymi programami **SFC**. Takie programy niskiego poziomu zwane są **programami potomnymi SFC**. **Program potomny SFC** to program równoległy, który może zostać uruchomiony, zamknięty, wstrzymany lub ponownie uruchomiony przez program-rodzica (program macierzysty). Zarówno program-rodzic jak i program potomny muszą być opisane w języku **SFC**. Program potomny SFC może posiadać zdefiniowane słowa i zmienne lokalne.

Kiedy program-rodzic uruchamia program potomny **SFC**, wstawia **znacznik** SFC (aktywuje go) w każdym etapie początkowym programu potomnego. Komenda ta opisywana jest instrukcją **GSTART**. Kiedy program-rodzic zamyka program potomny **SFC**, usuwa wszystkie znaczniki znajdujące się w **etapach** programu potomnego. Komenda taka opisywana jest instrukcją **GKILL**.

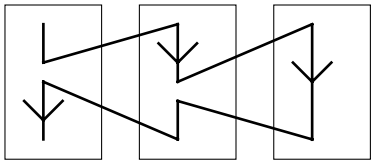
Kiedy program-rodzic wstrzymuje program potomny **SFC**, usuwa wszystkie znaczniki znajdujące się w programie potomnym i zachowuje ich pozycje w pamięci. Komenda taka opisywana jest instrukcją **GFREEZE**. Kiedy program-rodzic ponownie uruchamia wstrzymany program potomny **SFC**, odzyskuje wszystkie znaczniki usunięte podczas wstrzymania programu potomnego. Komenda taka opisywana jest instrukcją **GRST**.

Dowolny program **FC** sekcji sekwencyjnej może sterować innymi podprogramami **FC**. Program-rodzic **FC** jest blokowany (czeka) podczas wykonywania podprogramu FC. Nie jest

niemożliwe równoczesne wykonywanie operacji przez program-rodzica FC i jeden z jego podprogramów FC.

B.1.4 Funkcje i podprogramy

Wykonanie podprogramu lub funkcji jest sterowane przez ich program macierzysty (rodzica). Wykonywanie programu macierzystego jest wstrzymywane do momentu zakończenia podprogramu lub funkcji :



główny **podprogramy**

Każdy program dowolnej sekcji może zawierać jeden lub więcej podprogramów. Podprogram jest kontrolowany tylko przez jeden program macierzysty. W podprogramie mogą występować definicje i zmienne lokalne. Do opisu podprogramu może być wykorzystany dowolny język prócz **SFC** i **FC**. Programy sekcji "**Funkcje**" to podprogramy, które są wywoływane przez inne programy projektu. W odróżnieniu od innych podprogramów, nie są one dedykowane jednemu programowi macierzystemu. Program sekcji "**Funkcje**" może wywołać inny program tej sekcji. Funkcja może zostać umieszczona w Bibliotece.

Uwaga: System ISaGRAF nie obsługuje **rekursywnych** wywołań funkcji. Jeśli program sekcji "**Funkcje**" zostanie wywołany przez siebie lub przez jeden z wywołanych przez niego podprogramów, to wystąpi błąd wykonania.
Uwaga: Funkcje i podprogramy nie „zachowują" lokalnej wartości zmiennych lokalnych. Funkcja lub podprogram nie jest instancyjny, a więc nie może odwołać się do bloków funkcyjnych.

Interfejs podprogramu musi być wyraźnie zdefiniowany za pomocą **typu** i **unikalnej nazwy** dla każdego parametru wywołania lub parametru zwracanego. Aby umożliwić obsługę konwencji języka **ST**, parametr zwracany musi mieć taką samą nazwę jak podprogram.

Poniższa tabela pokazuje, jak należy ustawiać wartości parametru zwracanego w treści podprogramu w różnych językach:

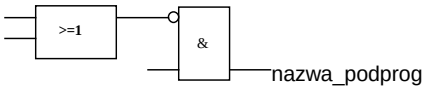
ST: przyporządkuj parametr zwracany wykorzystując jego nazwę
(taką samą jak nazwa podprogramu):

```
nazwa_podprog := <wyrażenie>;
```

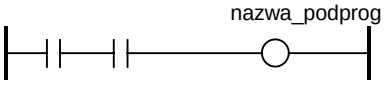
IL: wartość aktualnego wyniku (rejestr IL)
na końcu sekwencji umieszczona jest w parametrze zwracanym:

```
LD    10
ADD   20   (* wartość parametru zwracanego = 30 *)
```

FBD: ustaw parametr zwracany wykorzystując jego nazwę:

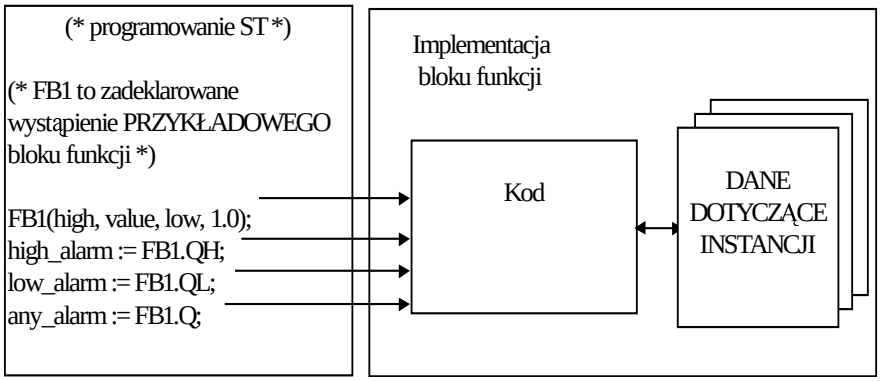


LD: zastosuj symbol cewki z nazwą parametru zwracanego:



B.1.5 Bloki funkcyjne

Bloki funkcyjne mogą wykorzystywać języki: LD, FBD, ST lub IL. Bloki funkcyjne są instancyjne. Oznacza to, że zmienne lokalne bloku funkcyjnego są kopiowane dla każdej instancji. Gdy jest wywołany blok w programie, wywoływana jest właściwie instancja tego bloku: wywoływany jest ten sam kod, lecz używane dane są danymi przyporządkowanymi do danego instancji. Wartości zmiennych dla instancji są przechowywane z cyklu na cykl.



Ostrzeżenia:

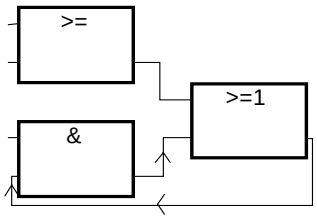
- Blok funkcyjny napisany w jednym z języków IEC nie może wywoływać innych bloków funkcyjnych: mechanizm instancyjności radzi sobie jedynie ze zmiennymi lokalnymi samego bloku. Oto lista standardowych bloków funkcyjnych, których nie można stosować wewnątrz bloku funkcyjnego IEC:
SR, RS, R_Trig, F_Trig, SEMA, CTU, CTD, CTUD, TON, TOF, TP, CMP, StackInt, AVERAGE, HYSTER, LIM_ALRM, INTEGRAL, DERIVATE, BLINK, SIG_GEN

- Z tego samego powodu nie można stosować Dodatnich czy Ujemnych styków ani cewek, jak również cewek S-R.

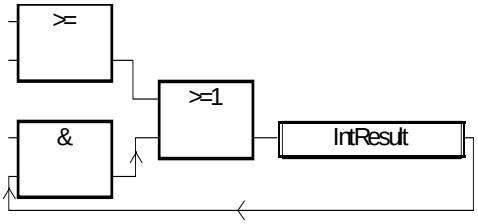
- funkcje TSTART i TSTOP uruchamiające i zatrzymujące liczniki, nie mogą być używane w blokach funkcyjnych oprogramowania wbudowanego w wersji 3.0x. Działają one od wersji 3.20.

- Jeśli w blokach funkcyjnych wymagana jest pętla, należy przed wykonaniem pętli zastosować zmienną lokalną. Patrz poniższy przykład:

Tak nie będzie działać:



Tak jest dobrze:



B.1.6 Języki

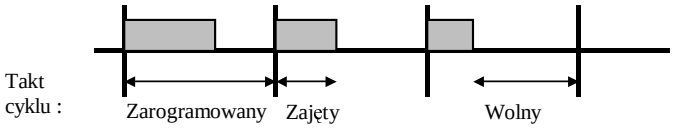
Program może być opisywany przy pomocy dowolnego z poniższych języków graficznych lub tekstowych:

Sequential Function Chart (SFC) Graf Funkcji Sekwencyjnych – programowanie wysokiego poziomu
Flow Chart (FC) Graf Przepływowy – programowanie wysokiego poziomu
Function Block Diagram (FBD) Diagram Bloków Funkcyjnych – skomplikowane operacje cykliczne
Ladder Diagram (LD) Diagram Drabinowy (Język stykowy) – tylko operacje logiczne
Structured Text (ST) Tekst Strukturalny – wszelkie operacje cykliczne
Instruction List (IL) Lista Instrukcji - – operacje niskiego poziomu

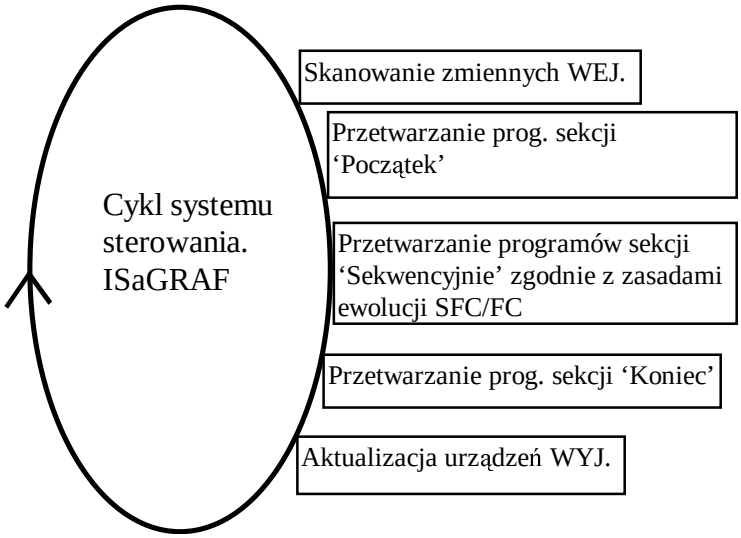
W jednym programie nie można stosować kilku języków. Język opisu programu wybierany jest przy tworzeniu i nie może być później zmieniony. Wyjątkiem są języki LD i FBD, które można łączyć w jednym programie.

B.1.7 Składowe cyklu sterowania

ISaGRAF jest systemem **synchronicznym**. Wszystkie operacje są inicjowane (wyzwalane) przez zegar. Podstawowa jednostka zegara nazywana jest taktem cyklu:



Podstawowe operacje przetwarzane podczas cyklu sterowania, to:



System ten:

- gwarantuje zachowanie przez zmienną wejściową stałej wartości w czasie trwania cyklu,
- gwarantuje aktualizację stanu urządzenia wyjściowego nie więcej niż jeden raz w ciągu cyklu,
- umożliwia bezpieczną pracę na jednej zmiennej globalnej w różnych programach,
- pozwala na oszacowanie i kontrolowanie czasu odpowiedzi zakończonej aplikacji.

B.2 Obiekty wspólne

Oto główne cechy i **obiekty** wspólne bazy danych programistycznych ISaGRAF. Obiekty takie mogą być wykorzystywane w dowolnym programie napisanym w dowolnym z następujących języków **SFC**, **FC**, **FBD**, **LD**, **ST** lub **IL**.

B.2.1 Typy podstawowe

Każda stała, wyrażenie czy zmienna użyta w programie (napisanym w dowolnym języku) musi być opisana przez podanie typu. W operacjach graficznych i wyrażeniach znakowych należy przestrzegać spójności typu. Oto podstawowe typy dostępne do opisu obiektów programistycznych:

BINARNY:	wartość binarna (prawda lub fałsz)
ANALOGOWY:	ciągła wartość typu integer lub real (zmiennoprzecinkowa)
TIMER:	wartość czasowa
KOMUNIKAT:	ciąg znaków

Uwaga: Wartości typu Timer zawierają informacje dotyczące okresów krótszych niż jeden dzień i nie mogą być używane do przechowywania dat.

B.2.2 Wyrażenia stałe

Wyrażenia stałe odnoszą się do jednego typu. Nie można używać tej samej notacji do przedstawienia wyrażeń stałych różnych typów.

B.2.3 Wyrażenia stałe binarne

Istnieją tylko dwa binarne wyrażenia stałe:

PRAWDA	odpowiada wartości całkowitej 1
FAŁSZ	Odpowiada wartości całkowitej 0

Słowa kluczowe „True” i „False” nie uwzględniają wielkości liter.

B.2.4 Wyrażenia stałe analogowe typu integer

Wyrażenia stałe typu integer są długimi 32-bitowymi liczbami całkowitymi ze znakiem przedziału **od -2147483647 do +2147483647,**Analogowe stałe typu integer mogą być wyrażane przy pomocy jednej z poniższych **podstaw**. Zastosowana podstawa jest określona **prefiksem** poprzedzającym wartość:

Podstawa	*Prefiks*	*Przykład*
DZIESIĘTNA	**(brak)**	**-908**
SZESNASTKOWA	**„16#”**	**16#1A2B3C4D**
ÓSEMKOWA	**„8#”**	**8#1756402**
BINARNA	**„2#”**	**2#1101_0001_0101_1101**

Znak podkreślenia (‘_’) może by ć używany do oddzielenia grup cyfr. Nie ma on szczególnego znaczenia, a jest stosowany do zwiększenia czytelności wyrażenia.

B.2.5 Wyrażenia stałe analogowe typu real

Wyrażenia stałe analogowe typu real mogą być zapisywane w formie dziesiętnej lub w formie wykładniczej. Kropka dziesiętna (‘.’) oddziela części całkowite od dziesiętnych. Musi ona być używana aby odróżnić wyrażenie stałe real od wyrażenia integer. Forma wykładnicza wykorzystuje literę ‘E’ lub ‘F’ do oddzielenia mantysy od wykładnika. Wykładnik rzeczywistego wyrażenia musi być wartością całkowitą ze znakiem z przedziału od -37 do +37. Poniżej znajdują się przykłady analogowych wartości stałych typu real:

3.14159	**-1.0E+12**
+1.0	**1.0F-15**
-789.56	**+1.0E-37**

Wyrażenie „**123**” nie stanowi rzeczywistego wyrażenia stałego. Jego prawidłową formą jest „**123.0**”.

B.2.6 Wyrażenia stałe typu Timer

Wyrażenia stałe typu Timer przedstawiają wartości czasowe do **0 sekund** do **23h59m59s999ms**. Najmniejszą dopuszczalną jednostką jest milisekunda. Standardowymi jednostkami czasu używanymi w wyrażeniach stałych są:

Godzina	Litera „**h**” musi znajdować się za liczbą godzin
Minuta	Litera „**m**” musi znajdować się za liczbą minut
Sekunda	Litera „**s**” musi znajdować się za liczbą sekund
Milisekunda	Litery „**ms**” muszą znajdować się za liczbą milisekund

Stałe wyrażenie czasowe musi rozpoczynać się prefiksem „**T#**” lub „**TIME#**”. Prefiksy i litery jednostek nie uwzględniają wielkości liter. Niektóre jednostki mogą nie pojawić się. Oto przykłady wyrażeń stałych typu Timer:

T#1H450MS	1 godzina, 450 milisekund
time#1H3M	1 godzina, 3 minuty

Wyrażenie „0”nie reprezentuje wartości czasowej, lecz stałą analogową.

B.2.7 Wyrażenia stałe typu komunikat

Wyrażenia stałe w postaci łańcucha lub komunikatu reprezentują ciągi znaków. Znaki muszą być poprzedzone cudzysłowem i zakończone apostrofem. Na przykład:

‘TO JEST KOMUNIKAT’

Uwaga: Znak apostrofu ‘’’ nie może być używany wewnątrz takiego wyrażenia stałego. Wyrażenie stałe w postaci ciągu znakowego musi być zapisane w jednej linii kodu źródłowego programu. Jego długość nie może przekroczyć 255 znaków, łącznie ze spacjami.
Puste wyrażenie stałe w formie łańcucha reprezentowane jest przez dwa apostrofy bez oddzielającej spacji czy znaku tabulacji:

‘’ (* to jest pusty łańcuch *)

Znak specjalny dolara (‘**$**’), po którym występują inne znaki specjalne, może być używany w stałym wyrażeniu łańcuchowym do oznaczenia znaku nie drukowalnego:

Sekwencja	*Znaczenie*	*ASCII (szenas.)*	*Przykład*
$$	znak ‘$’	16#24	‘Zapłaciłem za to $$5’
$’	apostrof	16#27	‘Wprowadź $’Y$’ jeśli TAK’
$L	wysuw wiersza	16#0a	‘następna $L linia’
$R	powrót karetki	16#0d	‘ llo $R He’
$N	nowa linia	16#0d0a	‘to jest linia$N’
$P	nowa strona	16#0c	‘ostatnia linia $P pierwsza linia
$T	tabulacja	16#09	‘nazwa$Twielkość$Tdata’
$hh (*)	dowolny znak	16#hh	‘ABCD = $41$42$43$44’

(*) „**hh**” to wartość szesnastkowa w kodzie ASCII wyświetlonego znaku.

B.2.8 Zmienne

Zmienne mogą być **LOKALNE** dla jednego programu lub **GLOBALNE**. Zmienne lokalne mogą być używane tylko przez jeden program. Zmienne globalne mogą być używane przez dowolny program projektu.
Nazwy zmiennych muszą odpowiadać następującym regułom:
- nazwa nie może być dłuższa niż **16** znaków
- pierwszy znak musi być **literą**
- kolejne znaki mogą być **literami**, **cyframi** lub znakiem podkreślenia

B.2.8.1 Zarezerwowane słowa kluczowe

Poniżej zaprezentowano listę zarezerwowanych słów kluczowych. Identyfikatory takie nie mogą być użyte jako nazwa programu, zmiennej czy funkcji lub bloku funkcyjnego „C”:

A ANA, ABS, ACOS, ADD, ANA, AND, AND_MASK, ANDN, ARRAY, ASIN, AT, ATAN,
B BCD_TO_BOOL, BCD_TO_INT, BCD_TO_REAL, BCD_TO_STRING, BCD_TO_TIME, BOO, BOOL, BOOL_TO_BCD, BOOL_TO_INT, BOOL_TO_REAL, BOOL_TO_STRING, BOOL_TO_TIME, BY, BYTE,
C CAL, CALC, CALCN, CALN, CALNC, CASE, CONCAT, CONSTANT, COS,
D DATE, DATE_AND_TIME, DELETE, DINT, DIV, DO, DT, DWORD,
E ELSE, ELSIF, EN, END_CASE, END_FOR, END_FUNCTION, END_IF, END_PROGRAM, END_REPEAT, END_RESSOURCE, END_STRUCT, END_TYPE, END_VAR, END_WHILE, ENO, EQ, EXIT, EXP, EXPT,
F FALSE, FEDGE, FIND, FOR, FUNCTION,
G GE, GFREEZE, GKILL, GRST, GSTART, GSTATUS, GT,
I IF, INSERT, INT, INT_TO_BCD, INT_TO_BOOL, INT_TO_REAL, INT_TO_STRING, INT_TO_TIME,
J JMP, JMPC, JMPCN, JMPN, JMPNC,

L LD, LDN, LE, LEFT, LEN, LIMIT, LINT, LN, LOG, LREAL, LT, LWORD,
M MAX, MID, MIN, MOD, MOVE, MSG, MUL, MUX,
N NE, NOT,
O OF, ON, OPERATE, OR, OR_MASK, ORN,
P PROGRAM
R R, REDGE, READ_ONLY, READ_WRITE, REAL, REAL_TO_BCD, REAL_TO_BOOL, REAL_TO_INT, REAL_TO_STRING, REAL_TO_TIME, REDGE, REPEAT, REPLACE, RESSOURCE, RET, RETAIN, RETC, RETCN, RETN, RETNC, RETURN, RIGHT, ROL, ROR,
S S, SEL, SHL, SHR, SIN, SINT, SQRT, ST, STN, STRING, STRING_TO_BCD, STRING_TO_BOOL, STRING_TO_INT, STRING_TO_REAL, STRING_TO_TIME, STRUCT, SUB, SYS_ERR_READ, SYS_ERR_TEST, SYS_INITALL, SYS_INITANA, SYS_INITBOO, SYS_INITTMR, SYS_RESTALL, SYS_RESTANA, SYS_RESTBOO, SYS_RESTTMR, SYS_SAVALL, SYS_SAVANA, SYS_SAVBOO, SYS_SAVTMR, SYS_TALLOWED, SYS_TCURRENT, SYS_TMAXIMUM, SYS_TOVERFLOW, SYS_TRESET, SYS_TWRITE, SYSTEM,
T TAN, TASK, THEN, TIME, TIME_OF_DAY, TIME_TO_BCD, TIME_TO_BOOL, TIME_TO_INT, TIME_TO_REAL, TIME_TO_STRING, TMR, TO, TOD, TRUE, TSTART, TSTOP, TYPE,
U UDINT, UINT, ULINT, UNTIL, USINT,
V VAR, VAR_ACCESS, VAR_EXTERNAL, VAR_GLOBAL, VAR_IN_OUT, VAR_INPUT, ,VAR_OUTPUT,
W WHILE, WITH, WORD,
X XOR, XOR_MASK, XORN

Wszystkie słowa kluczowe rozpoczynające się od znaku podkreślenia ('_') są wewnętrznymi słowami kluczowymi i nie wolno używać ich w komendach tekstowych.

B.2.8.2 Zmienne o reprezentacji bezpośredniej

ISaGRAF umożliwia stosowanie w programach źródłowych **zmiennych o reprezentacji bezpośredniej** do reprezentowania wolnego kanału. Wolne kanały to kanały nie skojarzone ze zmiennymi We/Wy. Identyfikator zmiennej o reprezentacji bezpośredniej rozpoczyna się zawsze znakiem „**%**".
Poniżej przedstawiono konwencje nazywania zmiennych o reprezentacji bezpośredniej dla kanału pojedynczej karty. „***s***" to numer gniazda karty. „***c***" to numer kanału.

%IX*s.c* wolny kanał karty wejść binarnych
%ID*s.c* wolny kanał karty wejść typu integer
%IS*s.c* wolny kanał karty wejść typu komunikat
%QX*s.c* wolny kanał karty wyjść binarnych
%QD*s.c* wolny kanał karty wyjść typu integer
%QS*s.c* wolny kanał karty wyjść typu komunikat

Poniżej przedstawiono konwencje nazywania zmiennych o reprezentacji bezpośredniej dla kanału zespołu. „***s***" to numer gniazda zespołu. „***b***" to indeks pojedynczej karty w zespole. „***c***" to numer kanału.

%IX*s.b.c* wolny kanał karty wejść binarnych
%ID*s.b.c* wolny kanał karty wejść typu integer
%IS*s.b.c* wolny kanał karty wejść typu komunikat
%QX*s.b.c* wolny kanał karty wyjść binarnych

%QD*s.b.c* wolny kanał karty wyjść typu integer
%QS*s.b.c* wolny kanał karty wyjść typu komunikat

Oto przykłady:
%QX1.6 6-ty kanał karty nr 1 (wyjście binarne)
%ID2.1.7 7-my kanał karty nr 1 w zespole nr 2 (wejście typu integer)

Zmienna o reprezentacji bezpośredniej nie może być typu „**real**„.

B.2.8.3 Zmienne binarne

Binarne inaczej - **dwustanowe**. Zmienne takie mogą przyjmować jedną z wartości binarnych: **PRAWDA** lub **FAŁSZ (True / False)**. Zmienne binarne są zazwyczaj stosowane w wyrażeniach logicznych. Mogą posiadać jeden z następujących **atrybutów**:

Wewnętrzna: zmienna w pamięci aktualizowana przez program
Wejściowa: zmienna skojarzona z urządzeniem wejściowym (odświeżana przez system)
Wyjściowa: zmienna skojarzona z urządzeniem wyjściowym

Uwaga: Przy deklarowaniu zmiennych binarnych można podczas debagowania definiować ciągi znaków zastępujące wartości ‘prawda’ i ‘fałsz’. Ciągi te nie mogą być wykorzystywane w programach, chyba że zostaną wprowadzone w języku jako **‘zdefiniowane słowa’**.

B.2.8.4 Zmienne analogowe

Analogowe inaczej **ciągłe**. Zmienne takie mają wartości typu integer lub real (zmiennoprzecinkowe) ze znakiem. Mogą być stosowane następujące formaty zmiennych analogowych:

Integer 32-bitowa liczba całkowita ze znakiem z przedziału od **-2147483647** do **+2147483647**
Real standardowa 32-bitowa liczba zmiennoprzecinkowa IEEE (pojedynczej precyzji)
1 bit znaku + 23 bity mantysy + 8 bitów wykładnika

Analogowa wartość wykładnika typu real nie może być mniejsza od **-37** ani większa od **+37**. Zmienne analogowe mogą posiadać jeden z następujących **atrybutów**:

Wewnętrzna: zmienna w pamięci aktualizowana przez program
Wejściowa: zmienna skojarzona z urządzeniem wejściowym (odświeżana przez system)
Wyjściowa: zmienna skojarzona z urządzeniem wyjściowym

Uwaga: kiedy zmienna typu real jest skojarzona z urządzeniem We/Wy, to odpowiadający mu sterownik We/Wy operuje na równoważnej wartości typu integer.

Uwaga: W danym wyrażeniu analogowym nie można łącznie stosować zmiennych analogowych typu integer i real ani wyrażenia stałego.

B.2.8.5 Zmienne typu Timer

Timer oznacza **zegar** lub **licznik**. Zmienne takie przyjmują wartości czasowe i są zazwyczaj używane w wyrażeniach czasowych. Wartość czasowa nie może przekraczać **23h59m59s99** i nie może być ujemna. Zmienne typu timer są przechowywane w słowach 32-bitowych. Wewnętrznie są reprezentowane przez dodatnią liczbę milisekund.

Uwaga: Zmienne typu timer nie mogą posiadać atrybutów WEJŚCIA czy WYJŚCIA.

Zmienne typu timer mogą być odświeżane automatycznie przez system ISaGRAF. Po **uaktywnieniu** timera jego wartość jest zwiększana automatycznie, zgodnie z zegarem czasu rzeczywistego systemu oprogramowania sterownika. Do sterowania timerem można użyć następujące wyrażenia języka **ST**:

TSTART	uruchamia automatyczne odświeżanie timera
TSTOP	zatrzymuje automatyczne odświeżanie timera

B.2.8.6 Zmienne typu łańcuch / komunikat

Zmienne typu komunikat lub łańcuchy zawierają ciągi znaków. Długość takiego ciągu może zmieniać się podczas operacji. Długość zmiennej nie może przekraczać długości maksymalnej, określonej przy deklarowaniu zmiennej. Długość komunikatu ograniczona jest do 255 znaków. Zmienne typu komunikat mogą posiadać jeden z następujących **atrybutów**:

Wewnętrzna: zmienna w pamięci aktualizowana przez program
Wejściowa: zmienna skojarzona z urządzeniem wejściowym (odświeżana przez system)
Wyjściowa: zmienna skojarzona z urządzeniem wyjściowym

Zmienne w postaci łańcuchów mogą zawierać dowolny znak ze standardowej tablicy ASCII (kod ASCII od **0** do **255**). Łańcuchy znaków może zawierać znak pusty. Niektóre funkcje „C" standardowej biblioteki ISaGRAF nie będą prawidłowo przetwarzać komunikatów zawierających znaki puste (**0**).

B.2.9 Komentarze

Komentarze można wstawiać dowolnie w językach tekstowych takich, jak **ST** i **IL**. Komentarz musi rozpoczynać się znakami specjalnymi „**(***" i kończyć się znakami „***)**". Komentarze mogą być wstawiane w dowolnym programie **ST** i mogą zajmować więcej niż jedną linię.
Oto przykład komentarzy:

```
counter := ivalue;   (* przydziela licznik główny *)
(* to jest komentarz
zajmujący dwie linie *)
c := counter (* komentarze można wstawiać w dowolnym miejscu *) + base_value +
1;
```

Nie można używać komentarzy przeplatanych. Oznacz to, że wewnątrz komentarza nie mogą być użyte znaki „**(***".

Uwaga: Język IL akceptuje jedynie komentarze stanowiące ostatni składnik linii instrukcji.

B.2.10 Zdefiniowane słowa

System ISaGRAF pozwala na ponowne definiowanie wyrażeń stałych, prawdziwych i fałszywych wyrażeń binarnych, słów kluczowych oraz wyrażeń złożonych **ST**. Aby tego dokonać, należy odpowiedniemu wyrażeniu nadać **identyfikator**. Na przykład:

```
TAK     to    PRAWDA
PI      to    3.14159
OK      to    (tryb_automatyczny AND NOT (alarm))
```

Po zdefiniowaniu takiej równoważności, **identyfikator** może być używany w dowolnym miejscu programu **ST** zamiast odpowiadającego mu wyrażenia. Oto przykład programu w **ST** z użyciem wartości zdefiniowanych:

```
If OK Then
  angle := PI / 2;
  isdone := YES;
End_if;
```

Zdefiniowane słowa mogą być **LOKALNE** dla jednego programu, **GLOBALNE** lub **WSPÓLNE**.
Zdefiniowane słowa lokalne mogą być wykorzystywane tylko przez jeden program.
Zdefiniowane słowa globalne mogą być używane przez dowolny program projektu.
Zdefiniowane słowa wspólne mogą być używane w dowolnym programie projektu.
Proszę zauważyć, że zdefiniowane słowa wspólne mogą być zapisane osobno za pomocą Menedżera archiwizacji.

Uwaga: Kiedy ten sam identyfikator zostanie zdefiniowany podwójnie w różnych równoważnościach w **ST**, to stosowane jest ostatnie zdefiniowane wyrażenie. Na przykład:

```
Zdefiniowanie:   OTWARTY   to   FAŁSZ
                 OTWARTY   to   PRAWDA

oznacza:         OTWARTY   to   PRAWDA
```

Nadawanie nazw słowom definiowanym musi być zgodne z następującymi zasadami:
- nazwa nie może przekraczać **16** znaków
- pierwszym znakiem musi być **litera**
- kolejnymi znakami mogą być **litery**, **cyfry** lub znak podkreślenia ('_')

Uwaga: Słowo zdefiniowane nie może powoływać się w swej definicji na inne słowo zdefiniowane, na przykład niemożliwe jest ustalenie, że:

```
PI      to    3.14159
PI2     to    PI*2
```

Trzeba zapisać całą równoważność przy użyciu stałych lub zmiennych i operacji:

```
PI2     to    6.28318
```

B.3 Język SFC

Graf Funkcji Sekwencyjnych (SFC) to język **graficzny** wykorzystywany do opisu **operacji sekwencyjnych**. Proces jest reprezentowany w formie zestawu dobrze zdefiniowanych **kroków**, połączonych **przejściami**. Do każdego przejścia dołączony jest **warunek logiczny**. **Operacje** w ramach kroków są szczegółowo opisywane przy użyciu innych języków (**ST**, **IL**, **LD** i **FDB**).

B.3.1 Główny format schematu SFC

Program SFC to graficzny zestaw **kroków** i **przejść**, połączonych ze sobą **łączami zorientowanymi**. Do przedstawienia zbieżności i rozbieżności stosuje się łącza wielokrotne. Niektóre części pełnego programu mogą być wydzielone i reprezentowane w schemacie głównym przy pomocy pojedynczego symbolu, zwanego **krokiem makro**. Podstawowe **zasady graficzne** SFC, to:

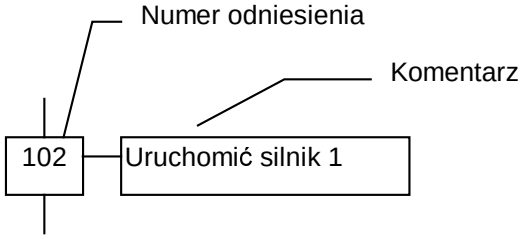
- Po kroku może następować kolejny krok
- Po przejściu nie może występować kolejne przejście

B.3.2 Podstawowe komponenty SFC

Podstawowymi komponentami (symbolami graficznymi) języka SFC są: kroki i kroki początkowe, przejścia, łącza zorientowane oraz skoki do kroku.

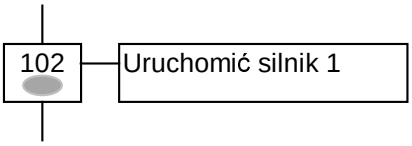
B.3.2.1 Kroki i kroki początkowe

Krok jest reprezentowany przy pomocy pojedynczego **kwadratu**. Do każdego kroku przyporządkowany jest **numer odniesienia** inny, wpisany w symbolu kwadratu kroku. Główny opis kroku wpisany jest w prostokącie połączonym z symbolem kroku. Opis ten, to **dowolny komentarz** (nie stanowiący części języka programowania). Powyższe informacje zwane są **Poziomem 1** kroku:

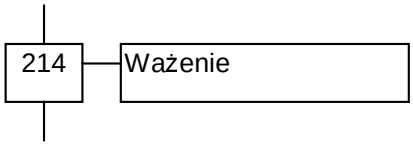


Podczas wykonywania kroku, **aktywność** kroku jest wskazywana przy pomocy **znacznika**:

Krok aktywny:

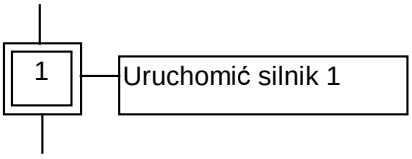


Krok nieaktywny:



Sytuacja początkowa programu SFC wyrażona jest przy pomocy **kroków początkowych**. Symbol graficzny kroku początkowego otoczony jest **podwójną ramką**. W każdym kroku początkowym przy jego uruchomieniu jest automatycznie umieszczany znacznik.

Krok początkowy:



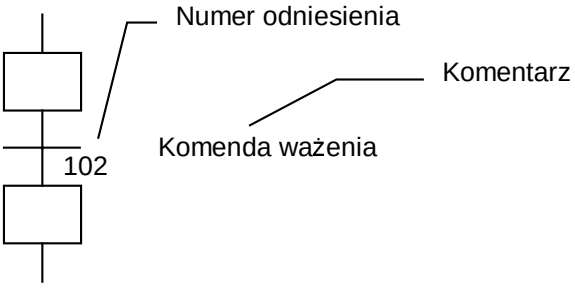
Program SFC musi zawierać **przynajmniej jeden** krok początkowy. Poniżej przedstawiono atrybuty każdego kroku. Pola takie mogą być wykorzystane w każdym z pozostałych języków:

GSnnn.x..... aktywność kroku (wartość binarna)
GSnnn.t..... czas trwania aktywności kroku (wartość czasowa)

(gdzie **nnn** to numer odniesienia kroku)

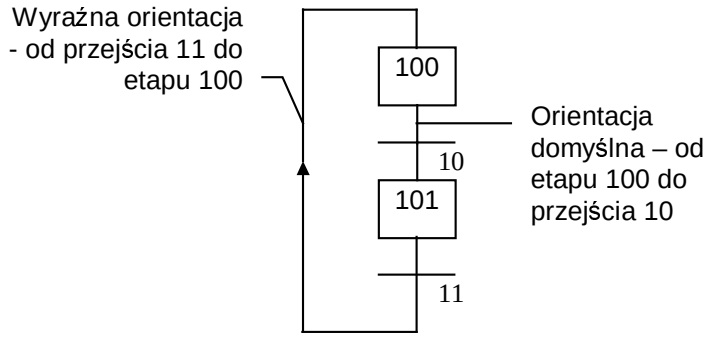
B.3.2.2 Przejścia

Przejścia reprezentowane są przy pomocy niewielkich poziomych pasków, przecinających łącza. Każde przejście jest **oznaczone** numerem, znajdującym się obok symbolu przejścia. Opis ten, to **dowolny komentarz** (nie stanowiący części języka programowania). Powyższe informacje zwane są **Poziomem 1** kroku:



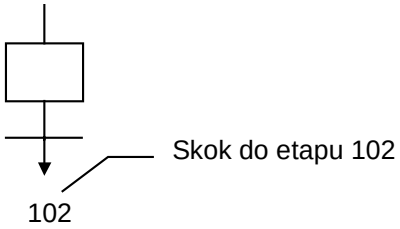
B.3.2.3 Zorientowane łącza

Do połączenia kroków i przejść używane są pojedyncze linie. Są to łącza zorientowane. Jeśli orientacja nie jest wyraźnie określona, łącze prowadzi z góry do dołu.

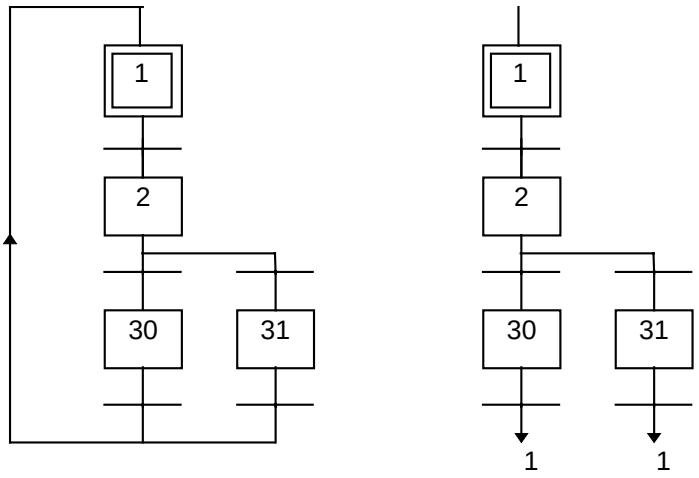


B.3.2.4 Skok do kroku

Symbole skoku mogą być używane do wskazania łącza od przejścia do kroku, bez konieczności rysowania linii łączącej. Symbol skoku musi być oznaczony numerem kroku docelowego:



Symbol skoku nie może zostać użyty do przedstawienia połączenia od kroku do przejścia. Przykład skoków - poniższe schematy odpowiadają sobie nawzajem:

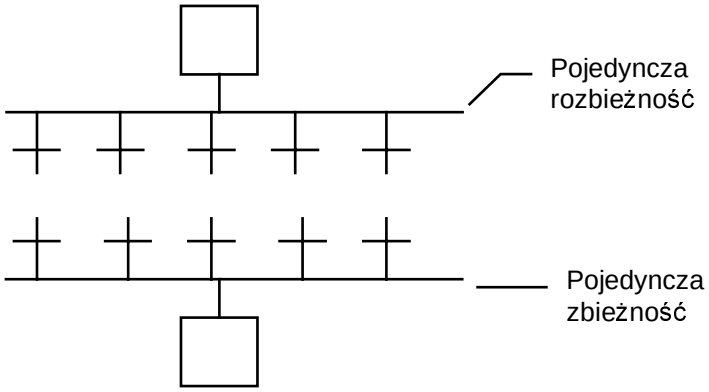


B.3.3 Rozbieżności i zbieżności

Rozbieżności, to **wielokrotne łącza** pomiędzy jednym symbolem SFC (krokiem lub przejściem), a wieloma innymi symbolami SFC. Zbieżności, to wielokrotne łącza pomiędzy więcej niż jednym symbolem SFC, a jednym z innych symboli. Rozbieżności i zbieżności mogą być pojedyncze lub podwójne.

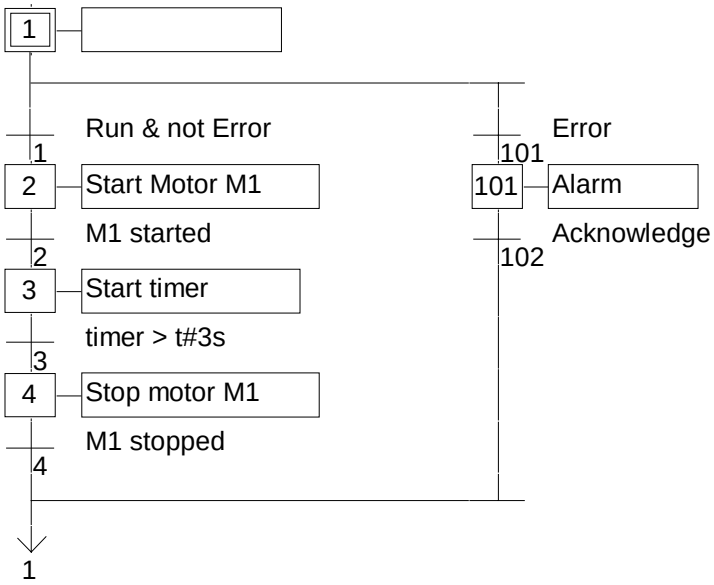
B.3.3.1 Rozbieżności pojedyncze

Pojedyncza rozbieżność, to łącze wielokrotne pomiędzy jednym krokiem, a wieloma przejściami. Pozwala ona przekazywać znacznik aktywny do jednej z wielu gałęzi. Zbieżność pojedyncza, to łącze wielokrotne pomiędzy wieloma przejściami jednego kroku. Zbieżność pojedyncza jest na ogół wykorzystywana do grupowania gałęzi SFC, rozpoczynających się od rozbieżności pojedynczej. Rozbieżności i zbieżności pojedyncze są reprezentowane przy pomocy pojedynczych linii poziomych.



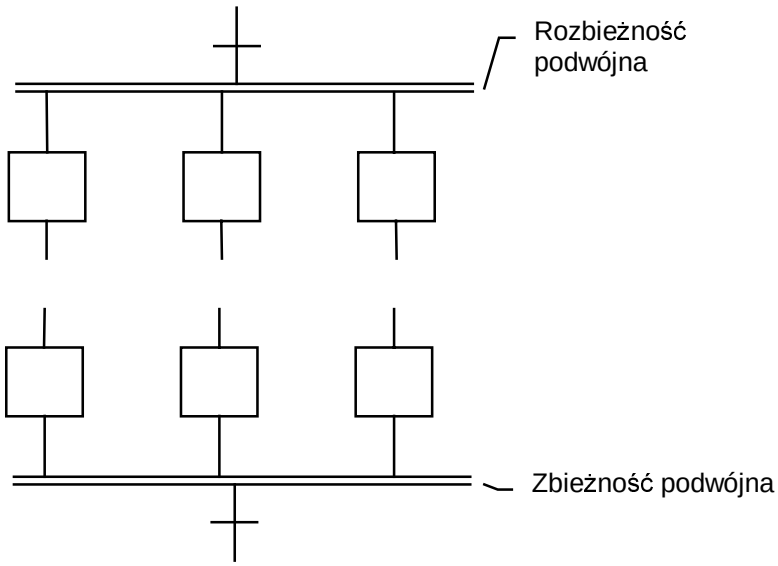
Uwaga: Warunki dołączone do różnych przejść na początku rozbieżności pojedynczej **domyślnie nie wyłączają się wzajemnie**. Wyłączność musi zostać wyraźnie wyszczególniona w warunkach przejść, aby zapewnić, iż w czasie wykonywania tylko jeden znacznik będzie przechodzić przez gałąź rozbieżności. Poniżej przedstawiono przykład pojedynczej rozbieżności i zbieżności:

(* Program SFC z pojedynczą rozbieżnością i zbieżnością *)



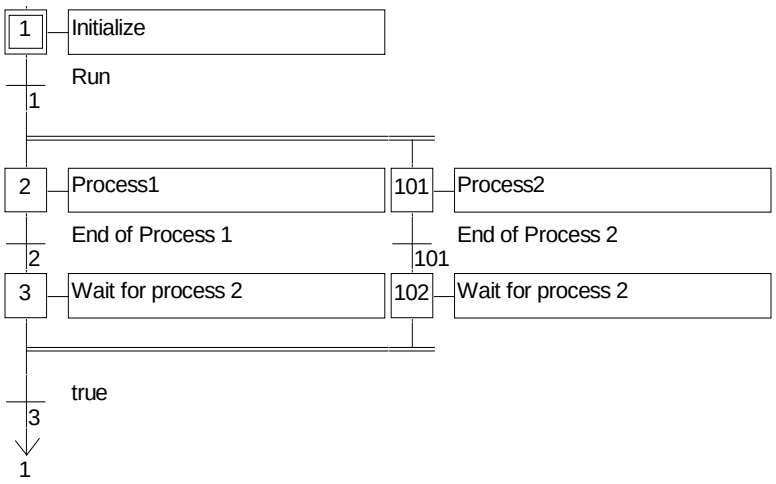
B.3.3.2 Rozbieżności podwójne

Podwójna rozbieżność, to łącze wielokrotne pomiędzy jednym krokiem, a wieloma przejściami. Odpowiada ona operacjom równoległym procesu. Zbieżność podwójna, to łącze wielokrotne pomiędzy wieloma krokami, a jednym przejściem Zbieżność podwójna jest na ogół wykorzystywana do grupowania gałęzi SFC, rozpoczynających się od rozbieżności podwójnej. Rozbieżności i zbieżności podwójne są reprezentowane przy pomocy podwójnych linii poziomych.



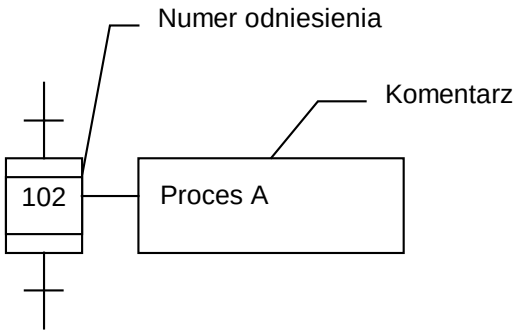
Przykład rozbieżności i zbieżności podwójnej:

(*Program SFC z podwójną rozbieżnością i zbieżnością *)



B.3.4 Etapy typu makro

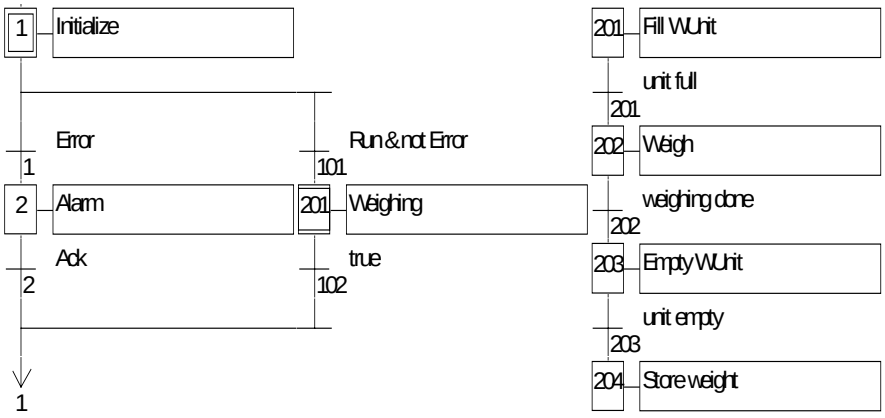
Etap typu makro, to nie powtarzająca się reprezentacja samodzielnej grupy kroków i przejść. Treść kroku makro jest opisana osobno, w innym miejscu tego samego programu SFC. Pojawia się on jako pojedynczy symbol na schemacie głównym SFC. Oto symbol używany do oznaczania kroku makro:



Numer odniesienia wpisany w symbolu kroku makro, to numer odniesienia pierwszego kroku w treści kroku makro. Krok typu makro musi rozpoczynać się **krokiem rozpoczynającym**, a kończyć **krokiem kończącym**. Schemat musi być samodzielny. Krok rozpoczynający nie posiada łącza górnego (brak przejścia wstecz). Krok kończący nie posiada łącza dolnego (brak przejścia naprzód). Symbol kroku makro może zostać umieszczony w treści innego kroku makro.

Uwaga: Ze względu na fakt, iż krok makro, to **niepowtarzalny** zestaw kroków i przejść, jeden krok makro nie może być użyty więcej niż raz w programie SFC.

Przykład kroku typu makro:
(* Schemat główny *) **(***
Treść kroku typu makro *)



B.3.5 Operacje w ramach kroków

Poziom 2 kroku SFC, to szczegółowy opis **operacji**, wykonywanych **w czasie aktywności kroku**. Opis ten jest sporządzany przy użyciu **funkcji tekstowych SFC** oraz innych języków , takich jak Tekst Strukturalny (**ST**). Podstawowe typy operacji, to:

Kilka operacji (tego samego lub różnych typów) może zostać opisanych w jednym kroku. Funkcje specjalne, które umożliwiają wykorzystanie dowolnego innego języka, to:

B.3.5.1 Operacje binarne

Operacje binarne przypisują zmienną binarną do operacji kroku. Zmienna binarna może być zmienną wyjściową lub wewnętrzną. Jest ona przypisywana za każdym razem, gdy operacja kroku rozpoczyna się lub kończy. Oto składnia podstawowych operacji binarnych:

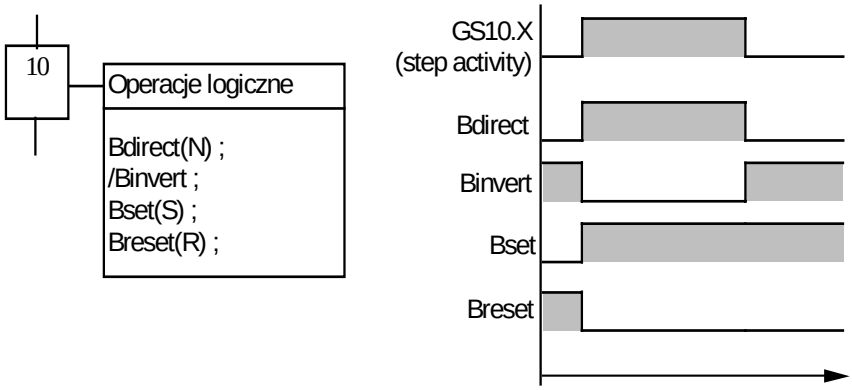
<zmienna_binarna> (N);	dołącza do zmiennej sygnał aktywności kroku
< zmienna_binarna>;	taki sam skutek (atrybut N jest opcjonalny)
/ < zmienna_binarna>;	dołącza do zmiennej negację sygnału aktywności kroku

Inne funkcje pozwalają ustawiać lub kasować zmienną binarną, kiedy krok uaktywnia się. Oto składnia operacji binarnych ustawiania i kasowania:

< zmienna_binarna> (S); ustawia zmienną na wartość PRAWDA, kiedy sygnał aktywności kroku ma wartość PRAWDA

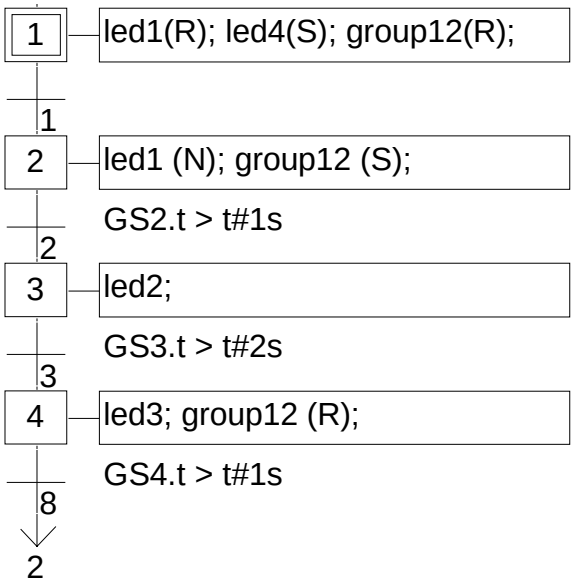
< zmienna_binarna> (R); kasuje wartość zmiennej na FAŁSZ, kiedy sygnał aktywności kroku ma wartość PRAWDA

Zmienna binarna musi być zmienną WYJŚCIOWĄ lub WEWNĘTRZNĄ. Poniższe programowanie SFC prowadzi do następujących zachowań:



Przykład operacji binarnych:

(* Program SFC wykorzystujący operacje BINARNE *)

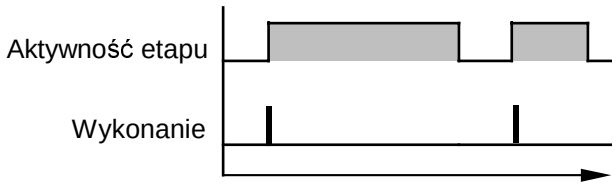


B.3.5.2 Operacje impulsowe

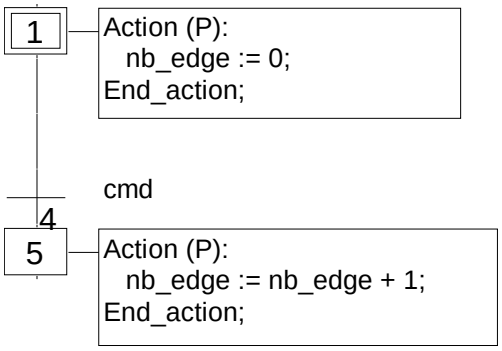
Operacja impulsowa, to zestaw instrukcji ST lub IL, które wykonywane są tylko **raz** w momencie **aktywizacji** kroku. Instrukcje zapisywane są zgodnie z następującą składnią SFC:

```
ACTION (P) :
        (* instrukcje ST *)
END_ACTION ;
```

Poniżej przedstawiono wyniki działania operacji impulsowej:



Przykład operacji impulsowej:

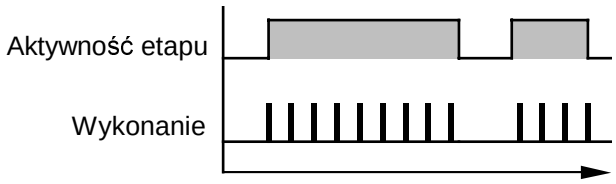


B.3.5.3 Operacje nie przechowywane

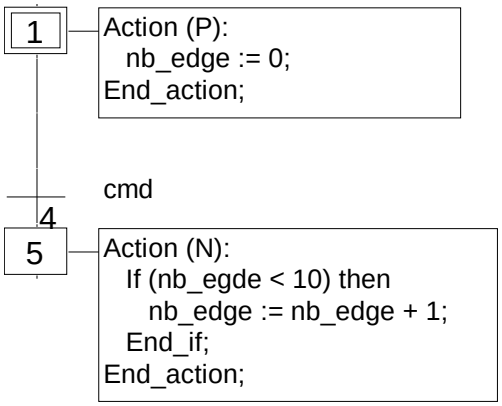
Operacja nie przechowywana (normalna), to lista instrukcji ST lub IL, które wykonywane są **w każdym cyklu** podczas całego okresu **aktywności** kroku. Instrukcje pisane są zgodnie z następującą składnią SFC:

```
ACTION (N) :
        (* instrukcje ST *)
END_ACTION ;
```

Oto wynik operacji nie przechowywanej:



Przykład operacji nie przechowywanej:



B.3.5.4 Operacje SFC

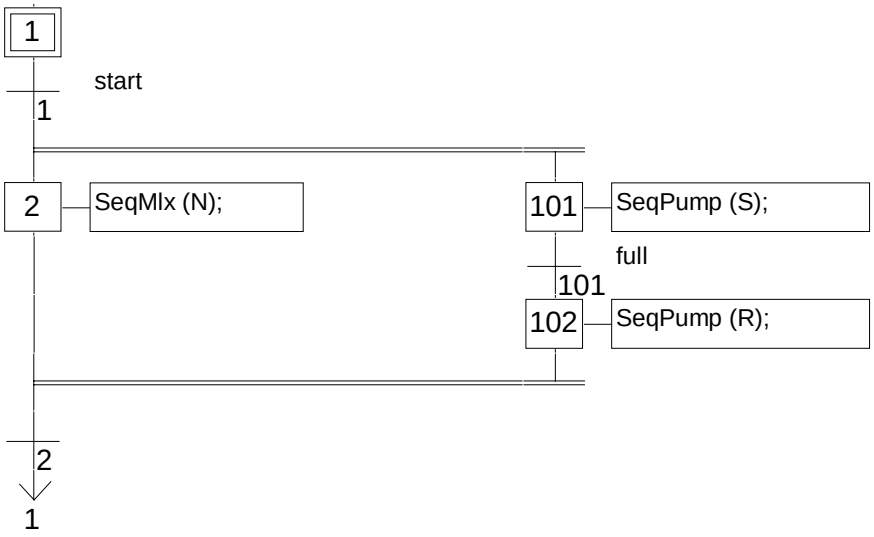
Operacja SFC, to sekwencja potomna SFC, uruchamiana lub zamykana w zależności od zmiany sygnału aktywności kroku. Operacja SFC może posiadać kwantyfikator **N** (Nie przechowywana), **S** (Ustaw) lub **R** (Kasuj). Oto składnia podstawowych działań SFC:

<program_potomny> (N);	uruchamia sekwencje potomną kiedy krok uaktywnia się i zamyka sekwencje potomną, gdy krok staje się nieaktywny
< program_potomny>;	taki sam skutek (atrybut N jest opcjonalny)
< program_potomny> (S);	uruchamia sekwencje potomną kiedy krok uaktywnia - gdy krok staje się nieaktywny, nic się nie dzieje
< program_potomny> (R);	zamyka sekwencje potomną kiedy krok uaktywnia - gdy krok staje się nieaktywny, nic się nie dzieje

Sekwencja SFC określona jako operacja, musi być **programem potomnym SFC** aktualnie edytowanego programu. Proszę zauważyć, iż wykorzystanie kwantyfikatorów **S** (Ustaw) lub **R** (Kasuj) operacji SFC odnosi dokładnie taki sam skutek, jak instrukcje **GSTART** i **GKILL**, zaprogramowane w operacji impulsowej **ST**.

Poniżej przedstawiono przykład operacji SFC. Program główny SFC nazwany został **Father**. Posiada on dwa programy potomne SFC, nazwane **SeqMlx** oraz **SeqPump**. Program-rodzic SFC tworzony jest następująco:

(* Program SFC wykorzystujący operacje SFC *)



B.3.5.5 Wywołania funkcji lub bloku funkcyjnego z poziomu operacji

Podprogramy, funkcje czy bloki funkcyjne (napisane w języku ST, IL, LD lub FBD), czy też funkcje "C" i bloki funkcyjne "C" mogą być bezpośrednio wywoływane z bloku operacji SFC, w oparciu o następującą składnię:

W przypadku podprogramów, funkcji i funkcji "C":

```
ACTION (P) :
        result := sub_program ( ) ;
END_ACTION;
```

lub

```
ACTION (N) :
        result := sub_program ( ) ;
END_ACTION;
```

W przypadku bloków funkcyjnych w "C" lub w ST, IL, LD, FBD:

```
ACTION (P) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
```

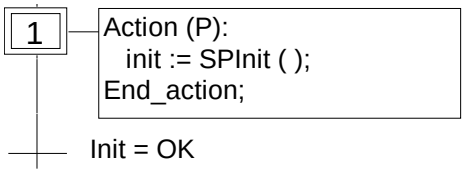
```
END_ACTION;
```

lub

```
ACTION (N) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
END_ACTION;
```

Szczegółowa składnia znajduje się w sekcji poświęconej językowi ST.
Przykład wywołania podprogramu w blokach operacji:

(* Program SFC z odwołaniem do podprogramu w bloku operacji *)



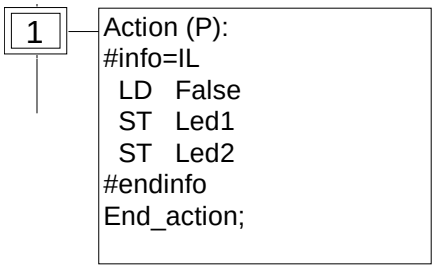
B.3.5.6 Konwencja IL

Program Listy Instrukcji (IL) może zostać włączony bezpośrednio do bloku operacji SFC, w oparciu o następującą składnię:

```
ACTION (P) :            (* lub N *)
#info=IL
   <instruction>
   <instruction>
   ....
#endinfo
END_ACTION;
```

Specjalne słowa kluczowe "#info=IL" i "#endinfo" muszą zostać wpisane dokładnie w ten sposób i **uwzględniają wielkość liter**. Znaków spacji lub tabulacji nie można wstawiać wewnątrz, po czy przed tymi słowami kluczowymi. Poniżej przedstawiono przykład programu IL w bloku operacji:

(* Program SFC z sekwencją IL w bloku operacji *)



B.3.6 Warunki dołączone do przejść

Do każdego przejścia dołączone jest **wyrażenie binarne**, które warunkuje czyszczenie przejścia. Warunek ten jest zazwyczaj wyrażony w języku ST lub przy użyciu języka LD (edytora LD). Jest to **Poziom 2** przejścia. Można jednak zastosować inne konstrukcje:

Konwencja ST
Konwencja LD
Konwencja IL
Wywoływanie funkcji z poziomu przejścia

Uwaga: Jeśli do przejścia nie jest dołączony warunek, warunkiem domyślnym jest **PRAWDA.**

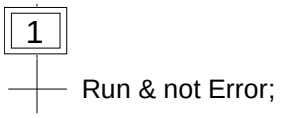
B.3.6.1 Konwencja ST

Język **Tekst Strukturalny** (ST) można wykorzystać do opisu warunku dołączonego do przejścia. Pełne wyrażenie musi być typu **binarnego** i musi być zakończone **średnikiem**, zgodnie z poniższą składnią:

< wyrażenie_binarne > ;

Wyrażenie może być wyrażeniem stałym Prawda lub FAŁSZ pojedynczym wejściem lub wewnętrzną zmienną binarną albo kombinacją zmiennych, prowadzących do wartości binarnej. Poniżej przedstawiono przykład programowania przejść w ST:

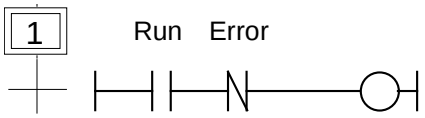
(* Program SFC program z przejściami zaprogramowanymi w ST *)



B.3.6.2 Konwencja LD

Język **Diagram Drabiniasty** (LD) może być wykorzystany do opisu **warunku** dołączonego do przejścia. Diagram taki składa się z tylko jednego szczebla z jedną cewką. Wartość cewki reprezentuje wartość przejścia.

Poniżej przedstawiono przykład przejść zaprogramowanych w LD:



B.3.6.3 Konwencja IL

Program typu Lista Instrukcji (IL) można zastosować bezpośrednio do opisu przejścia SFC , zgodnie z następującą składnią:

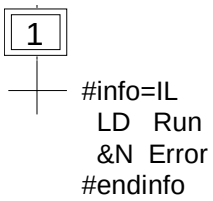
#info=IL
<instruction>
<instruction>
....
#endinfo

Wartość przedstawiana przez **wynik bieżący** (rejestr IL) pod koniec sekwencji IL powoduje dołączenie do przejścia wynikającego z niej warunku:

wynik bieżący = 0	→	warunek jest **FAŁSZEM**
wynik bieżący <> 0	→	warunek jest **PRAWDĄ**

Specjalne słowa kluczowe "#info=IL" i "#endinfo" muszą zostać wpisane dokładnie w ten sposób i **uwzględniają wielkość liter**. Znaków spacji lub tabulacji nie można wstawiać wewnątrz, po czy przed tymi słowami kluczowymi. Poniżej przedstawiono przykład programu IL dla przejść:

(* Program SFC zawierający program IL dla przejść *)



B.3.6.4 Wywołanie funkcji z poziomu przejścia

Dowolny podprogram lub funkcja (napisana w języku FBD, LD, ST lub IL), czy też funkcja "C" może zostać wywołana w celu oceny warunku dołączonego do przejścia, zgodnie z następującą składnią:

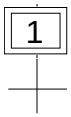
< podprogram > ( ) ;

Wartość zwracana przez podprogram lub funkcje musi być wartością binarną, zwracającą wynikający z niej warunek:

wartość zwracana = 0	→	warunek jest **FAŁSZEM**
wartość zwracana <> 0	→	warunek jest **PRAWDĄ**

Poniżej przedstawiono przykład podprogramu wywoływanego w przejściu:

(* Program SFC z wywołaniem podprogramu dla przejścia *)



B.3.7 Zasady dynamiczne SFC

Pięć zasad dynamicznych języka SFC, to:

Sytuacja początkowa

Sytuacja początkowa charakteryzuje się **krokami początkowymi**, które są z definicji aktywne na początku operacji. W każdym programie SFC musi być obecny p**rzynajmniej jeden** krok początkowy.

Czyszczenie przejścia

Przejście może być **włączone** lub **wyłączone**. Włączone jest wtedy, gdy wszystkie bezpośrednio poprzedzające go kroki, połączone z odpowiednimi symbolami przejścia są **aktywne**. W przeciwnym wypadku jest ono wyłączone. Przejście nie może zostać **wyczyszczone**, chyba że:

- jest włączone oraz
- warunek powiązanego przejścia to prawda.

Zmiana stanu kroków aktywnych

Wyczyszczenie przejścia prowadzi równocześnie do uaktywnienia stanu kroków następujących bezpośrednio po nim i dezaktywacji stanu kroków bezpośrednio go poprzedzających.

Równoczesne czyszczenie przejść

Podwójne linie mogą zostać wykorzystane do wskazania przejść, które mają być czyszczone równocześnie. Jeśli przejścia takie są przedstawione osobno, stan aktywności kroków poprzedzających (GSnnn.x) może zostać wykorzystany do wyrażenia ich warunków.

Równoczesne uaktywnienie i dezaktywizacja kroku

Jeżeli podczas operacji krok jest równocześnie aktywizowany i deaktywowany, aktywizacja posiada priorytet.

B.3.8 Hierarchia programu SFC

System ISaGRAF umożliwia opis struktury pionowej programów SFC. Programy SFC są zorganizowane w formie **drzewa hierarchii**. Każdy program SFC może sterować

(uruchamiać, zamykać...) innymi programami SFC. Programy takie zwane są **programami potomnymi** programu SFC, który nimi steruje. Programy SFC są powiązane ze sobą w ramach głównego **drzewa hierarchii**, wykorzystując relacje "**rodzic - potomek**".

Podstawowe zasady narzucane przez strukturę hierarchiczną, to:

- Programy SFC, które nie posiadają rodzica zwane są programami "**głównymi**" SFC
- Programy główne SFC są uaktywniane przez system, kiedy uruchamiana jest aplikacja
- Program może posiadać kilka programów potomnych
- Program potomny nie może mieć więcej niż jednego rodzica
- Program potomny może być sterowany jedynie przez swego rodzica
- Program nie może sterować programami potomnymi jednego ze swych własnych programów potomnych

Podstawowe operacje, które może wykonać program-rodzic SFC w celu sterowania swym programem potomnym, to:

Uruchom	(**GSTART**) Uruchamia program potomny: aktywizuje każdy z jego kroków wstępnych. Programy potomne tego programu potomnego nie są uruchamiane automatycznie.
Zakończ	(**GKILL**) Zamyka program potomny deaktywując wszystkie z jego kroków aktywnych. Wszystkie programy potomne tego programu potomnego są również zamykane.
Zamroź	(**GFREEZE**) Zawiesza wykonywanie programu (deaktywuje operacje każdego z kroków aktywnych i zawiesza przeliczanie przejść) oraz zapamiętuje status kroków programu, aby można go było uruchomić ponownie. Wszystkie programy potomne tego programu potomnego są również wstrzymywane.
Uruchom ponownie	(**GRST**) Uruchamia ponownie wstrzymany program SFC, reaktywując zawieszone kroki. Wszystkie programy potomne tego programu potomnego nie są automatycznie uruchamiane ponownie.
Podaj status	(**GSTATUS**) Uzyskuje bieżący status (aktywny, nieaktywny lub wstrzymany) programu potomnego.

B.4 Język FC

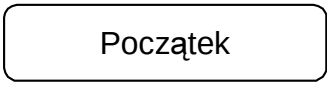
Flow Chart (**FC**) to język graficzny używany do opisu **operacji sekwencyjnych**. Graf przepływowy tworzą **Operacje** i **Testy**. Pomiędzy operacjami a testami występują **połączenia zorientowane,** reprezentujące przepływ danych. Operacje i Testy mogą być opisywane językami ST, LD lub IL. Funkcje i bloki funkcyjne dowolnego języka (poza SFC) mogą być wywoływane z operacji lub testów. Program Flow Chart może wywoływać inny program Flow Chart. Wywoływany program FC jest **podprogramem** programu wywołującego FC.

B.4.1 Komponenty FC

Poniżej przedstawiono komponenty graficzne języka Flow Chart:

Rozpoczynanie grafu FC

Na początku programu Flow Chart musi znajdować się symbol "**początek**". Jest on niepowtarzalny i nie może zostać pominięty. Reprezentuje on stan początkowy grafu w momencie jego aktywacji. Poniżej znajduje się rysunek symbolu "początek":



Symbol "Początek" zawsze posiada połączenie (w dolnej części) z innymi obiektami grafu. Graf blokowy nie jest prawidłowy, jeśli z symbolu "Początek" nie jest poprowadzone połączenie z innym obiektem.

Kończenie grafu FC

Symbol "**koniec**" musi znajdować się na końcu programu Flow Chart. Jest on niepowtarzalny i nie może zostać pominięty. Możliwe jest, iż do symbolu "Koniec" nie jest poprowadzone żadne połączenie (graf z pętlą wieczną), jednak pomimo tego symbol "Koniec" musi zostać umieszczony na końcu grafu. Reprezentuje on stan końcowy grafu, kiedy jego wykonanie zostało zakończone. Poniżej znajduje się rysunek symbolu "koniec":

Symbol "Koniec" zazwyczaj posiada połączenie (w górnej części) z innymi obiektami grafu. Graf blokowy może nie posiadać połączenia z obiektem "Koniec" (graf z pętlą wieczną). Obiekt "koniec" jest w takim przypadku nadal widoczny w dolnej części grafu.

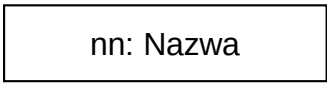
Połączenia FC

Połączenie to linia reprezentująca przepływ pomiędzy dwoma punktami diagramu. Połączenie jest zawsze zakończone strzałką. Poniżej znajduje się rysunek połączenia:

Dwa połączenia nie mogą być połączone z tym samym źródłowym punktem przyłączenia.

Operacje FC

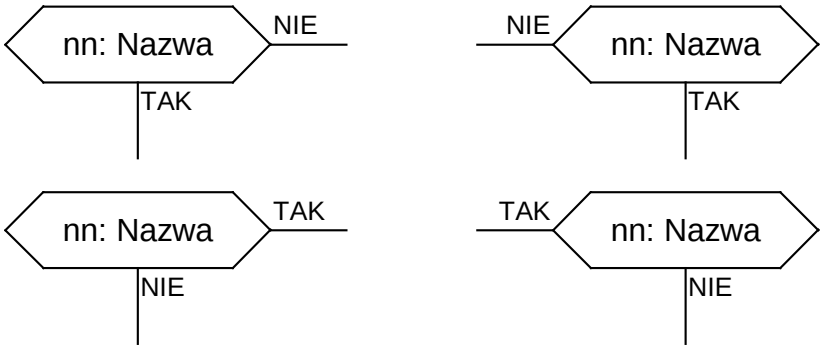
Symbol **operacji** reprezentuje działania, które mają zostać wykonane. Operacja jest identyfikowana przy pomocy numeru i nazwy. Poniżej znajduje się rysunek symbolu „operacji":



Dwa różne obiekty tego samego grafu nie mogą posiadać takiej samej nazwy ani numeru logicznego. Językiem programowania dla operacji może być ST, LD lub IL. Operacja jest zawsze połączona z łączami, jednym dochodzącym i jednym wychodzącym z niej.

Warunki FC

Warunek reprezentuje logiczny **test**. Warunek identyfikowany jest przy pomocy numeru i nazwy. W zależności od oceny powiązanego wyrażenia ST, LD lub IL, przepływ kierowany jest na ścieżkę "TAK" lub "NIE". Poniżej przedstawiono możliwe rysunki symbolu warunku:



Dwa różne obiekty tego samego grafu nie mogą posiadać takiej samej nazwy ani numeru logicznego. Program testu to:
- wyrażenie ST, lub
- pojedynczy szczebel w LD, bez symboli dołączonych do charakterystycznej cewki, lub
- kilka instrukcji w IL. Rejestr IL (lub aktualny wynik) jest używany do oceny warunku.

Przy programowaniu językiem tekstowym ST, wyrażenia mogą być opcjonalnie zakończone średnikiem. Przy programowaniu w LD, charakterystyczna cewka reprezentuje wartość warunku. Warunek równy:
- 0 lub FAŁSZ kieruje przepływ do NIE
- 1 lub PRAWDA kieruje przepływ do TAK

Test jest zawsze połączony z łączem przychodzącym i muszą zostać zdefiniowane oba połączenia w kierunku przekazywania.

Podprogram FC

System umożliwia opisywanie struktury pionowej programów FC. Programy FC są zorganizowane w **hierarchii drzewiastej**. Każdy program FC może wywołać inny program FC. Program taki zwany jest **programem potomnym** programu FC, który go wywołał. Programy FC wywołujące inne podprogramy FC są zwane **programami-rodzicami**. Programy FC są połączone w głównym drzewie hierarchii z wykorzystaniem relację rodzic-potomek:

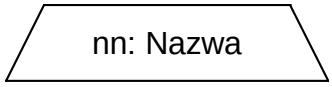
Symbol **podprogramu** w Flow Chart reprezentuje wywołanie podprogramu Flow Chart. Wykonywanie programu wywołującego FC jest zawieszane do momentu ukończenia wykonywania podprogramu. Podprogram Flow Chart jest identyfikowany za pośrednictwem numeru i nazwy, podobnie jak inne programy, funkcje czy bloki funkcyjne. Poniżej znajduje się rysunek symbolu "wywołania podprogramu":

Dwa różne obiekty tego samego grafu nie mogą posiadać takiej samej nazwy ani numeru logicznego. Podstawowe zasady wynikające z hierarchicznej struktury FC, to:
- Programy FC nie posiadające rodzica zwane są programami głównymi FC.
- Programy główne FC są aktywowane przez system podczas uruchomienia aplikacji
- Program może posiadać kilka programów potomnych
- potomek programu nie może mieć więcej niż jednego rodzica
- Program potomny może zostać wywołany jedynie przez swego rodzica
- Program nie może odwoływać się do potomków jednego ze swoich potomków

Ten sam podprogram może pojawiać się kilka razy na diagramie programu-rodzica. Wywołanie podprogramu Flow Chart reprezentuje pełne wykonanie grafu potomnego. Wykonywanie grafu-rodzica zostaje zawieszone podczas wykonywania grafu potomnego. Bloki wywołania podprogramu muszą być odpowiadać takim samym zasadom połączeń, jakie są określone dla operacji.

Operacja FC charakterystyczna dla We/Wy

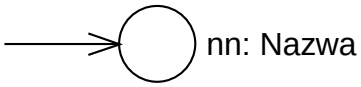
Symbol **operacji specyficznej dla We/Wy** reprezentuje działania, które mają zostać wykonane. Podobnie jak inne operacje, tak operacje charakterystyczne dla We/Wy są identyfikowane przy pomocy numeru i nazwy. W operacjach standardowych oraz operacjach charakterystycznych dla We/Wy stosuje się tę samą składnię. Celem operacji charakterystycznych We/Wy jest zwiększenie czytelności grafu i skupienie się na stałych elementach grafów. Wykorzystanie operacji charakterystycznych dla We/Wy stanowi funkcję opcjonalną. Poniżej przedstawiono rysunek symbolu "operacji specyficznej dla We/Wy":



Bloki specyficzne We/wy zachowują się dokładnie tak samo, jak operacje standardowe. Dotyczy to ich właściwości, programowania ST, LD lub IL oraz zasad łączenia.

Łączniki FC

Łączniki są wykorzystywane do reprezentowania połączenia pomiędzy dwoma punktami grafu bez łączenia ich. Łącznik jest reprezentowany przez koło i jest podłączony do źródła przepływu. Rysowanie łącznika odbywa się, po właściwej stronie (w zależności od kierunku przepływu danych), poprzez identyfikację punkty docelowego (zazwyczaj nazwy symbolu docelowego). Poniżej znajduje się standardowy rysunek łącznika:



Łącznik zawsze kieruje się na zdefiniowany symbol Flow Chart. Symbol docelowy identyfikowany jest przy pomocy numeru logicznego.

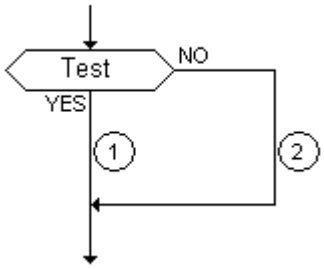
Komentarze FC

Blok **komentarza** zawiera tekst, bezsensowny z punktu widzenia składni. Można go wstawić w dowolnym, wolnym miejscu okna dokumentu Flow Chart, a używany jest on do dokumentowania programu. Poniżej znajduje się symbol "komentarza":

Tekst komentarza może zajmować kilka linii...

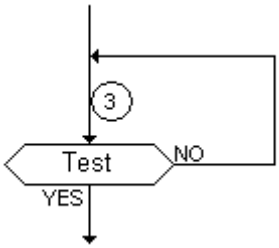
B.4.2 Przykłady struktur złożonych FC

Niniejsza sekcja prezentuje przykłady **struktur złożonej**, które można zdefiniować na diagramie Flow Chart. Struktury takie, to kombinacje połączonych ze sobą podstawowych obiektów.



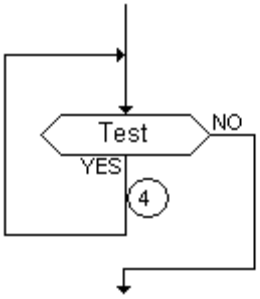
IF / THEN / ELSE

(1) umieść dla wstawianych operacji "THEN"
(2) umieść dla wstawianych operacji "ELSE"



REPEAT / UNTIL

(3) umieść dla wstawianych operacji powtarzanych



WHILE / DO

(4) umieść dla wstawianych operacji powtarzanych

B.4.3 Zachowanie dynamiczne FC

Wykonanie diagramu Flow Chart można wyjaśnić następująco:

- Symbol Początek zajmuje jeden cykl oprogramowania wbudowanego
- Symbol Koniec zajmuje jeden cykl oprogramowania wbudowanego i kończy wykonywanie grafu. Po dojściu do tego symbolu nie jest już wykonywana żadna operacja grafu.
- Przepływ jest przerywany za każdym razem, gdy napotykany jest element (operacja, decyzja), który był już wykonywany w tym cyklu oprogramowania wbudowanego. W takim przypadku przepływ będzie kontynuowany w następnym cyklu.

Uwaga: W przeciwieństwie do SFC, operacja nie jest stanem stabilnym. Kiedy symbol operacji jest podświetlony nie występuje żadne powtarzanie instrukcji.

B.4.4 Kontrola FC

Poza dołączonymi programami ST, LD czy IL, do samego grafu przepływowego odnosi się kilka **zasad składniowych**. Poniżej wymieniono główne z nich:
- Wszystkie punkty "przyłączenia" wszystkich symboli muszą być przyłączone. (połączenie z symbolem "Koniec" może zostać pominięte)
- Wszystkie symbole muszą być powiązane ze sobą (nie powinna pojawić się żadna odizolowana część)
- Wszystkie łączniki powinny posiadać prawidłowe elementy docelowe

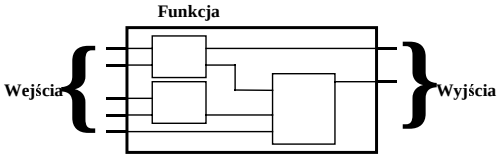
Zgłoszone mogą zostać również inne, drobne błędy składni:
- Operacje puste (bez programu) są uznawane za kroki podczas planowania realizacji
- Testy puste (bez programu) są uznawane za "zawsze prawdziwe"

B.5 Język FBD

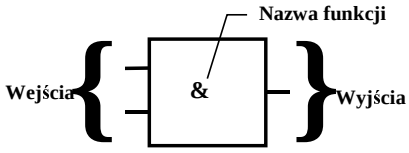
Diagram Bloków Funkcyjnych (FBD - Functional Block Diagram) to język graficzny. Pozwala on programiście budować złożone procedury poprzez wykorzystanie istniejących **funkcji** z biblioteki ISaGRAF oraz **zmontowanie** ich razem w obszarze schematu graficznego.

B.5.1 Główny format schematu FBD

Schemat FBD opisuje funkcję pomiędzy **zmiennymi wejściowymi** a **zmiennymi wyjściowymi**. Funkcję opisuje się jako zespół **elementarnych bloków funkcyjnych**. Zmienne wejściowe i wyjściowe są połączone z blokami **liniami łączącymi**. Wyjście bloku funkcyjnego może również być połączone z wejściem innego bloku.



Całość funkcji obsługiwanych przez program FBD jest budowana ze standardowych **elementarnych** bloków funkcyjnych z biblioteki ISaGRAF. Każdy blok funkcyjny posiada ustaloną ilość **podłączeń wejściowych** i ustaloną ilość **podłączeń wyjściowych**. Blok funkcyjny jest przedstawiany za pomocą pojedynczego **prostokąta**. Wejścia są doprowadzone do jego **lewej** krawędzi. Wyjścia są widoczne przy jego **prawej** krawędzi. Elementarny blok funkcyjny wykonuje pojedynczą **funkcję** pomiędzy swoimi wejściami i wyjściami. Nazwa wykonywanej przez blok funkcyjny operacji jest wpisana w jej prostokątnym symbolu. **Typ** każdego wejścia i wyjścia bloku jest odpowiednio zdefiniowany.



Zmienne wejściowe programu FBD muszą być połączone z podłączeniami wejściowymi bloków funkcyjnych. Typ każdej zmiennej musi być taki sam, jak typ oczekiwany na powiązanym z nią wejściu. Wejście wykresu FBD może być wyrażeniem **stałym**, dowolną zmienną **wewnętrzną** lub **wejściową** albo zmienną **wyjściową**.

Zmienne wyjściowe programu FBD muszą być połączone z podłączeniami wyjściowymi bloków funkcyjnych. Typ każdej zmiennej musi być taki sam, jak typ oczekiwany na

powiązanym z nią wyjściu bloku. Wyjście diagramu FBD może być dowolną zmienną **wewnętrzną** lub **wyjściową** albo nazwą programu (dotyczy tylko **podprogramów**). Kiedy wyjście to nazwa edytowanego aktualnie podprogramu, reprezentuje ono przypisanie wartości zwracanej tego podprogramu (zwracanej do programu wywołującego).

Zmienne wejściowe i wyjściowe, wejścia i wyjścia bloków funkcyjnych są połączone ze sobą **liniami łączącymi**. Pojedyncze linie mogą zostać użyte do **połączenia** dwóch punktów logicznych schematu:

- Zmiennej wejściowej i wejścia bloku funkcyjnego
- Wyjścia bloku funkcyjnego i wejścia innego bloku
- Wyjścia bloku funkcyjnego i zmiennej wyjściowej

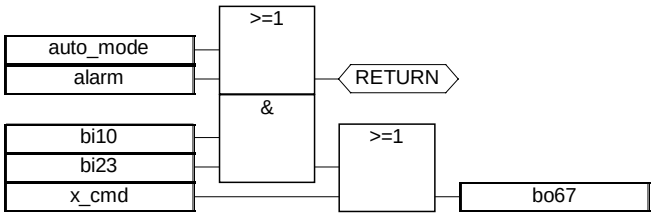
Połączenie to jest **zorientowane**, co oznacza, iż linia przenosi powiązane dane z lewego do prawego miejsca skrajnego. **Typ** prawego i lewego końca linii łączącej musi być **taki sam**.

Wielokrotne połączenia prawe mogą zostać wykorzystane do przesłania informacji z ich lewego końca do każdego z prawych końców. Typ wszystkich końców połączenia musi być taki sam.

B.5.2 Instrukcja RETURN

Słowo kluczowe "**<RETURN**>" może pojawiać się jako wyjście diagramu. Należy połączyć je z podłączeniem binarnym bloku funkcyjnego. Instrukcja RETURN stanowi **warunkowe zakończenie** programu: jeśli wyjście ramki połączonej z tą instrukcją posiada wartość binarną **PRAWDA**, końcówka (pozostała część) schematu nie jest wykonywana.

(* Przykład programu FBD wykorzystującego instrukcję RETURN *)




```
(* równoważność ST: *)
If auto_mode OR alarm Then
        Return;
End_if;
bo67 := (bi10 AND bi23) OR x_cmd;
```

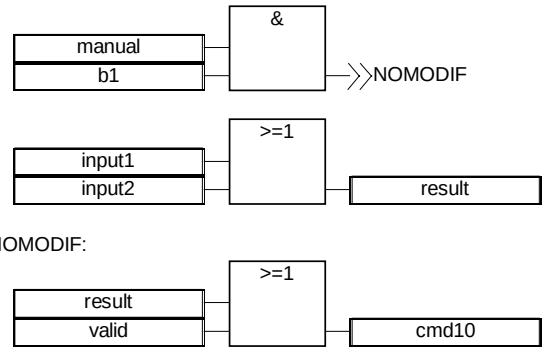
B.5.3 Skoki i etykiety

Etykiety i skoki są używane do sterowania wykonywaniem schematu. Po prawej stronie symbolu skoku lub etykiety nie może zostać podłączony żaden obiekt. Stosuje się następującą notację:

>>LAB ..... skok do etykiety (nazwa etykiety to "LAB")
LAB: ..... definicja etykiety (nazwa etykiety to "LAB")

Jeśli linia łącząca po **lewej** stronie symbolu skoku posiada wartość binarną **PRAWDA**, program wykonuje skok do miejsca bezpośrednio za odpowiednim symbolem etykiety.

(* Przykład programu FBD wykorzystującego etykiety i skoki *)

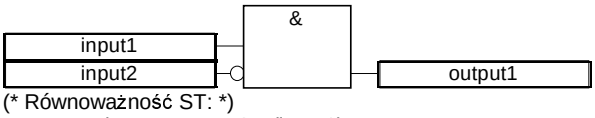



```
(* Równoważność IL: *)
                ld      manual
                and     b1
                jmpc    NOMODIF
                ld      input1
                or      input2
                st      result
NOMODIF:        ld      result
                and     valid
                st      cmd10
```

B.5.4 Negacja binarna

Pojedyncza linia łącząca o prawym końcu połączonym z wejściem bloku funkcyjnego może kończyć się **negacją binarna**. Negacja przedstawiana jest przy pomocy niewielkiego okręgu. Kiedy zastosowana jest negacja binarna, lewy i prawy koniec linii łączącej musi być typu **BINARNEGO**.

(* Przykład programu FBD wykorzystującego negację binarną*)

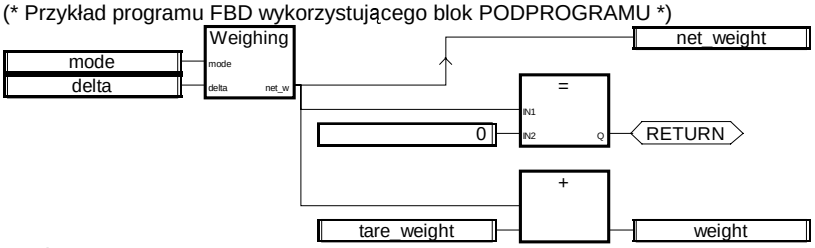



```
(* Równoważność ST: *)
output1 := input1 AND NOT (input2);
```

B.5.5 Wywoływanie funkcji lub bloku funkcyjnego z poziomu FBD

Język FBD pozwala na wywoływanie podprogramów, funkcji lub bloków funkcyjnych. Podprogram, funkcja lub blok funkcyjnych przedstawiany jest przy pomocy ramki funkcji. Nazwa wpisana w ramce to nazwa podprogramu, funkcji lub bloku funkcyjnego.
W przypadku podprogramów lub funkcji, wartość zwracana jest jedynym wyjściem z ramki funkcji.
Blok funkcyjny może posiadać więcej niż jedno wyjście.

(* Przykład programu FBD wykorzystującego blok PODPROGRAMU *)

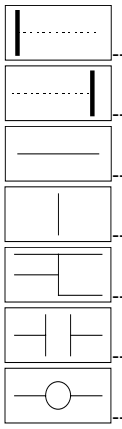


(* Równoważność ST *)
net_weight := Weighing (mode, delta); (* wywołanie podprogramu *)
If (net_weight = 0) Then Return; End_if;
weight := net_weight + tare_weight;

B.6 Język LD

Diagram Drabinkowy (Język stykowy) (LD) to graficzna reprezentacja równań binarnych, łączących **styki** (argumenty wejściowe) z **cewkami** (wynikami wyjściowymi). Język LD umożliwia opis testów i modyfikację danych **binarnych** przez umieszczanie w grafie programu **symboli graficznych**. Symbole graficzne LD są zorganizowane w grafie dokładnie tak, jak na elektrycznym schemacie połączeń. Diagramy LD są z prawej i lewej strony podłączone do pionowych **szyn zasilających**.

Oto podstawowe komponenty graficzne diagramu LD:



---------- Lewa pionowa szyna zasilająca

---------- Prawa pionowa szyna zasilająca

---------- Pozioma linia łącząca

---------- Pionowa linia łącząca

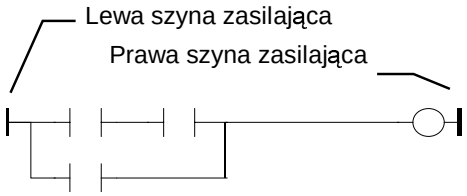
---------- Linie połączenia wielokrotnego (wszystkie połączone ze sobą)

---------- Styk powiązany ze zmienną

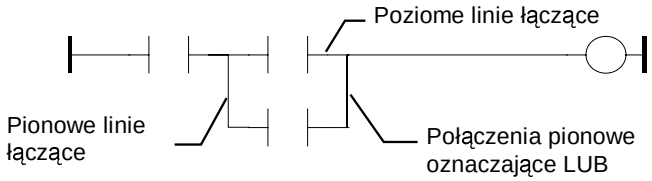
---------- Cewka powiązana ze zmienną wyjściową lub zewnętrzną

B.6.1 Szyny zasilające i linie łączące

Diagram LD jest zakończony z prawej i z lewej strony pionowymi liniami, nazywanych odpowiednio **prawą szyną zasilającą** i **lewą szyną zasilającą**.



Symbole graficzne diagramu LD są połączone z szynami zasilającymi lub innymi symbolami za pomocą **linii łączących**. Linie łączące mogą być poziome lub pionowe.



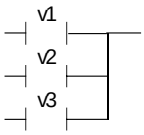
Każdemu segmentowi linii odpowiada stan **FAŁSZ** lub **PRAWDA**. Stan ten jest taki sam dla wszystkich połączonych ze sobą segmentów. Każda pozioma linia połączona z lewą **pionową szyną zasilającą** ma stan **PRAWDA**.

B.6.2 Połączenie wielokrotne

Stan logiczny przypisany pojedynczej linii poziomej jest taki sam na prawym i lewym końcu linii. Łączenie pionowych i poziomych linii łączących pozwala na budowanie **połączeń wielokrotnych**. Stan logiczny końców połączenia wielokrotnego zgodny jest z zasadami logiki.

Połączenie wielokrotne po lewej łączy **więcej niż jedną** linię poziomą doprowadzoną z **lewej** strony linii pionowej oraz **jedną** linię występującą po jej **prawej** stronie. Stan logiczny prawego końca to **LOGICZNE „LUB”** pomiędzy wszystkimi lewymi końcami.

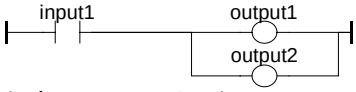
(* Przykład wielokrotnego łącza LEWEGO *)



(* stan końca prawego to (v1 LUB v2 LUB v3) *)

Połączenie wielokrotne po prawej łączy **jedną** linię poziomą połączoną z **lewą** stroną linii pionowej oraz **więcej niż jedną** linię połączoną z jej **prawą** stroną. Stan logiczny lewego końca jest rozprowadzany do każdego końca prawego.

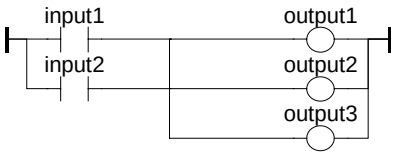
(* Przykład wielokrotnego łącza PRAWEGO *)




```
(* równoważność ST: *)
output1 := input1;
output2 := input1;
```

Połączenie wielokrotne po lewej i po prawej łączy **więcej niż jedną** linię poziomą doprowadzoną z **lewej** strony linii pionowej oraz **więcej niż jedną** linię występującą po jej **prawej** stronie. Stan logiczny każdego końca prawego to **LOGICZNE „LUB”** pomiędzy wszystkimi lewymi końcami

(* Przykład wielokrotnego łącza PRAWEGO i LEWEGO*)



(* Równoważność ST: *)
output1 := input1 OR input2;
output2 := input1 OR input2;
output3 := input1 OR input2;

B.6.3 Podstawowe styki i cewki LD

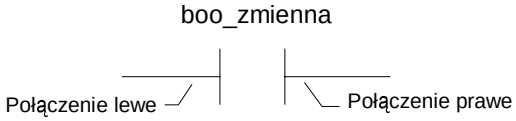
Istnieje kilka symboli umożliwiających oznaczenie styków wejściowych
-Styk bezpośredni
-Styk zanegowany
-Styki z wykrywaniem narastającego zbocza sygnału
-Styki z wykrywaniem opadającego zbocza sygnału

Istnieje kilka symboli umożliwiających oznaczenie cewek wyjściowych
-Cewka bezpośrednia
-Cewka zanegowana
-Cewka SET (ustaw)
-Cewka RESET (kasuj)
-Cewki z wykrywaniem narastającego zbocza sygnału
-Cewki z wykrywaniem opadającego zbocza sygnału

Nazwa zmiennej umieszczana jest ponad dowolnym z poniższych symboli graficznych.

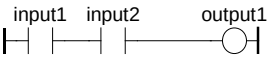
Styk bezpośredni

Styk bezpośredni umożliwia **operację binarną** pomiędzy stanem **linii łączącej** a **zmienną** binarną.



Stan linii łączącej po prawej stronie styku to **LOGICZNE „I”** pomiędzy stanem lewej linii łączącej a wartością zmiennej powiązanej ze stykiem.

(* Przykład wykorzystania styków BEZPOŚREDNICH *)



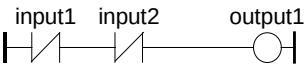

```
(* Równoważność ST: *)
output1 := input1 AND input2;
```

Styki zanegowane

Styk zanegowany umożliwia **operację binarną** pomiędzy stanem **linii łączącej** a logiczną negacją **zmiennej** binarnej.

Stan linii łączącej po prawej stronie styku to **LOGICZNE „I”** pomiędzy stanem lewej linii łączącej a **logiczną negacją** zmiennej powiązanej ze stykiem.

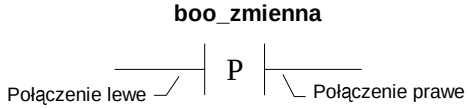
```
(* Przykład wykorzystania styków ZANEGOWANYCH *)
```




```
(* Równoważność ST: *)
output1 := NOT (input1) AND NOT (input2);
```

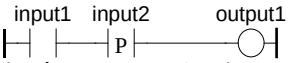
Styki z wykrywaniem narastającego zbocza sygnału

Poniższy styk (dodatni) umożliwia **operację binarną** pomiędzy stanem **linii łączącej** narastającym zboczem sygnału **zmiennej** binarnej.



Stan linii łączącej po prawej stronie styku ustawiany jest na **PRAWDA**, kiedy stan linii łączące po lewej stronie styku **PRAWDA**, a stan powiązanej z nim zmiennej **narasta** od FAŁSZ do PRAWDA. We wszystkich pozostałych przypadkach jest on zmieniany na FAŁSZ.

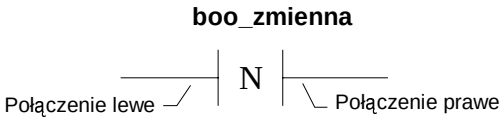
```
(*Przykład wykorzystania styków O ZBOCZU NARASTAJĄCYM *)
```




```
(* Równoważność ST: *)
output1 := input1 AND (input2 AND NOT (input2prev));
(* input2prev to wartość input2 w poprzednim cyklu *)
```

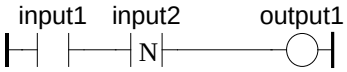
Styki z wykrywaniem opadającego zbocza sygnału

Poniższy styk (ujemny) umożliwia **operację binarną** pomiędzy stanem **linii łączącej** opadającym zboczem sygnału **zmiennej** binarnej.



Stan linii łączącej po prawej stronie styku ustawiany jest na **PRAWDA**, kiedy stan linii łączące po lewej stronie styku to **PRAWDA**, a stan powiązanej z nim zmiennej **opada** od PRAWDA do FAŁSZ. We wszystkich pozostałych przypadkach jest on zmieniany na FAŁSZ.

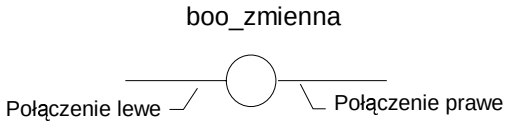
(*Przykład wykorzystania styków O ZBOCZU OPADAJĄCYM *)




```
(* Równoważność ST: *)
output1 := input1 AND (NOT (input2) AND input2prev);
(* input2prev to wartość input2 w poprzednim cyklu *)
```

Cewka bezpośrednia

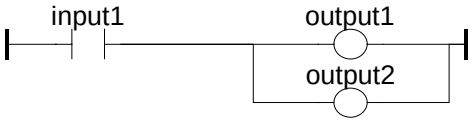
Cewki bezpośrednie umożliwiają uzyskanie **wyjścia binarnego** ze stanu **linii łączącej**.



Powiązanej zmiennej przyporządkowywana jest binarna wartość **lewego połączenia**. Stan lewego połączenia jest rozprowadzany do połączenia prawego. Połączenie prawe może łączyć się z prawą pionową szyna zasilającą.

Powiązana zmienna binarna musi być zmienną **WYJŚCIOWĄ** lub **WEWNĘTRZNĄ**.
Powiązana nazwa może być nazwą programu (tylko dla **podprogramów**). Odpowiada ona przyporządkowaniu wartości zwracanej podprogramu.

(* Przykład wykorzystania cewek BEZPOŚREDNICH *)

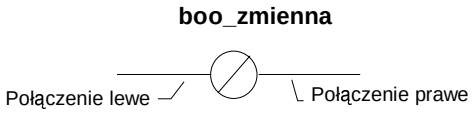


(* Równoważność ST: *)

output1 := input1;
output2 := input1;

Cewka zanegowana

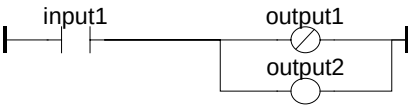
Cewki zanegowane umożliwiają uzyskanie **wyjścia binarnego** będącego logiczną **negacją** stanu **linii łączącej**.



Do powiązanej zmiennej jest przyporządkowywana binarna **negacja lewego połączenia**. Stan lewego połączenia jest rozprowadzany do połączenia prawego. Połączenie prawe może łączyć się z prawą pionową szyna zasilającą.

Powiązana zmienna binarna musi być zmienną **WYJŚCIOWĄ** lub **WEWNĘTRZNĄ**.
Powiązana nazwa może być nazwą programu (tylko dla **podprogramów**). Odpowiada ona przyporządkowaniu wartości zwracanej podprogramu.

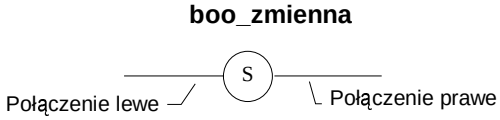
(* Przykład wykorzystania cewek ZANEGOWANYCH *)



(* Równoważność ST: *)
output1 := NOT (input1);
output2 := input1;

Cewki SET (ustaw)

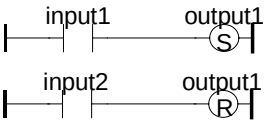
Cewki "Set" umożliwiają uzyskanie **wyjścia binarnego** ze stanu **linii łączącej**.



Powiązana zmienna zostaje **USTAWIONA PRAWDA**, kiedy **lewe połączenie** przyjmuje wartość PRAWDA. Zmienna wyjściowa utrzymuje tę wartość do momentu wydania polecenia odwrotnego przez cewkę „RESET". Stan lewego połączenia jest rozprowadzany do połączenia prawego. Połączenie prawe może łączyć się z prawą pionową szyna zasilającą.

Powiązana zmienna binarna musi być zmienną **WYJŚCIOWĄ** lub **WEWNĘTRZNĄ**.

(* Przykład wykorzystania cewek "SET" i "RESET" *)

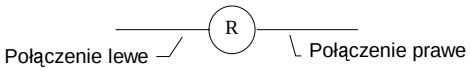



```
(* Równoważność ST: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

Cewka RESET

Cewki "Reset" umożliwiają uzyskanie **wyjścia binarnego** ze stanu **linii łączącej**.

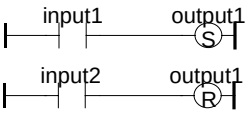
boo_zmienna



Powiązana zmienna zostaje **USTAWIONA NA FAŁSZ**, kiedy **lewe połączenie** staje się wartością **PRAWDA**. Zmienna wyjściowa utrzymuje tę wartość do momentu wydania odwrotnego polecenia przez cewkę „SET". Stan lewego połączenia jest rozprowadzany do połączenia prawego. Połączenie prawe może łączyć się z prawą pionową szyna zasilającą.

Powiązana zmienna binarna musi być zmienną **WYJŚCIOWĄ** lub **WEWNĘTRZNĄ**.

```
 (* Przykład wykorzystania cewek "SET" i "RESET" *)
```

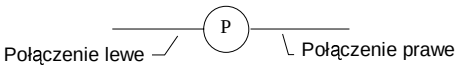



```
(* Równoważność ST: *)
IF input1 THEN
  output1 := TRUE;
END_IF;
IF input2 THEN
  output1 := FALSE;
END_IF;
```

Cewka z wykrywaniem narastającego zbocza sygnału

Cewki "dodatnie" umożliwiają uzyskanie **wyjścia binarnego** ze stanu **linii łączącej**. Ten typ cewek jest dostępny jedynie gdy używa się edytora LD.

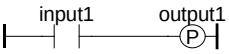
boo_zmienna



Powiązana zmienna zostaje ustawiona na **PRAWDA**, kiedy **lewe połączenie** rośnie od wartości FAŁSZ do PRAWDA. We wszystkich pozostałych przypadkach zmienna przestawia się na wartość FAŁSZ. Stan lewego połączenia jest rozprowadzany do połączenia prawego. Połączenie prawe może łączyć się z prawą pionową szyna zasilającą.

Powiązana zmienna binarna musi być zmienną **WYJŚCIOWĄ** lub **WEWNĘTRZNĄ**.

(* Przykład wykorzystania cewki "dodatniej" *)

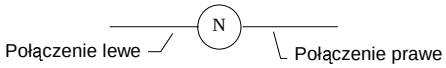



```
(* Równoważność ST: *)
IF (input1 and NOT(input1prev)) THEN
  output1 := TRUE;
ELSE
  output1 := FALSE;
END_IF;
(* input2prev to wartość input2 w poprzednim cyklu *)
```

Cewka z wykrywaniem opadającego zbocza sygnału

Cewki "ujemne" umożliwiają uzyskanie **wyjścia binarnego** ze stanu **linii łączącej**. Ten typ cewek jest dostępny jedynie wtedy, gdy używa się edytora LD.

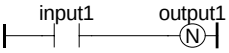
boo_zmienna



Powiązana zmienna zostaje ustawiona na **PRAWDA,** kiedy **lewe połączenie** opada od wartości PRAWDA do FAŁSZ. We wszystkich pozostałych przypadkach zmienna przechodzi na wartość FAŁSZ. Stan lewego połączenia jest rozprowadzany do połączenia prawego. Połączenie prawe może łączyć się z prawą pionową szyna zasilającą.

Powiązana zmienna binarna musi być zmienną **WYJŚCIOWĄ** lub **WEWNĘTRZNĄ**.

(* Przykład wykorzystania cewki "ujemnej" *)




```
(* Równoważność ST: *)
IF (NOT(input1) and input1prev) THEN
  output1 := TRUE;
ELSE
```

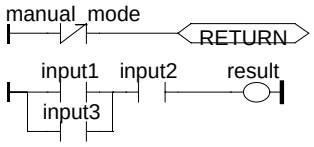
```
  output1 := FALSE;
END_IF;
(* input2prev to wartość input2 w poprzednim cyklu *)
```

B.6.4 Instrukcja RETURN

Etykieta **RETURN** może zostać użyta jako wyjście reprezentujące warunkowe zakończenie programu. Po prawej stronie symbolu RETURN nie można umieścić żadnego połączenia.

Jeżeli **lewa linia łącząca** ma stan **PRAWDA**, to program kończy się bez wykonania równań wpisanych w kolejnych liniach diagramu.
Uwaga: Kiedy program LD jest podprogramem, jego nazwa musi zostać powiązana z cewką wyjściową w celu ustawienia wartości zwracanej (zwracanej do programu wywołującego).

(*Przykład wykorzystania symbolu RETURN *)




```
(* Równoważność ST: *)
If Not (manual_mode) Then RETURN; End_if;
result := (input1 OR input3) AND input2;
```

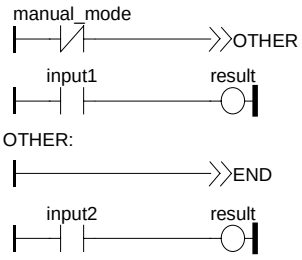
B.6.5 Skoki i etykiety

Symbole etykiet i SKOKÓW warunkowych i bezwarunkowych można wykorzystać do kontroli wykonywania diagramu. Na prawo od symbolu etykiety i skoku nie może zostać umieszczone żadne połączenie. Stosowana jest następująca notacja:

>>LAB ..... skok do etykiety o nazwie "LAB"
LAB: ..... definicja etykiety o nazwie "LAB"

Jeśli **połączenie po lewej** stronie symbolu skoku ma stan **PRAWDA**, to wykonywanie programu jest przenoszone poza symbol etykiety.

(*Przykład wykorzystania symboli SKOKU i ETYKIETY *)



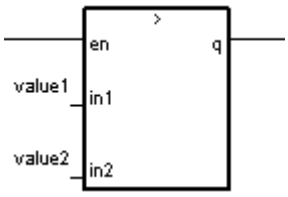

```
(* Równoważność IL: *)
            ldn     manual_mode
            jmpc    other
            ld      input1
            st      result
            jmp     END
OTHER:      ld      input2
            st      result
END:        (* koniec programu *)
```

B.6.6 Bloki w LD

Używając edytor LD, można połączyć ramki funkcji z liniami stanów binarnych. Funkcja może w rzeczywistości być operatorem, blokiem funkcyjnym lub funkcją. Jako, że nie wszystkie bloki zawsze posiadają wejście binarne i/lub wyjście binarne, więc wstawienie bloków do diagramu LD prowadzi do dodania nowych parametrów EN, ENO do interfejsu bloku. Parametry EN, ENO nie są dodawane jeśli wykorzystuje się edytor FBD/LD ponieważ można przyłączyć zmienną o wymaganym typie.

- ***Wejście "EN"***

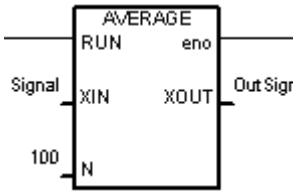
W przypadku niektórych operatorów, funkcji lub bloków funkcyjnych, pierwsze wejście nie zawiera danych binarnych. Ponieważ pierwsze wejście musi być zawsze połączone ze szczeblem, kolejne wejście jest wstawiane automatycznie w pierwszej pozycji zwanej "**EN**". Blok jest wykonywany tylko wtedy, gdy wejście **EN** ma wartość PRAWDA. Poniżej znajduje się przykład operatora porównawczego oraz odpowiadający mu kod wyrażony w ST:




```
IF rung_state THEN
   q := (value1 > value 2);
ELSE
   q := FALSE;
END_IF;
(*kontynuacja  szczebla  ze  stanem
q *)
```

- ***Wyjście "ENO"***

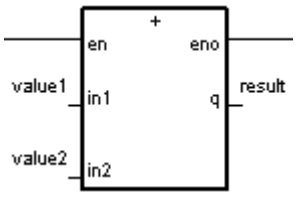
W przypadku niektórych operatorów, funkcji lub bloków funkcyjnych, pierwsze wyjście nie zawiera danych binarnych. Ponieważ pierwsze wyjście musi być zawsze połączone ze szczeblem, kolejne wyjście jest wstawiane automatycznie w pierwszej pozycji zwanej "**ENO**". Wyjście **ENO** zawsze przyjmuje taką samą wartość jak pierwsze wejście bloku. Poniżej znajduje się przykład bloku funkcyjnego AVERAGE oraz odpowiadający jej kod wyrażony w ST:




```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(*  kontynuacja  szczebla  ze  stanem  eno
*)
```

Korzystanie z "EN" i "ENO"

W niektórych przypadkach wymagane są zarówno **EN** jak i **ENO**. Poniżej znajduje się przykład operatora arytmetycznego oraz odpowiadający mu kod wyrażony w ST:




```
IF rung_state THEN
   result := (value1 + value2);
END_IF;
eno := rung_state;
(* kontynuacja szczebla ze stanem eno *)
```

B.7 Język ST

ST (**Tekst Strukturalny**) jest językiem strukturalnym wysokiego poziomu przeznaczonym do programowania procesów automatyki. Język ten jest używany głównie do implementacji złożonych procedur, których nie można wyrazić w prosty sposób przy pomocy języków graficznych. ST jest językiem domyślnym opisu operacji w ramach etapów i warunków dołączonych do przejść języka **SFC**.

B.7.1 Główna składnia ST

Program ST jest listą **instrukcji** ST. Każda instrukcja zakończona jest separatorem średnika (**";"**). Nazwy zastosowane w kodzie źródłowym (identyfikatory zmiennych, stałe, słowa kluczowe języka...) są oddzielone **separatorami nieaktywnymi** (znakami spacji, znakami zatrzymania końca wiersza lub tabulatora) lub **separatorami aktywnymi**, posiadającymi z góry określone znaczenie (na przykład separator **">"** oznacza porównanie "większy niż"). Komentarze mogą być wstawiane swobodnie w tekście. Komentarz musi rozpoczynać się znakami **"(*"** i kończyć znakami **"*)"**. Każda instrukcja kończy się separatorem średnika (**";"**). Oto podstawowe typy instrukcji ST:

- instrukcja **przyporządkowania** (zmienna := wyrażenie;)
- wywołanie **podprogramu** lub **funkcji**
- wywołanie **bloku funkcyjnego**
- instrukcje **wyboru** (IF, THEN, ELSE, CASE...)
- instrukcje **iteracyjne** (FOR, WHILE, REPEAT...)
- instrukcje **sterujące** (RETURN, EXIT...)
- instrukcje specjalne odnoszące się do połączeń z innymi językami, takimi jak **SFC**

Separatory nieaktywne mogą być wstawiane swobodnie pomiędzy separatory aktywne, wyrażenia stałe i identyfikatory. Separatory nieaktywne ST, to: znak **Spacji** (pusty), **Tabulatory** i znaki **Końca wiersza**. W odróżnieniu od języków formatowanych liniowo, takich jak IL, końce wierszy mogą być wstawiane w dowolnym miejscu programu. Przedstawione poniżej zasady powinny być przestrzegane przy używaniu separatorów nieaktywnych do podniesienia czytelności programu ST:

- Nie wpisywać więcej niż jednej instrukcji w jednej linii
- Używać tabulatorów do tworzenia wcięć w złożonych instrukcjach
- Wstawiać komentarze, aby podnieść czytelność linii lub akapitów

B.7.2 Wyrażenie i nawiasy

Wyrażenia ST łączą **operatory** i zmienne lub **argumenty** stałe ST. **Typ** argumentu musi być taki sam dla każdego pojedynczego wyrażenia (łączącego argumenty z jednym operatorem ST). Takie wyrażenie pojedyncze posiada taki sam typ jak jego argumenty i może być używane w bardziej złożonych wyrażeniach. Na przykład:

(boo_var1 AND boo_var2)	binarne
not (boo_var1)	binarne
(sin (3.14) + 0.72)	analogowe typu real
(t#1s23 + 1.78)	nieprawidłowe wyrażenie

Nawiasy są używane do oddzielenia części podrzędnych wyrażenia oraz do wyraźnego uporządkowania priorytetu operacji. Kiedy w wyrażeniu złożonym nie występują nawiasy, kolejność operacji jest ustalana na podstawie domyślnego **priorytetu** pomiędzy operatorami ST. Na przykład:

2 + 3 * 6	jest równe 2+18=20	ponieważ operator mnożenia posiada wyższy priorytet
(2+3) * 6	jest równe 5*6=30	priorytet jest nadawany przez nawiasy

Uwaga: W wyrażeniu może być zagnieżdżonych do **8** poziomów nawiasów.

B.7.3 Wywołania funkcji i bloków funkcyjnych

Standardowe wywołania funkcji ST mogą zostać zastosowane do każdego z następujących obiektów:
- Podprogramy
- Funkcje i bloki funkcyjnych biblioteki napisane w językach IEC
- funkcje i bloki funkcyjnych "C"
- Funkcje konwersji typów

Wywołania podprogramów lub funkcji

Nazwa: nazwa wywoływanego podprogramu
lub funkcji biblioteki napisana w języku IEC lub w "C"

Znaczenie: wywołuje podprogram ST, IL, LD lub FBD albo funkcję, czy też funkcję "C"
i uzyskuje wartość zwracaną

Składnia: <**zmienna**> **:=** <**podprog**> **(**<**par1**>**,** ... <**parN**> **);**

Argumenty: Typ wartości zwracanej i parametrów wywołania musi występować po określeniu interfejsu dla podprogramu.

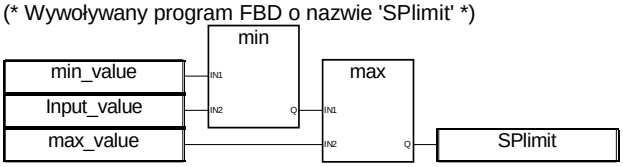
Wartość zwracana: wartość zwracana przez podprogram

Wywołania podprogramu mogą być używane jako wyrażenia. Mogą one być również używane w przejściu SFC.

Przykład 1: Wywołanie podprogramu

(* Program główny ST *)

```
(* otrzymuje wartość analogową i konwertuje ją na ograniczona wartość czasową *)
ana_timeprog := SPlimit ( tprog_cmd );
appl_timer := tmr (ana_timeprog * 100);

(* Wywoływany program FBD o nazwie 'SPlimit' *)
```



Przykład 2: Wywołanie funkcji

```
(* funkcje wykorzystywane w wyrażeniach złożonych: min, max, right, mlen oraz left, to
standardowe funkcje "C" *)
limited_value := min (16, max (0, input_value) );
rol_msg := right (message, mlen (message) - 1) + left (message, 1);
```

Wywołania bloków funkcyjnych

Nazwa: nazwa instancji bloku funkcyjnego

Znaczenie: wywołuje blok funkcyjne z biblioteki ISaGRAF lub z biblioteki użytkownika i uzyskuje dostęp do jego parametrów zwracanych

Składnia:
(* wywołanie bloku funkcyjnego *)
<**nazwabloku**> **(** <**p1**>**,** <**p2**> ... **);**
(uzyskanie jego parametrów zwracanych *)
<**wynik**> **:=** <**nazwabloku**>**.** <**param_zwr1**>**;**
...
<**wynik**> **:=** <**nazwabloku**>**.** <**param_zwrN**>**;**

Argumenty: parametry, to wyrażenia, które pasują do typu parametrów określonych dla tego bloku funkcyjnego

Wartość zwracana: Patrz Składnia aby uzyskać parametry zwracane.

Aby dowiedzieć się więcej na temat znaczenia i typu każdego parametru bloku funkcyjnego - patrz biblioteka ISaGRAF. Instancja bloku funkcyjnego (nazwa kopii) musi zostać zadeklarowana w słowniku

Przykład:

```
(* Program ST wywołujący blok funkcyjny *)

(* deklaracja instancji bloku w słowniku: *)
(* trigb1 : blok R_TRIG - wykrywanie zbocza rosnącego *)

(* aktywacja bloku funkcyjnego z języka ST *)
trigb1 (b1);
```

```
(* dostęp do parametrów zwracanych *)
If (trigb1.Q) Then nb_edge := nb_edge + 1; End_if;
```

B.7.4 Operatory binarne specyficzne dla ST

Następujące operatory binarne są specyficzne dla języka ST:

-REDGE	wykrywanie zbocza rosnącego
-FEDGE	wykrywanie zbocza opadającego

Wykorzystane mogą zostać inne standardowe operatory binarne, takie jak:

-NOT	negacja logiczna
-AND (&)	logiczne I
-OR	logiczne LUB
-XOR	logiczne LUB wyłączające.

Ich opis można znaleźć w sekcji 'Standardowe operatory, bloki funkcyjne i funkcje'.

Operator "REDGE"

Nazwa:	**REDGE**
Znaczenie:	ocenia zbocze rosnące pełnego wyrażenia binarnego
Składnia:	**<zbocze> := REDGE (<wyr_binarne>,<zmienna_memo>);**
Argumenty:	pierwszy argument jest zmienną binarną wyrażeniem złożonym drugi argument jest wewnętrzną zmienną binarną, używaną do przechowywania ostatniego stanu wyrażenia
Wartość zwracana:	PRAWDA, kiedy wyrażenie zmienia się z FAŁSZ na PRAWDA FAŁSZ we wszystkich pozostałych przypadkach

Zbocze rosnące wyrażenia nie może zostać wykryte więcej niż jeden raz w jednym cyklu wykonawczym, wykorzystującym operator REDGE. Operator ten może zostać wykorzystany do opisu warunku dołączonego do przejścia SFC.

Uwaga: "Pamięć" zmiennej binarnej, wykorzystywana do przechowywania ostatniego stanu wyrażenia, nie może być użyta do wyzwalania zboczy innych wyrażeń.

Kiedy wyrażenie jest zmienną binarną o nazwie "**xxx**", dla zmiennej tej należy zadeklarować i używać w wyrażeniach REDGE niepowtarzalnej zmiennej wewnętrznej o nazwie "**EDGE_xxx**". Metoda ta zapewnia, iż zmienna pamięci nie zostanie nadpisana podczas innych ocen REDGE.

Przykład:

```
(* Program ST wykorzystujący operator REDGE *)

(* program ten zlicza zbocza rosnące wejścia binarnego*)
(* Bi120 jest wejściową zmienną binarną *)
(* Edge_Bi120 to pamięć stanu zmiennej Bi120 *)

If REDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
End_if;
```

Uwaga: operator ten nie jest zgodny z normą IEC1131-3. Możliwe, że będą Państwo woleli użyć standardowego bloku R_TRIG. Zachowano go ze względu na kompatybilność.

Operator "FEDGE"

Nazwa:	**FEDGE**
Znaczenie:	ocenia zbocze opadające wyrażenia binarnego
Składnia:	**<zbocze> := FEDGE (<wyr_binarne>,<zmienna_memo>);**
Argumenty:	pierwszy argument jest zmienną binarną lub wyrażeniem złożonym drugi argument jest wewnętrzną zmienną binarną, używaną do przechowywania ostatniego stanu wyrażenia
Wartość zwracana:	PRAWDA, kiedy wyrażenie zmienia się z PRAWDA na FAŁSZ FAŁSZ we wszystkich pozostałych przypadkach

Zbocze opadające wyrażenia nie może zostać wykryte więcej niż jeden raz w jednym cyklu wykonawczym, wykorzystującym operator REDGE. Operator ten może zostać wykorzystany do opisu warunku dołączonego do przejścia SFC.
Uwaga: "Pamięć" zmiennej binarnej, wykorzystywana do przechowywania ostatniego stanu wyrażenia, nie może być użyta do wyzwalania zboczy innych wyrażeń.

Kiedy wyrażenie jest zmienną binarną o nazwie "**xxx**", dla zmiennej tej należy zadeklarować i używać w wyrażeniach FEDGE niepowtarzalnej zmiennej wewnętrznej o nazwie "**EDGE_xxx**". Metoda ta zapewnia, iż zmienna pamięci nie zostanie nadpisana podczas innych ocen FEDGE.

Przykład:

```
(* Program ST wykorzystujący operator FEDGE *)

(* program ten zlicza zbocza opadające wejścia binarnego*)
(* Bi120 jest wejściową zmienną binarną *)
(* Edge_Bi120 to pamięć stanu zmiennej Bi120 *)

If FEDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
End_if;
```

Uwaga: operator ten nie jest zgodny z normą IEC1131-3. Możliwe, że będą Państwo woleli użyć standardowego bloku F_TRIG. Zachowano go ze względu na kompatybilność.

B.7.5 Podstawowe instrukcje ST

Podstawowymi instrukcjami języka ST są:

-Przyporządkowanie
-instrukcja RETURN
-Struktura IF-THEN-ELSIF-ELSE
-Instrukcja CASE
-Instrukcja iteracyjna WHILE

-Instrukcja iteracyjna REPEAT
-Instrukcja iteracyjna FOR
-Instrukcja EXIT

Przyporządkowanie

Nazwa: **:=**
Znaczenie: przyporządkowuje zmienną do wyrażenia
Składnia: **<zmienna> := <dowolne_wyr> ;**
Argumenty: zmienna musi być zmienną wewnętrzną lub wyjściową
zmienna i wyrażenie muszą być tego samego typu

Wyrażenie może być odwołaniem do podprogramu lub funkcji z biblioteki ISaGRAF

Przykład:

(* Program ST z przyporządkowaniami *)

(* zmienna <<= zmienna *)
bo23 := bo10;

(*zmienna <<= wyrażenie *)
bo56 := bx34 OR alrm100 & (level >= over_value);
result := (100 * input_value) / scale;

(* przyporządkowanie wartości zwracanej podprogramu *)
rc := PSelect ( );

(* przyporządkowanie z wywołaniem funkcji *)
limited_value := min (16, max (0, input_value));

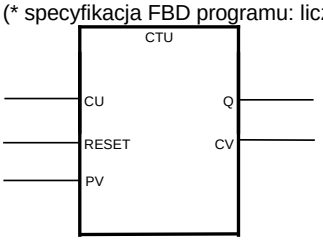
Instrukcja RETURN

Nazwa: **RETURN**
Znaczenie: kończy wykonywanie bieżącego programu
Składnia: **RETURN ;**
Argumenty: (brak)

W bloku operacji SFC, instrukcja RETURN wskazuje na koniec wykonywania tylko tego bloku.

Przykład:

(* specyfikacja FBD programu: licznik programowalny *)



```
(* implementacja ST programu, wykorzystująca instrukcję RETURN *)

If not (CU) then
        Q := false;
        CV := 0;
        RETURN;   (* kończy program *)
end_if;

if R then
        CV := 0;
else
        if (CV < PV) then
                CV := CV + 1;
        end_if;
end_if;
Q := (CV >= PV);
```

Instrukcja IF-THEN-ELSIF-ELSE

Nazwa: **IF ... THEN ... ELSIF ... THEN ... ELSE ... END_IF**

Znaczenie: wykonuje dwie listy instrukcji ST
wyboru dokonuje się zgodnie z wartością
wyrażenia binarnego

Składnia:

```
IF <wyrażenie_binarne> THEN
  <instrukcja> ;
  <instrukcja> ;
  ...
ELSIF <wyrażenie_binarne> THEN
  <instrukcja> ;
  <instrukcja> ;
  ...
ELSE
  <instrukcja> ;
  <instrukcja> ;
  ...
END_IF;
```

Instrukcje ELSE oraz ELSIF są opcjonalne. Jeśli instrukcja ELSE nie zostanie zapisana, kiedy warunkiem jest FAŁSZ, nie będzie wykonywana żadna instrukcja.

Przykład:

```
(* program ST wykorzystujący instrukcję IF *)

IF manual AND not (alarm) THEN
        level := manual_level;
        bx126 := bi12 OR bi45;
ELSIF over_mode THEN
        level := max_level;
ELSE
        level := (lv16 * 100) / scale;
END_IF;
```

```
(* struktura IF bez ELSE *)
If overflow THEN
        alarm_level := true;
END_IF;
```

Instrukcja CASE

Nazwa: **CASE ... OF ... ELSE ... END_CASE**
Znaczenie: wykonuje jedną z kilku list instrukcji ST
wyboru dokonuje się zgodnie z wyrażeniem integer
Składnia:

```
CASE <wyrażenie_integer> OF
  <wartość> : <instrukcje> ;
  <wartość> , <wartość> : <instrukcje> ;
  ...
ELSE
  <instrukcje> ;
END_CASE;
```

Wartości przypadków muszą być stałymi wyrażeniami integer. Kilka wartości, oddzielonych przecinkami, może, prowadzić do jednej listy instrukcji. Instrukcja ELSE jest opcjonalna.

Przykład:

```
(*program ST wykorzystujący instrukcję CASE *)

CASE error_code OF
        255:    err_msg := 'Division by zero';
                fatal_error := TRUE;
        1:      err_msg := 'Overflow';
        2, 3:   err_msg := 'Bad sign';
ELSE
        err_msg := 'Unknown error';
END_CASE;
```

Instrukcja WHILE

Nazwa: **WHILE ... DO ... END_WHILE**
Znaczenie: struktura iteracyjna grupy instrukcji ST
warunek "kontynuuj" jest oceniany PRZED jakąkolwiek iteracją
Składnia:

```
WHILE <wyrażenie_binarne> DO
  <instrukcja> ;
  <instrukcja> ;
  ...
END_WHILE ;
```

Uwaga: Ze względu na fakt, iż ISaGRAF jest systemem **synchronicznym**, zmienne wejściowe nie są odświeżane podczas iteracji WHILE. Zmiana stanu zmiennej wejściowej nie może zostać wykorzystana do opisu warunku instrukcji WHILE.

Przykład:

```
(* program ST wykorzystujący instrukcję WHILE *)

(* program ten używa specyficznych funkcji "C" do odczytu znaków *)
(* na porcie szeregowym *)

string := ''; (* pusty ciąg *)
nbchar := 0;

WHILE ((nbchar < 16) & ComIsReady ( )) DO
        string := string + ComGetChar ( );
        nbchar := nbchar + 1;
END_WHILE;
```

Instrukcja REPEAT

Nazwa: **REPEAT ... UNTIL ... END_REPEAT**

Znaczenie: struktura iteracyjna dla grupy instrukcji ST
warunek "kontynuuj" jest oceniany PO jakiejkolwiek iteracji

Składnia:

```
REPEAT
  <instrukcja> ;
  <instrukcja> ;
  ...
UNTIL <warunek_binarny>
END_REPEAT ;
```

Uwaga: Ze względu na fakt, iż ISaGRAF jest systemem **synchronicznym**, zmienne wejściowe nie są odświeżane podczas iteracji REPEAT. Zmiana stanu zmiennej wejściowej nie może zostać wykorzystana do opisu warunku końcowego instrukcji WHILE.

Przykład:

```
(* program ST wykorzystujący instrukcję REPEAT *)

(* program ten używa specyficznych funkcji "C" do odczytu znaków *)
(* na porcie szeregowym *)

string := ''; (* ciąg pusty *)
nbchar := 0;
IF ComIsReady ( ) THEN
        REPEAT
                string := string + ComGetChar ( );
                nbchar := nbchar + 1;
        UNTIL ( (nbchar >= 16) OR NOT (ComIsReady ( )) )
        END_REPEAT;
END_IF;
```

Instrukcja FOR

Nazwa: **FOR ... TO ... BY ... DO ... END_FOR**
Znaczenie: wykonuje ograniczoną ilość iteracji,

wykorzystując analogową zmienną indeksową typu integer

Składnia: **FOR <index> := <mini> TO <maxi> BY <step> DO**
<instrukcja> ;
<instrukcja> ;
END_FOR;

Argumenty:
index: wewnętrzna zmienna analogowa, powiększająca się przy każdej pętli
mini: wartość początkowa indeksu (przed pierwszą pętlą)
maxi: maksymalna dopuszczalna wartość indeksu
step: przyrost indeksu przy każdej pętli

Instrukcja [BY step] jest opcjonalna. Jeśli jej nie określono, krok przyrostu to 1

Uwaga: Ze względu na fakt, iż ISaGRAF jest systemem **synchronicznym**, zmienne wejściowe nie są odświeżane podczas iteracji FOR.

Oto odpowiednik instrukcji FOR wyrażony w "while":

```
index := mini;
while (step <= maxi) do
  <statement> ;
  <statement> ;
  index := index + step;
end_while;
```

Przykład:

```
(* program ST wykorzystujący instrukcję FOR *)
(* program ten wydziela znaki cyfr z ciągu *)

length := mlen (message);
target := ''; (* pusty ciąg *)
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code >= 48) & (code <= 57) THEN
                target := target + char (code);
        END_IF;
END_FOR;
```

Instrukcja EXIT

Nazwa: **EXIT**
Znaczenie: wyjście z instrukcji iteracji FOR, WHILE lub REPEAT
Składnia: **EXIT;**
Argumenty: (brak)

EXIT jest używana powszechnie w ramach instrukcji IF, wewnątrz bloku FOR, WHILE lub REPEAT.

Przykład:

```
(* program ST wykorzystujący instrukcję EXIT *)
```

```
(* program ten poszukuje ciągu znaków *)

length := mlen (message);
found := NO;
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code  = searched_char) THEN
                found := YES;
                EXIT;
        END_IF;
END_FOR;
```

B.7.6 Rozszerzenia ST

Następujące funkcje są rozszerzeniami języka ST:

- TSTART - TSTOP: sterowanie licznikiem (timer)

Dostępne są następujące instrukcje i funkcje, pozwalające sterować wykonywaniem programów potomnych SFC. Mogą być one stosowane wewnątrz bloków ACTION(): ... END_ACTION; w etapach SFC.

-GSTART	uruchamia program SFC
-GKILL	zamyka program SFC
-GFREEZE	wstrzymuje program SFC
GRST	uruchamia ponownie wstrzymany program SFC
GSTATUS	pobiera bieżący status programu SFC

Uwaga: Funkcje te nie są zgodne z normą IEC 1131-3.
Dla GSTART oraz GKILL można łatwo znaleźć odpowiedniki, stosując następującą składnię w etapie SFC:

```
nazwa_potomka(S); (* odpowiada GSTART(nazwa_potomka); *)
nazwa_potomka(R); (* odpowiada GKILL(nazwa_potomka); *)
```

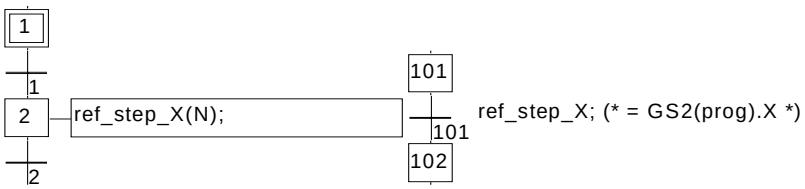
Poniższe pola mogą zostać wykorzystane do uzyskania dostępu do statusu etapu SFC:

GSnnn.x	wartość binarna, która reprezentuje operacje etapu
GSnnn.t	czas, który upłynął od ostatniej aktywizacji etapu ("**nnn**" to numer odniesienia etapu SFC)

Możliwe jest również przetestowanie operacji etapu, zadeklarowanej w innym programie SFC, wykorzystując następującą składnię:

GSnnn(nazwaprog).x

Uwaga: odniesienia do etapu innego programu przy użyciu tej składni nie są zgodne z normą IEC 1131-3. Prosty sposób wykonania tego samego w zgodzie z zasadami IEC, to zadeklarowanie w słowniku globalnej zmiennej binarnej, która reprezentować będzie operację etapu, która ma być przetestowana (na przykład ref_step_X). Następnie w etapie wstawiamy zmienną z kwantyfikatorem N (ref_step_X(N);), po czym stosujemy tą zmienną w programie, w którym chcemy przetestować operację etapu.
Program Prog inny program, który wymaga w etapie operacji programu Prog



Instrukcja TSTART

Nazwa: **TSTART**
Znaczenie: rozpoczyna zliczanie zmiennej typu timer
wartość zmiennej timer nie jest modyfikowana przez komendę TSTART, tj. zliczanie rozpoczyna się od bieżącej wartości zmiennej timer.
Składnia: **TSTART (<zmienna_timer >);**
Argumenty: dowolna aktywna zmienna typu timer
Wartość zwracana: (brak)

Przykład:

```
(* Program SFC wykorzystujący instrukcje TSTART i TSTOP *)
```

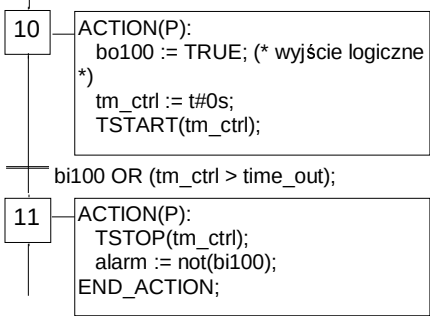
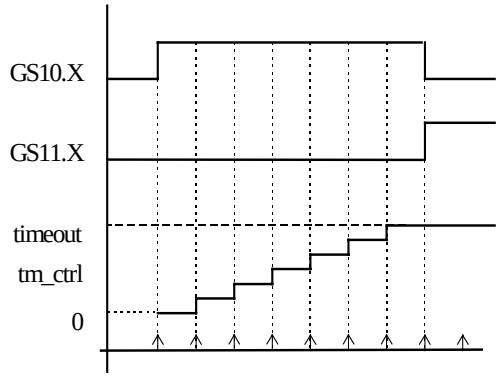


Diagram czasowy if bi100 jest zawsze FAŁSZYWY.



Zmienna timer utrzymuje taką samą wartość w czasie jednego cyklu.

Instrukcja TSTOP

Nazwa:	**TSTOP**
Znaczenie:	zatrzymuje aktualizację zmiennej typu timer wartość zmiennej timer nie jest modyfikowana przez komendę TSTOP
Składnia:	**TSTOP (<zmienna_timer>);**
Argumenty:	dowolna aktywna zmienna typu timer
Wartość zwracana:	(brak)

Przykład: Patrz TSTART (funkcja opisana jest powyżej)

Instrukcja GSTART

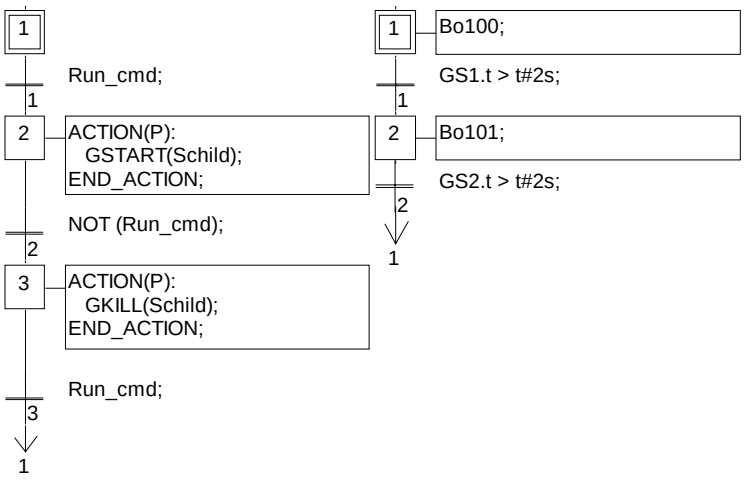
Nazwa:	**GSTART**
Znaczenie:	uruchamia program potomny SFC, umieszczając znacznik w każdym z jego etapów początkowych
Składnia:	**GSTART (<program_potomny>);**
Argumenty:	określony program SFC musi być programem potomnym programu, w którym znajduje się instrukcja
Wartość zwracana:	(brak)

Programy potomne tego programu potomnego nie są uruchamiane automatycznie przez instrukcję GSTART.
Uwaga: Jako, że GSTART nie jest zgodna z normą IEC 1131-3, do uruchamiania programu potomnego SFC należy preferować stosowanie kwantyfikatora S, zachowując następującą składnię:

```
Nazwa_pot(S);
```

Przykład: Użycie GSTART i GKILL

```
(* Sekwencja 'Sfather' *)        (*Sekwencja 'Schild' *)
```



Instrukcja GKILL

Nazwa: **GKILL**
Znaczenie: zamyka program potomny SFC usuwając znaczniki znajdujące się aktualnie w jego etapach
Składnia: **GKILL (<program_pot>);**
Argumenty: określony program SFC musi być programem potomnym programu, w którym znajduje się instrukcja
Wartość zwracana: (brak)

Programy potomne tego programu potomnego są zamykane automatycznie wraz z określonym programem.
Uwaga: Jako, że GKILL nie jest zgodna z normą IEC 1131-3, do uruchamiania programu potomnego SFC należy preferować stosowanie kwantyfikatora R, zachowując następującą składnię:

Nazwa_pot(R);

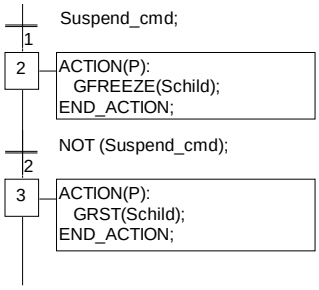
Przykład: Patrz GSTART (funkcje opisano powyżej)

Instrukcja GFREEZE

Nazwa: **GFREEZE**
Znaczenie: Zawiesza wykonywanie programu potomnego SFC
Wstrzymany program może zostać uruchomiony ponownie instrukcją GRST.
Składnia: **GFREEZE (<program_pot>);**
Argumenty: określony program SFC musi być programem potomnym programu, w którym znajduje się instrukcja
Wartość zwracana: (brak)

Programy potomne tego programu potomnego są wstrzymywane automatycznie wraz z określonym programem.
Uwaga: GKILL nie jest zgodna z normą IEC 1131-3.

Przykład:



Instrukcja GRST

Nazwa: **GRST**
Znaczenie: uruchamia ponownie program potomny SFC wstrzymany przy pomocy instrukcji GFREEZE
Składnia: **GRST (<program_pot>);**
Argumenty: określony program SFC musi być programem potomnym programu, w którym znajduje się instrukcja
Wartość zwracana: (brak)

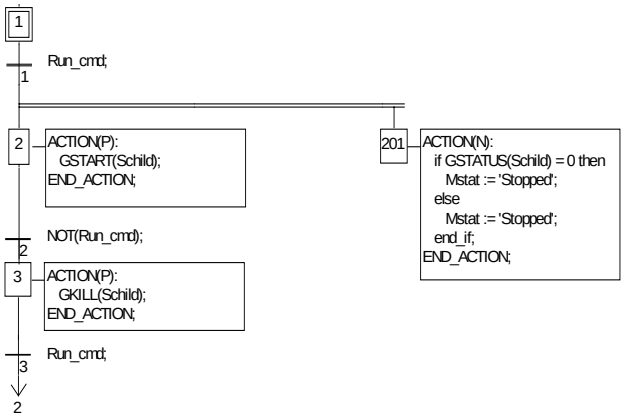
Programy potomne tego programu potomnego są automatycznie uruchamiane ponownie wraz z programem GRST.
Uwaga: GRST nie jest zgodna z normą IEC 1131-3.

Przykład: Patrz GFREEZE (funkcja opisana powyżej)

Instrukcja GSTATUS

Nazwa: **GSTATUS**
Znaczenie: zwraca bieżący status programu SFC
Składnia: **<wart_ana> := GSTATUS (<program_pot>);**
Argumenty: określony program SFC musi być programem potomnym programu, w którym znajduje się instrukcja
Wartość zwracana: 0 = program jest nieaktywny (zamknięty)
1 = program jest aktywny (uruchomiony)
2 = program jest wstrzymany
Uwaga: GFREEZE nie jest zgodna z normą IEC 1131-3.

Przykład:


1
Run_cmd;
1
2
ACTION(P):
GSTART(Schild);
END_ACTION;
201
ACTION(N):
if GSTATUS(Schild) = 0 then
Mstat := 'Stopped';
else
Mstat := 'Stopped';
end_if;
END_ACTION;
NOT(Run_cmd);
2
3
ACTION(P):
GKILL(Schild);
END_ACTION;
Run_cmd;
3
2

B.8 Język IL

Instruction List (Lista Instrukcji) , czyli **IL**, to język niższego poziomu. Jest on bardzo skuteczny w przypadku małych aplikacji lub optymalizacji części aplikacji. Instrukcje zawsze odnoszą się do **wyniku bieżącego** (lub **rejestru IL**). Operator wskazuje, jaka operacja musi zostać wykonana z użyciem wartości bieżącej i argumentu operacji. Wynik tej operacji zostaje zapamiętany jako wynik bieżący.

B.8.1 Główna składnia IL

Program IL jest listą **instrukcji**. Każda instrukcja musi rozpoczynać się w nowej linii i musi zawierać **operator** uzupełniony opcjonalnymi **modyfikatorami** oraz, w razie potrzeby, w operacjach specjalnych, jednym lub kilkoma **argumentami** oddzielonymi przecinkami (**','**). Instrukcję może poprzedzać **etykieta** zakończona dwukropkiem (**':'**). Jeśli do instrukcji dołączony jest **komentarz**, musi on stanowić ostatni element linii. Komentarze zawsze rozpoczynają się znakami **'(*'**, a kończą **'*)'**. Pomiędzy instrukcjami można wstawiać puste linie. W pustych liniach mogą znajdować się komentarze.

poniżej podano przykłady linii instrukcji:

```
Etykieta        Operator    Argument    Komentarz

Start:          LD          IX1             (* naciśnij przycisk *)
                ANDN        MX5             (* komenda dozwolona *)
        ST          QX2             (* uruchom silnik *)
```

Etykiety

Instrukcję może poprzedzać **etykieta** zakończona średnikiem (**':'**). Etykietę można wstawić w pustej linii. Etykiety są wykorzystywane jako argumenty w niektórych operacjach, takich jak skoki. Nadawanie nazw etykietom musi być zgodne z następującymi konwencjami:

- Nazwa nie może przekraczać **16** znaków
- pierwszy znak musi być **literą**
- kolejne znaki muszą być **literami**, **cyframi** lub znakiem **'_'**

Jednej nazwy nie można wykorzystać do więcej niż jednej etykiety w tym samym programie IL. Etykieta może posiadać taką samą nazwę jak zmienna.

Modyfikatory operatorów

Poniżej przedstawiono dostępne modyfikatory operatorów. Znak modyfikatora musi uzupełniać nazwę operatora nie pozostawiając między nimi znaków pustych:

N negacja binarna argumentu
(operacja ze zwłoką
C operacja warunkowa

Modyfikator **'N'** oznacza negację binarną argumentu. Na przykład, instrukcję **ORN IX12** interpretuje się jako: **wynik := wynik OR NOT (IX12)**.

Modyfikator nawiasowy **'('** wskazuje, iż ocena instrukcji musi zostać opóźniona do momentu pojawienia się operatora nawiasu zamykającego **')'**.

Modyfikator **'C'** wskazuje, iż dołączona instrukcja musi zostać wykonana jedynie wtedy, gdy bieżący wynik posiada wartość binarną PRAWDA (dla wartości nie binarnych - różną od 0). Modyfikator **'C'** może być łączony z modyfikatorem **'N'** wskazując, iż instrukcja musi zostać wykonana jedynie wtedy, gdy bieżący wynik posiada wartość binarną FAŁSZ (0 dla wartości nie binarnych).

Operacje ze zwłoką

Ze względu na fakt, iż istnieje tylko jeden rejestr IL (wynik bieżący), niektóre operacje mogą wymagać zwłoki, aby można było zmienić kolejność wykonywania lub instrukcje. Operacje ze zwłoką są oznaczane nawiasami:

'('	to modyfikator	oznacza, iż operacja ma zostać opóźniona
')'	to operator	wykonuje opóźnioną operację

Modyfikator nawiasu otwierającego **'('** wskazuje, iż ocena instrukcji musi zostać opóźniona do momentu napotkania operatora nawiasu zamykającego **')'**. Na przykład, następująca sekwencja:

```
AND(    IX12
OR      IX35
)
```

jest interpretowana jako:

wynik := wynik AND (IX12 OR IX35)

B.8.2 Operatory IL

Poniższa tabela przedstawia zestawienie operatorów języka IL:

Operator	*Modyfikatory*	*Argumenty*	*Opis*
LD	N	Zmienna, stała	Ładuje argument
ST	N	Zmienna	Zachowuje wynik bieżący
S		zmienna BOO (bin.)	Ustawia wartość na PRAWDA
R		zmienna BOO (bin.)	Przestawia wartość na FAŁSZ
CAL	C N	nazwa instancji FB	Wywołuje blok funkcyjny

JMP	C N	Etykieta	Wykonuje skok do etykiety
RET	C N	Powrót z podprogramu	
)		Wykonanie operacji opóźnionej	
Function		Wywołanie funkcji lub podprogramu	
AND	N (	BOO (log.)	binarne I
&	N (	BOO (log.)	binarne I
OR	N (	BOO (log.)	binarne LUB
XOR	N (	BOO (log.)	wyłączające LUB
ADD	(	Zmienna, stała	Dodawanie
SUB	(	Zmienna, stała	Odejmowanie
MUL	(	Zmienna, stała	Mnożenie
DIV	(	Zmienna, stała	Dzielenie
GT	(	Zmienna, stała	Test: >
GE	(	Zmienna, stała	Test: >=
EQ	(	Zmienna, stała	Test: =
LE	(	Zmienna, stała	Test: <=
LT	(	Zmienna, stała	Test: <
NE	(	Zmienna, stała	Test: <>

W następnej sekcji opisane są jedynie operatory charakterystyczne dla języka IL. Inne operatory standardowe znajdują się w sekcji "Operatory standardowe, funkcje i bloki funkcyjne".

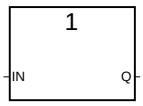
B.9 Standardowe operatory, bloki funkcyjne i funkcje

B.9.1 Standardowe operatory

Poniżej znajdują się standardowe operatory języków IEC.

Obsługa danych:	Przypisanie, Negacja numeryczna
Operacje binarne:	logiczne I logiczne LUB logiczne ex-LUB
Operacje arytmetyczne:	Dodawanie Odejmowanie Mnożenie Dzielenie
Operacje logiczne:	maskowanie bitowe „i” maskowanie bitowe „lub” maskowanie bitowe „ex-lub” maskowanie bitowe Negacja
Testy porównawcze:	Mniejszy niż Mniejszy lub równy Większy niż Większy lub równy Równy Różny
Konwersja danych:	Konwertuj na dane binarne Konwertuj na dane analogowe integer Konwertuj na dane analogowe real Konwertuj na dane typu Timer Konwertuj na dane typu komunikat
Inne:	Łączenie komunikatów Dostęp do systemu Obsługa kanału We/Wy

1 gain



Argumenty:

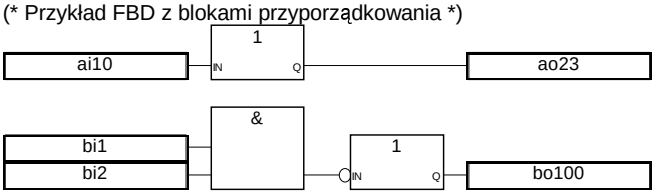
IN	dowolnego typu
Q	dowolnego typu

Opis:

Przepisanie jednej zmiennej do innej

Blok ten jest bardzo przydatny do bezpośredniego łączenia wejścia diagramu z wyjściem diagramu. Można go również użyć (z linią negacji binarnej) do odwrócenia stanu linii dołączonej do wyjścia diagramu.

(* Przykład FBD z blokami przyporządkowania *)




```
(* Reprezentacja ST: *)
ao23 := ai10;
bo100 := NOT (bi1 AND bi2);

(* Reprezentacja IL: *)
LD      ai10
ST      ao23
LD      bi1
AND     bi2
STN     bo100
```

NEG

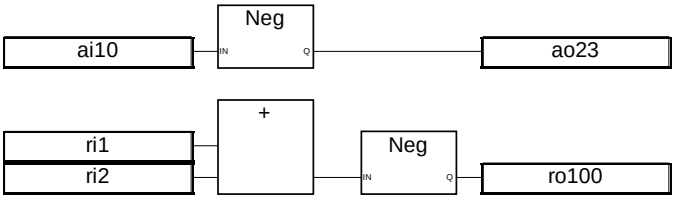
Argumenty:

IN	INT-REAL	wejście i wyjście muszą mieć taki sam format
Q	INT-REAL	

Opis:

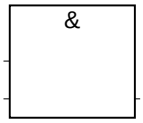
Przyporządkowanie negację zmiennej.

(* Przykład FBD z blokami negacji *)




```
(* Reprezentacja ST: *)
ao23 := - (ai10);
ro100 := - (ri1 + ri2);

(* Reprezentacja IL: *)
LD          ai10
MUL         -1
ST          ao23
LD          ri1
ADD         ri2
MUL         -1.0
ST          ro100
```

& AND



Uwaga: Dla tego operatora ilość wejść może zostać zwiększona do więcej niż dwa.

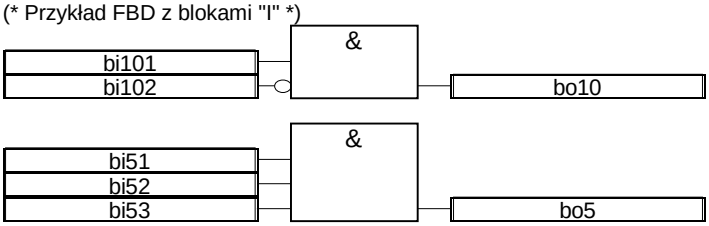
Argumenty:

(wejścia)	BINARNE
wyjście	BINARNE „I” składników wejściowych

Opis:

Logiczne I dwóch lub więcej składników.

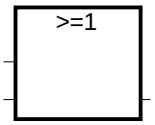
```
(* Przykład FBD z blokami "I" *)
```




```
(* Reprezentacja ST: *)
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

```
(* Reprezentacja IL*)
        LD      bi101     (* bieżący wynik := bi101 *)
        ANDN    bi102     (* bieżący wynik := bi101 AND not(bi102) *)
        ST      bo10      (* bo10 := bieżący wynik *)
        LD      bi51      (* bieżący wynik := bi51;
        &       bi52      (* bieżący wynik := bi51 AND bi52 *)
        &       bi53      (* bieżący wynik := (bi51 AND bi52) AND bi53 *)
        ST      bo5       (* bo5 := bieżący wynik *)
```

>=1 OR



Uwaga: Dla tego operatora ilość wejść może zostać zwiększona do więcej niż dwa.

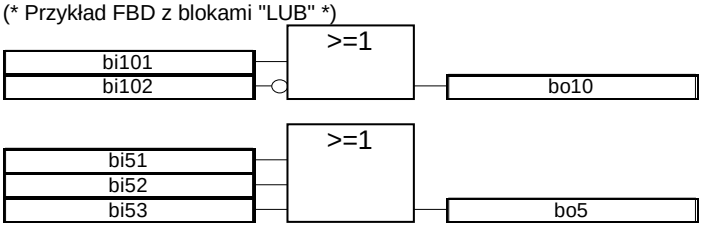
Argumenty:

(wejścia)	BINARNE
wyjście	BINARNE LUB składników wejściowych

Opis:

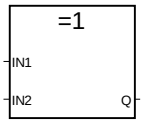
Logiczne LUB dwóch lub większej liczby składników.

```
(* Przykład FBD z blokami "LUB" *)
```




```
(* Reprezentacja ST: *)
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;

(* Reprezentacja IL: *)
LD      bi101
ORN     bi102
ST      bo10
LD      bi51
OR      bi52
OR      bi53
ST      bo5
```

=1 XOR



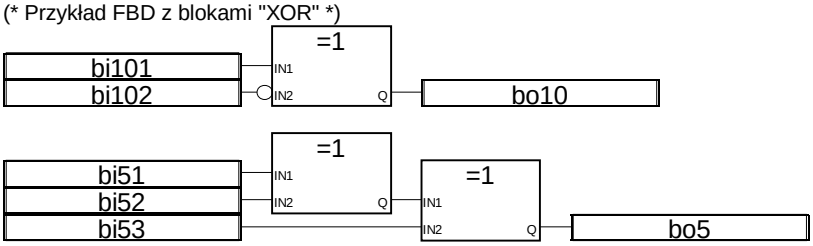
Argumenty:

IN1	BINARNE
IN2	BINARNE
Q	BINARNE LUB wyłączające składników wejściowych

Opis:

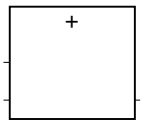
Logiczne LUB wyłączające dwóch lub większej liczby składników.

(* Przykład FBD z blokami "XOR" *)




```
(* Reprezentacja ST: *)
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;

(* Reprezentacja IL: *)
LD      bi101
XORN    bi102
ST      bo10
LD      bi51
XOR     bi52
XOR     bi53
ST      bo5
```

+



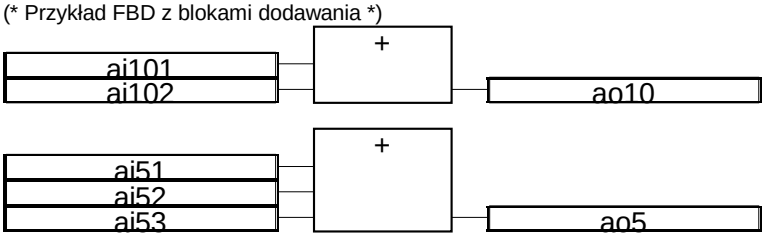
Uwaga: Dla tego operatora ilość wejść może zostać zwiększona do więcej niż dwa.

Argumenty:

(wejścia)	INT-REAL	może mieć wartość INTEGER lub REAL (wszystkie wejścia muszą mieć taki sam format)
wyjście	INT-REAL	dodawanie składników wejściowych ze znakiem

Opis:
Dodawanie dwóch lub większej ilości zmiennych analogowych.

(* Przykład FBD z blokami dodawania *)

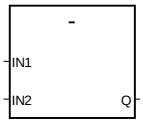


(* Reprezentacja ST: *)
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;

(* Reprezentacja IL: *)

LD	ai101
ADD	ai102
ST	ao10
LD	ai51
ADD	ai52
ADD	ai53
ST	ao5

-

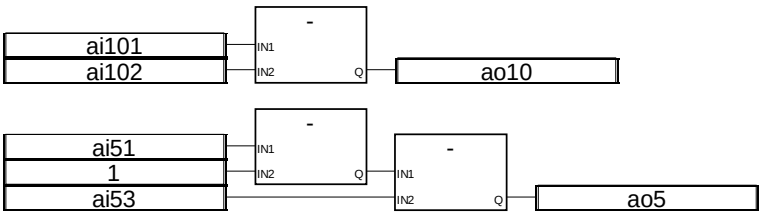


Argumenty:

IN1	INT-REAL	może mieć wartość INTEGER lub REAL
IN2	INT-REAL	(IN1 i IN2 muszą mieć taki sam format)
Q	odejmowanie INT-REAL (pierwszy - drugi)	

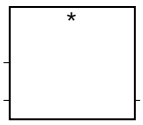
Opis:
Odejmowanie dwóch zmiennych analogowych (pierwsza - druga).

(* Przykład FBD z blokami odejmowania *)




```
 (* Reprezentacja ST: *)
ao10 := ai101 - ai102;
ao5 := (ai51 - 1) - ai53;

(* Reprezentacja IL: *)
LD          ai101
SUB         ai102
ST          ao10
LD          ai51
SUB         1
SUB         ai53
ST          ao5
```

*



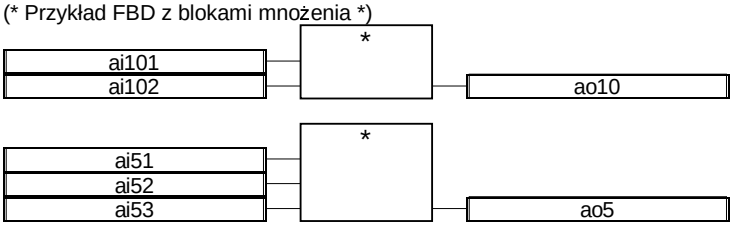
Uwaga: Dla tego operatora ilość wejść może zostać powiększona do więcej niż dwóch.

Argumenty:

(wejścia)	INT-REAL	może mieć wartość INTEGER lub REAL (wszystkie wejścia muszą mieć taki sam format)
wyjście	INT-REAL	mnożenie składników wejściowych ze znakiem

Opis:
Mnożenie dwóch lub większej ilości zmiennych analogowych.

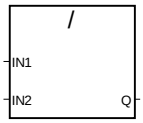
(* Przykład FBD z blokami mnożenia *)



(* Reprezentacja ST *)

```
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;

(* Reprezentacja IL: *)
LD      ai101
MUL     ai102
ST      ao10
LD      ai51
MUL     ai52
MUL     ai53
ST      ao5
```

/



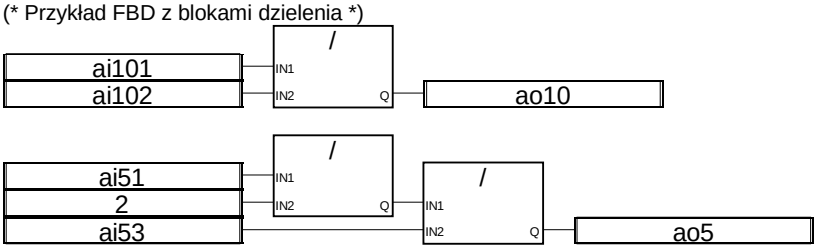
Argumenty:

IN1	INT-REAL	może mieć wartość INTEGER lub REAL (argument)
IN2	INT-REAL	niezerowa wartość analogowa (dzielnik) (IN1 i IN2 muszą mieć taki sam format)
Q	INT-REAL	dzielenie wielkości integer lub real IN1 przez IN2 ze znakiem

Opis:

Dzielenie dwóch zmiennych analogowych (pierwsza podzielona przez drugą).

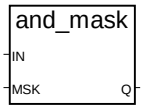
(* Przykład FBD z blokami dzielenia *)




```
(* Reprezentacja ST: *)
ao10 := ai101 / ai102;
ao5 := (ai5 / 2) / ai53;

(* Reprezentacja IL: *)
LD      ai101
DIV     ai102
ST      ao10
LD      ai51
```

```
DIV     2
DIV     ai53
ST      ao5
```

AND_MASK



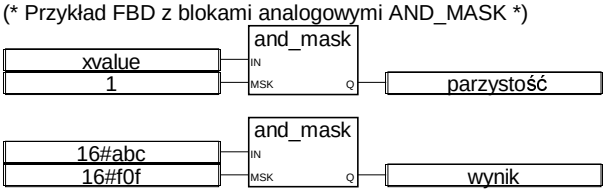
Argumenty:

IN	INT	musi mieć format integer
MSK	INT	musi mieć format integer
Q	INT	logiczne I wielkości IN oraz MSK od bitu do bitu

Opis:

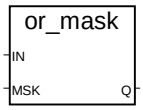
Operacja bitowa I na zmiennych integer.

(* Przykład FBD z blokami analogowymi AND_MASK *)




```
(* Reprezentacja ST: *)
parzystość := AND_MASK (xvalue, 1); (* 1 jeśli xvalue jest nieparzysta *)
wynik := AND_MASK (16#abc, 16#f0f); (* równe 16#a0c *)

(* Reprezentacja IL: *)
LD          xvalue
AND_MASK    1
ST          parzystość
LD          16#abc
AND_MASK    16#f0f
ST          wynik
```

OR_MASK



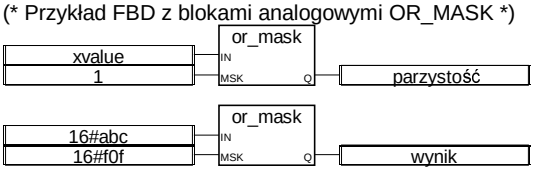
Argumenty:

IN	INT	musi mieć format integer

MSK	INT	musi mieć format integer
Q	INT	logiczne LUB wielkości IN oraz MSK od bitu do bitu

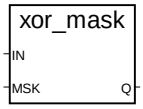
Opis:
Operacja bitowa LUB na zmiennych integer.

(* Przykład FBD z blokami analogowymi OR_MASK *)




```
(* Reprezentacja ST: *)
is_odd := OR_MASK (xvalue, 1); (* sprawia, iż wartość jest zawsze nieparzysta *)
wynik := OR_MASK (16#abc, 16#f0f); (* równe 16#fbf *)

(* Reprezentacja IL: *)
LD          xvalue
OR_MASK     1
ST          is_odd
LD          16#abc
OR_MASK     16#f0f
ST          wynik
```

XOR_MASK

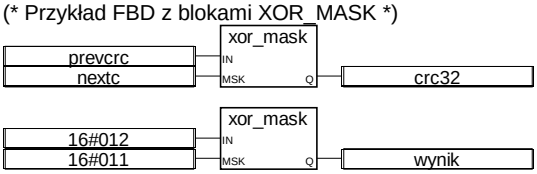


Argumenty:

IN	INT	musi mieć format integer
MSK	INT	musi mieć format integer
Q	INT	logiczne LUB wyłączające wielkości IN oraz MSK od bitu do bitu

Opis:
Operacja bitowa ex-LUB na zmiennych integer

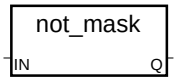
(* Przykład FBD z blokami XOR_MASK *)




```
(* Reprezentacja ST: *)
crc32 := XOR_MASK (prevcrc, nextc);
```

```
wynik := XOR_MASK (16#012, 16#011); (* równe 16#003 *)

(* Reprezentacja IL: *)
LD          prevcrc
XOR_MASK    nextc
ST          crc32
LD          16#012
XOR_MASK    16#011
ST          wynik
```

NOT_MASK

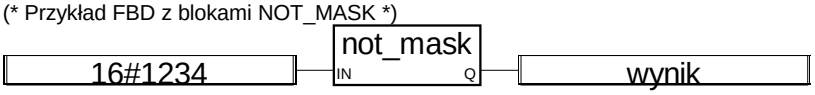


Argumenty :

IN	INT	musi mieć format integer
Q	INT	negacja 32-bitowego IN od bitu do bitu

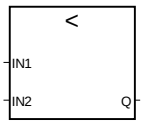
Opis:
Negacja bitowa zmiennej integer

```
(* Przykład FBD z blokami NOT_MASK *)
```




```
(*Reprezentacja ST: *)
wynik := NOT_MASK (16#1234);
(* wynik to 16#FFFF_EDCB *)

(* Reprezentacja IL: *)
LD          16#1234
NOT_MASK
ST          wynik
```

<



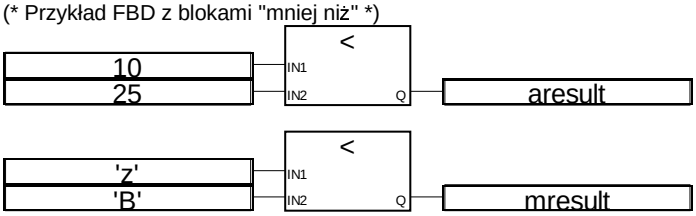
Argumenty:

IN1	INT-REAL-TMR-MSG	
IN2	INT-REAL-TMR-MSG	oba wejścia muszą posiadać taki sam typ

Q BINARNA PRAWDA, jeśli IN1 < IN2

Opis:
Sprawdza , czy jedna wartość jest MNIEJSZA NIŻ inna (analogowa, timer lub komunikat)

(* Przykład FBD z blokami "mniej niż" *)

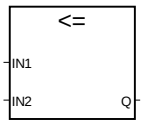


(* Reprezentacja ST: *)
aresult := (10 < 25); (* aresult to PRAWDA *)
mresult := ('z' < 'B'); (* mresult to FAŁSZ *)

(* Reprezentacja IL: *)

LD	10
LT	25
ST	aresult
LD	'z'
LT	'B'
ST	mresult

<=

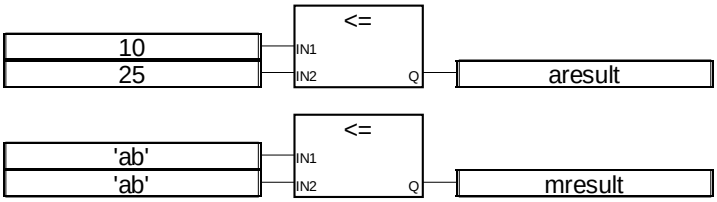


Argumenty:

IN1	INT-REAL-MSG
IN2	INT-REAL-MSG oba wejścia muszą być takiego samego typu
Q	BINARNA PRAWDA, jeśli IN1 <= IN2

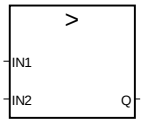
Opis:
Sprawdza, czy jedna wartość jest MNIEJSZA lub RÓWNA drugiej (analogowe lub komunikaty)

(* Przykład FBD z blokami "mniejsze lub równe" *)




```
(* Reprezentacja ST: *)
aresult := (10 <= 25); (* aresult to PRAWDA *)
mresult := ('ab' <= 'ab'); (* mresult to PRAWDA*)

(* Reprezentacja IL: *)
LD      10
LE      25
ST      aresult
LD      'ab'
LE      'ab'
ST      mresult
```

>

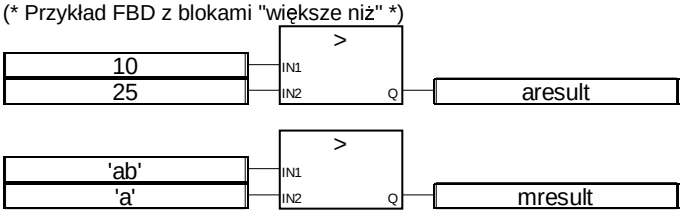


Argumenty:

IN1	INT-REAL-TMR-MSG	
IN2	INT-REAL-TMR-MSG	oba wejścia muszą być takiego samego typu
Q	PRAWDA, jeśli IN1 > IN2	

Opis:

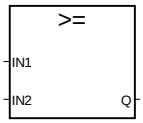
Sprawdza, czy jedna wartość jest WIĘKSZA NIŻ inna (analogowe, timer lub komunikaty)




```
(* Reprezentacja ST: *)
aresult := (10 > 25); (* aresult to FAŁSZ *)
mresult := ('ab' > 'a'); (* mresult to PRAWDA *)
```

```
(* Reprezentacja IL: *)
LD          10
GT          25
ST          aresult
LD          'ab'
GT          'a'
ST          mresult
```

>=



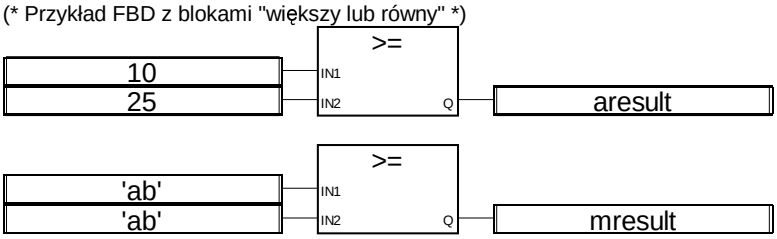
Argumenty:

IN1	INT-REAL-MSG
IN2	INT-REAL-MSG oba wejścia muszą być takiego samego typu
Q	BINARNA PRAWDA, jeśli IN1 >= IN2

Opis:

Sprawdza, czy jedna wartość jest WIĘKSZA lub RÓWNA innej (analogowe lub komunikaty)

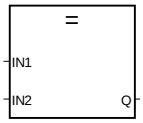
(* Przykład FBD z blokami "większy lub równy" *)




```
(* Reprezentacja ST: *)
aresult := (10 >= 25); (* aresult to FAŁSZ *)
mresult := ('ab' >= 'ab'); (* mresult to PRAWDA *)
```

```
(* Reprezentacja IL: *)
LD          10
GE          25
ST          aresult
LD          'ab'
GE          'ab'
ST          mresult
```

=



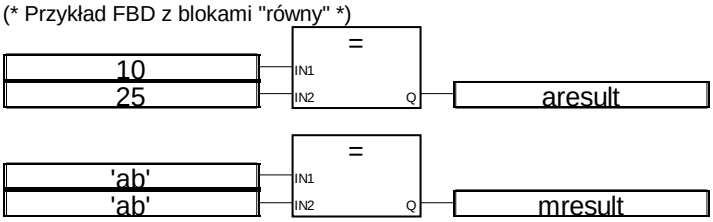
Argumenty:

IN1	INT-REAL-MSG
IN2	INT-REAL-MSG oba wejścia muszą być takiego samego typu
Q	BINARNA PRAWDA, jeśli IN1 = IN2

Opis:

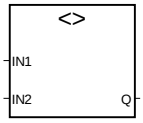
Sprawdza, czy jedna wartość jest RÓWNA innej (analogowe lub komunikaty)

(* Przykład FBD z blokami "równy" *)




```
(* Reprezentacja ST: *)
aresult := (10 = 25); (* aresult to FAŁSZ *)
mresult := ('ab' = 'ab'); (* mresult to PRAWDA *)

(* Reprezentacja IL: *)
LD      10
EQ      25
ST      aresult
LD      'ab'
EQ      'ab'
ST      mresult
```

<>



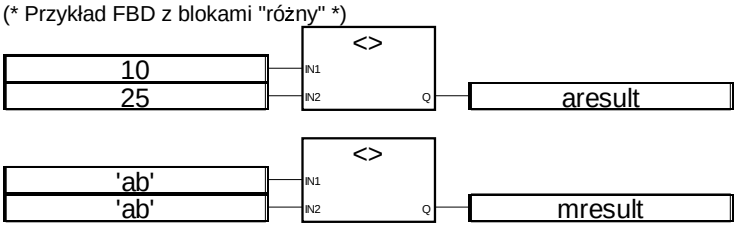
Argumenty:

IN1	INT-REAL-MSG
IN2	INT-REAL-MSG oba wejścia muszą być takiego samego typu
Q	BINARNA PRAWDA, jeśli pierwszy <> drugi

Opis:

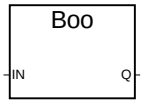
Sprawdza, czy jedna wartość jest RÓŻNA drugiej (wartości analogowe lub komunikaty)

(* Przykład FBD z blokami "różny" *)




```
(* Reprezentacja ST: *)
aresult := (10 <> 25); (* aresult to PRAWDA *)
mresult := ('ab' <> 'ab'); (* mresult to FAŁSZ *)

(* Reprezentacja IL: *)
LD      10
NE      25
ST      aresult
LD      'ab'
NE      'ab'
ST      mresult
```

BOO



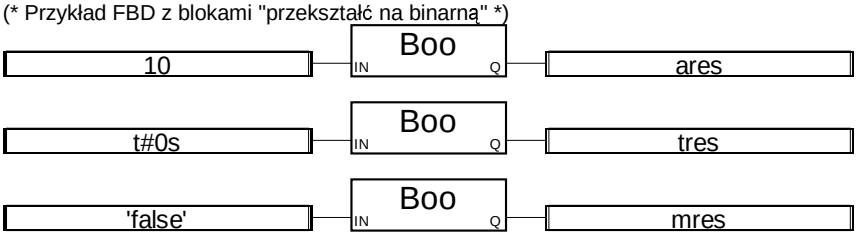
Argumenty:

IN	ANY	dowolna wartość nie-binarna
Q	BOO	PRAWDA dla niezerowej wartości liczbowej FAŁSZ dla zerowej wartości liczbowej PRAWDA dla komunikatu 'PRAWDZIWEGO' FAŁSZ dla komunikatu 'FAŁSZYWEGO'

Opis:

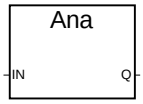
Przekształca dowolną zmienną na binarną

(* Przykład FBD z blokami "przekształć na binarną" *)



```
(* Reprezentacja ST: *)
ares := BOO (10);                (* ares to PRAWDA *)
tres := BOO (t#0s);              (* tres to FAŁSZ *)
mres := BOO ('false');           (* mres to FAŁSZ *)

(* Reprezentacja IL: *)
LD          10
BOO
ST          ares
LD          t#0s
BOO
ST          tres
LD          'false'
BOO
ST          mres
```

ANA



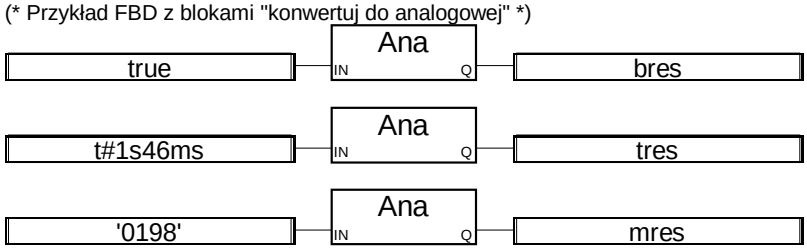
Argumenty:

IN	ANY	dowolna wartość analogowa nie typu integer
Q	INT	0 jeśli IN to FAŁSZ / 1 jeśli IN to PRAWDA
		liczba milisekund dla licznika
		część integer liczby analogowej real
		liczba dziesiętna reprezentowana przez łańcuch

Opis:

Konwertuje dowolną zmienną na zmienną integer

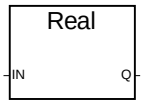
(* Przykład FBD z blokami "konwertuj do analogowej" *)




```
(* Reprezentacja ST: *)
bres := ANA (true);              (* bres to 1 *)
tres := ANA (t#1s46ms);          (* tres to 1046 *)
mres := ANA ('0198');            (* mres to 198 *)

(* Reprezentacja IL: *)
LD          true
```

```
ANA
ST          bres
LD          t#1s46ms
ANA
ST          tres
LD          '0198'
ANA
ST          mres
```

REAL



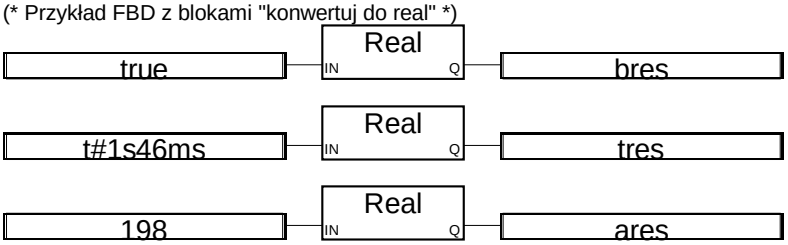
Argumenty:

IN	BOO-INT-TMR	dowolna wartość analogowa nie typu real (różna od komunikatu)
Q	REAL	0.0 jeśli IN to FAŁSZ / 1.0 jeśli IN to PRAWDA liczba milisekund dla licznika odpowiednik analogowej liczby integer

Opis:

Konwertuje dowolną zmienną do zmiennej real

(* Przykład FBD z blokami "konwertuj do real" *)




```
(* Reprezentacja ST: *)
bres := REAL (true);                (* bres to 1.0 *)
tres := REAL (t#1s46ms);            (* tres to 1046.0 *)
ares := REAL (198);                 (* ares to 198.0 *)

(* Reprezentacja IL: *)
LD          true
REAL
ST          bres
LD          t#1s46ms
REAL
ST          tres
LD          198
```

REAL
ST ares

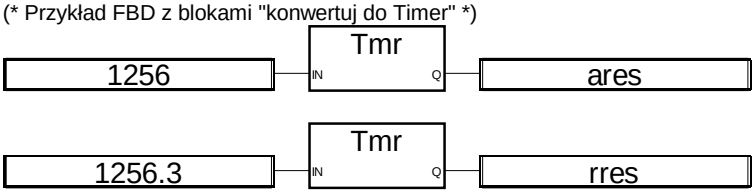
TMR

Argumenty:

IN	INT-REAL	dowolna wartość różna od timer IN (lub część integer liczby IN jeśli jest to liczba real) jest liczbą milisekund
Q	TIMER	wartość czasowa reprezentowana przez IN

Opis:
Konwertuje dowolną zmienną analogową do zmiennej typu timer

(* Przykład FBD z blokami "konwertuj do Timer" *)



(* Reprezentacja ST: *)
ares := TMR (1256); (* ares := t#1s256ms *)
rres := TMR (1256.3); (*rres := t#1s256ms *)

(* Reprezentacja IL: *)
LD 1256
TMR
ST ares
LD 1256.3
TMR
ST rres

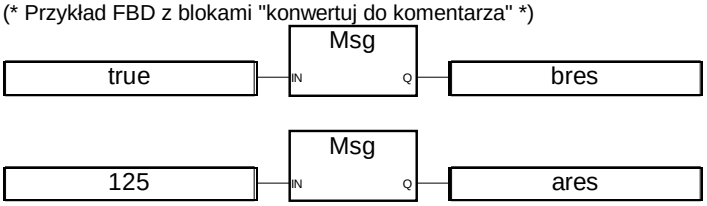
MSG

Argumenty:
IN BOO-

	INT-REA	dowolna wartość różna od komentarza
Q	MSG	'fałsz' lub 'prawda' jeśli IN jest wartością binarną reprezentacja dziesiętna, jeśli IN jest wartością analogową

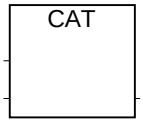
Opis:
Konwertuje dowolną zmienną do zmiennej typu komentarz

(* Przykład FBD z blokami "konwertuj do komentarza" *)




```
(* Reprezentacja ST: *)
bres := MSG (true); (* bres to 'PRAWDA' *)
ares := MSG (125); (* ares to '125' *)

(* Reprezentacja IL: *)
LD      true
MSG
ST      bres
LD      125
MSG
ST      ares
```

CAT



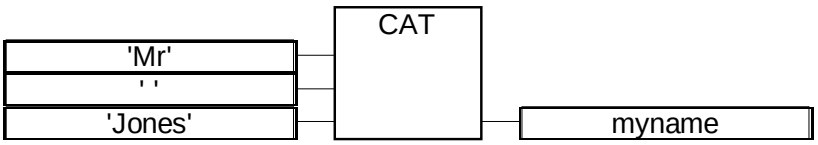
Uwaga: Dla tego operatora ilość wejść może zostać zwiększona do więcej niż dwa.

Argumenty:

(wejścia)	MSG	(suma wszystkich długości komunikatów nie może przekraczać dopuszczalnej wielkości komunikatu wyjściowego)
wyjście	MSG	łączenie komunikatów wejściowych

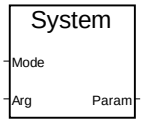
Opis:
łączy kilka komunikatów w jeden

(* Przykład FBD z blokami "łączenie komunikatów" *)




```
(* Reprezentacja ST: zastosuj operator + *)
myname := ('Mr' + ' ') + 'Jones';
(* oznacza: myname := 'Mr Jones' *)

(* Reprezentacja IL: *)
LD          'Mr'
ADD         ' '
ADD         'Jones'
ST          myname
```

SYSTEM



Argumenty:

Mode	INT	określa parametr systemowy i tryb dostępu
Arg	INT-TMR	nowa wartość dla dostępu typu "zapis"
Param	INT	wartość udostępnionego parametru

Opis:

Dostęp do parametrów systemu

Poniżej znajduje się lista dostępnych komend (wcześniej zdefiniowanych słów kluczowych) funkcji SYSTEM:

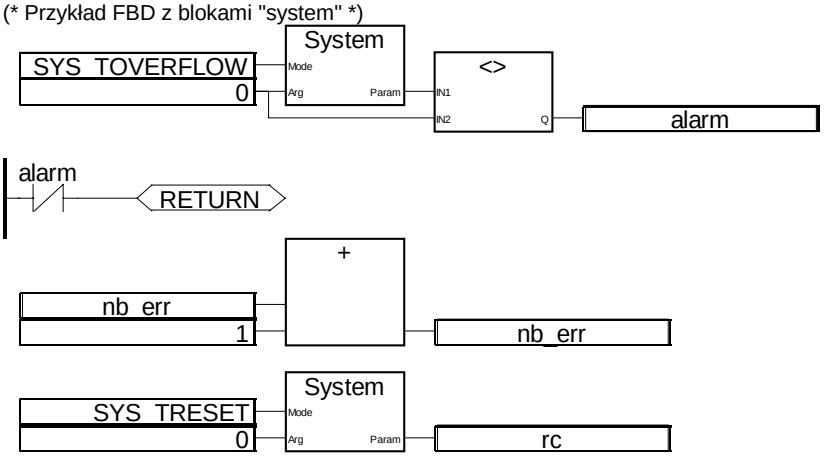
komenda	*znaczenie*
SYS_TALLOWED	odczytaj dopuszczalne parametry czasowe cyklu
SYS_TCURRENT	odczytaj bieżące parametry czasowe cyklu
SYS_TMAXIMUM	odczytaj maksymalne parametry czasowe cyklu
SYS_TOVERFLOW	odczytaj przepełnienia parametrów czasowych cyklu
SYS_TRESET	zeruj liczniki parametrów czasowych
SYS_TWRITE	zmień parametry czasowe cyklu
SYS_ERR_TEST	sprawdź istnienie błędów wykonywania
SYS_ERR_READ	odczytaj najstarszy z błędów wykonania

Oto oczekiwane argumenty wstępnie zdefiniowanych funkcji w funkcji SYSTEM:

komenda	argument	wartość zwracana
SYS_TALLOWED	0	dopuszczalne parametry czasowe cyklu
SYS_TCURRENT	0	bieżące parametry czasowe cyklu
SYS_TMAXIMUM	0	maksymalne wykryte parametry czasowe

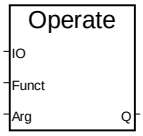
SYS_TOVERFLOW	0	ilość przepełnień parametrów czasowych
SYS_TRESET	0	0
SYS_TWRITE	nowe dopuszcz. parametry czas. cyklu	zapisany czas
SYS_ERR_TEST	0	0 jeśli nie wykryto błędu
SYS_ERR_READ	0	kod najstarszego błędu

(* Przykład FBD z blokami "system" *)




```
(* Reprezentacja ST: *)
alarm := (SYSTEM (SYS_TOVERFLOW, 0) <> 0);
If (alarm) Then
        nb_err := nb_err + 1;
        rc := SYSTEM (SYS_TRESET, 0);
End_If;
```

OPERATE



Argumenty:

IO	ANY	zmienna wejściowa lub wyjściowa
Funct	INT	operacja do wykonania
Arg	INT	argument operacji We/Wy
Q	INT	kontrola zwrotu

Opis:

Dostęp do kanału We/Wy

Znaczenie argumentów OPERATE zależy od implementacji interfejsu We/Wy. Aby dowiedzieć się więcej na temat możliwości bloku OPERATE, patrz - podręcznik sprzętowy lub opis techniczny odpowiedniej karty We/Wy.

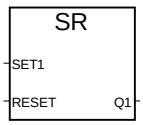
B.9.2 Standardowe bloki funkcyjne

Poniżej podano standardowe bloki funkcyjne obsługiwane przez system ISaGRAF. Takie bloki funkcyjne są wstępnie zdefiniowane i nie muszą być deklarowane w bibliotece.

Binarne:	SR	Przerzutnik z dominującym ustawianiem
	RS	Przerzutnik z dominującym kasowaniem
	R_TRIG	Wykrywanie zbocza narastającego
	F_TRIG	Wykrywanie zbocza opadającego
	SEMA	Semafor
Liczenie:	CTU	Licznik w górę
	CTD	Licznik w dół
	CTUD	Licznik rewersyjny
Timer:	TON	Timer włączany
	TOF	Timer wyłączany
	TP	Timer wyzwalany
Analogowe integer:	CMP	komparator
	STACKINT	Stos zmiennych integer
Analogowe real:	AVERAGE	Obliczanie średniej z N próbek
	HYSTER	Binarna wartość Histerezy różnicy wartości rzeczywistych
	LIM_ALRM	Alarm przekroczenia górnego / dolnego limitu histerezy
	INTEGRAL	Całkowanie po czasie
	DERIVATE	Różniczkowanie po czasie
Generowanie sygnałów:		BLINKGenerator bistabilny
	SIG_GEN	Generator sygnałowy

Uwaga: Kiedy tworzone są nowe bloki funkcyjne "C", to można je wywoływać z języka FBD.

SR



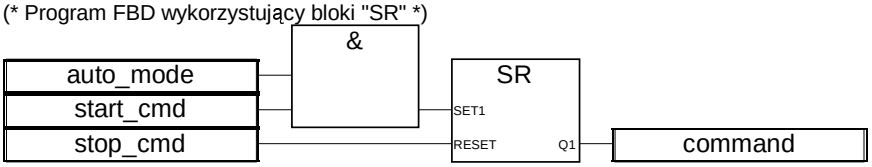
Argumenty:

SET1	BOO	jeśli PRAWDA, ustawia Q1 na PRAWDA (dominacja)
RESET	BOO	jeśli PRAWDA, zeruje Q1 do FAŁSZ
Q1	BOO	wielkość binarna

Opis:
Przerzutnik bistabilny z dominującym wejściem ustawiającym.: Patrz tablica prawdy:

SET1	*RESET*	*Q1*	*wynik Q1*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(* Program FBD wykorzystujący bloki "SR" *)

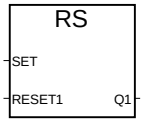


(* Reprezentacja ST: Można przypuszczać, iż SR1 jest instancją bloku SR *)
SR1((auto_mode & start_cmd), stop_cmd);
command := SR1.Q1;

(* Reprezentacja IL: *)

LD	auto_mode
AND	start_cmd
ST	SR1.set1
LD	stop_cmd
ST	SR1.reset
CAL	SR1
LD	SR1.Q1
ST	command

RS



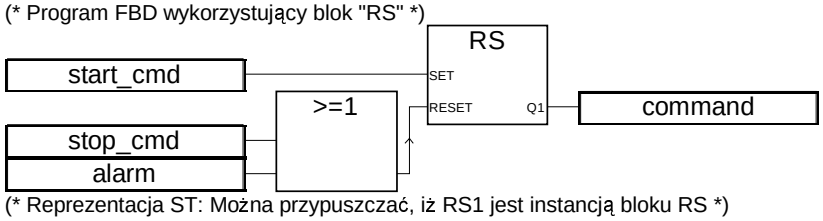
Argumenty:

SET	BOO	jeśli PRAWDA, ustawia Q1 na PRAWDA
RESET1	BOO	jeśli PRAWDA, ZERUJE Q1 do FAŁSZ (dominacja)
Q1	BOO	wartość binarna

Opis:
Przerzutnik bistabilny z dominującym wejściem kasującym.: Patrz tablica prawdy:

SET	*RESET1*	*Q1*	*wynik Q1*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

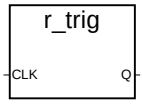
(* Program FBD wykorzystujący blok "RS" *)



(* Reprezentacja ST: Można przypuszczać, iż RS1 jest instancją bloku RS *)
```
RS1(start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;
```

(* Reprezentacja IL: *)
```
LD      start_cmd
ST      RS1.set
LD      stop_cmd
OR      alarm
ST      RS1.reset1
CAL     RS1
LD      RS1.Q1
ST      command
```

R_TRIG



Argumenty:

CLK	BOO	dowolna zmienna dwuwartościowa
Q	BOO	PRAWDA, kiedy CLK wzrasta od FAŁSZ do PRAWDA FAŁSZ we wszystkich pozostałych przypadkach

Opis:
Wykrywa zbocze narastające zmiennej binarnej

```
(* Program FBD wykorzystujący blok "R_TRIG" *)

(* Reprezentacja ST: Można przypuszczać, iż R_TRIG1 jest instancją bloku R_TRIG *)
R_TRIG1(cmd);
nb_edge := ANA(R_TRIG1.Q) + nb_edge;

(* Reprezentacja IL: *)
LD      cmd
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

F_TRIG

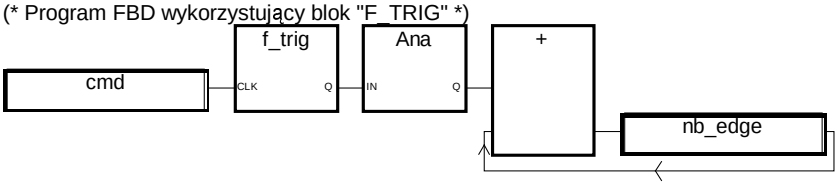
Argumenty:

CLK	BOO	dowolna zmienna binarna
Q	BOO	PRAWDA, kiedy CLK zmienia się z PRAWDA na FAŁSZ FAŁSZ we wszystkich pozostałych przypadkach

Opis:

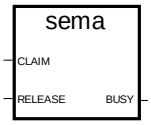
Wykrywa zbocze opadające zmiennej binarnej.

(* Program FBD wykorzystujący blok "F_TRIG" *)




```
(* Reprezentacja ST: Można przypuszczać, iż F_TRIG1 jest instancją bloku F_TRIG *)
F_TRIG1(cmd);
nb_edge := ANA(F_TRIG1.Q) + nb_edge;

(* Reprezentacja IL: *)
LD      cmd
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

SEMA



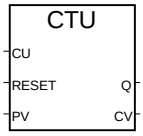
Argumenty:

CLAIM	BINARNA	komenda "testuj i ustaw"
RELEASE	BINARNA	wielkość zwalnia semafor
BUSY	BINARNA	wartość semafora

Opis:

```
(* "x" to zmienna binarna inicjalizowana jako FAŁSZ *)
busy := x;
If claim Then
        x := True;
Else
        If release Then
                busy := False;
                x := False;
        End_if;
End_if;
```

CTU



Argumenty:

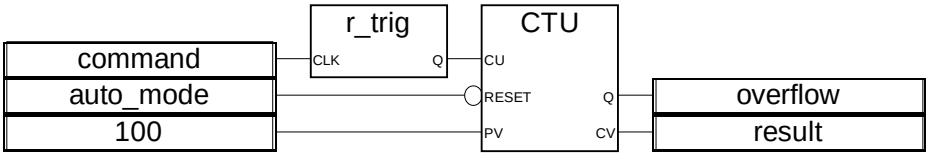
CU	BOO	wejście zliczające (zliczanie, gdy CU to PRAWDA)
RESET	BOO	zeruj komendę (dominacja)
PV	INT	zaprogramowana wartość maksymalna
Q	BOO	przepełnienie: PRAWDA kiedy CV = PV
CV	INT	wynik liczenia

Uwaga: Blok CTU nie wykrywa zboczy narastających czy opadających wejścia zliczającego (CU). Musi on być powiązany z blokiem "R_TRIG" lub "F_TRIG" aby stworzyć licznik impulsowy.

Opis:

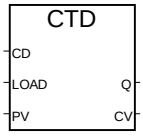
Zliczanie (integer) od 0 w górę do danej wartości 1 po 1

(* Program FBD wykorzystujący blok "CTU" *)




```
(* Reprezentacja ST: Można przypuszczać, iż R_TRIG1 jest instancją bloku R_TRIG, a CTU1
jest instancją bloku CTU *)
CTU1(R_TRIG1(command),NOT(auto_mode),100);
overflow := CTU1.Q;
wynik := CTU1.CV;

(* Reprezentacja IL: *)
LD          command
ST          R_TRIG1.clk
CAL         R_TRIG1
LD          R_TRIG1.Q
ST          CTU1.cu
LDN         auto_mode
ST          CTU1.reset
LD          100
ST          CTU1.pv
CAL         CTU1
LD          CTU1.Q
ST          overflow
LD          CTU1.cv
ST          wynik
```

CTD



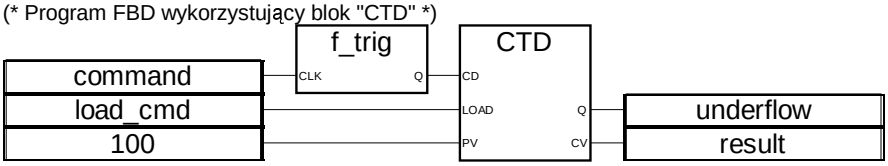
Argumenty:

CD	BOO	wejście zliczające (zliczanie w dół, kiedy CD to PRAWDA)
LOAD	BOO	ładowanie komendy (dominacja) (CV = PV, kiedy LOAD to PRAWDA)
PV	INT	zaprogramowana wartość początkowa
Q	BOO	niedomiar: PRAWDA, kiedy CV = 0
CV	INT	wynik zliczania

Uwaga: Blok CTD nie wykrywa zboczy narastających czy opadających wejścia zliczającego (CU). Musi on być powiązany z blokiem "R_TRIG" lub "F_TRIG" aby stworzyć licznik impulsowy.

Opis:
Zliczanie (integer) od 0 w dół do danej wartości 0 1 po 1

(* Program FBD wykorzystujący blok "CTD" *)



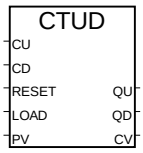
(* Reprezentacja ST: Można przypuszczać, iż F_TRIG1 jest instancją bloku F_TRIG, a CTD1 jest instancją bloku CTD *)

```
CTD1(F_TRIG1(command),load_cmd,100);
underflow := CTD1.Q;
wynik := CTD1.CV;
```

(* Reprezentacja IL: *)

```
LD      command
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ST      CTD1.cd
LD      load_cmd
ST      CTD1.load
LD      100
ST      CTD1.pv
CAL     CTD1
LD      CTD1.Q
ST      underflow
LD      CTD1.cv
ST      wynik
```

CTUD



Argumenty:

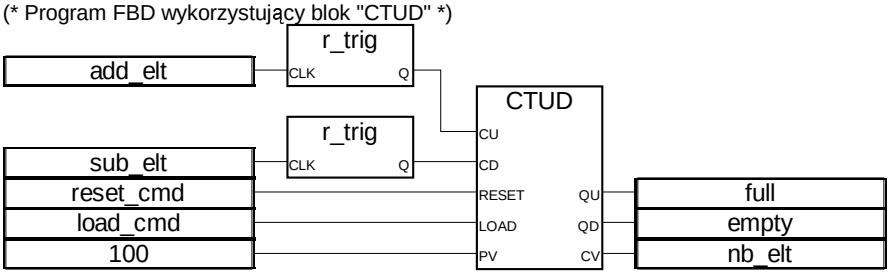
CU	BOO	zliczanie w górę (kiedy CU to PRAWDA)
CD	BOO	zliczanie w dół (kiedy CD to PRAWDA)
RESET	BOO	komenda zerowania (dominacja) (CV = 0, kiedy RESET to PRAWDA)
LOAD	BOO	komenda ładowania (CV = PV, kiedy LOAD to PRAWDA)
PV	INT	zaprogramowana wartość maksymalna
QU	BOO	przepełnienie: PRAWDA, kiedy CV = PV
QD	BOO	niedomiar: PRAWDA, kiedy CV = 0

CV	INT	wynik zliczania

Uwaga: Blok CTUD nie wykrywa zboczy narastających ani opadających wejść zliczających (CU i CD). Musi on być powiązany z blokiem "R_TRIG" lub "F_TRIG" aby stworzyć licznik impulsowy.

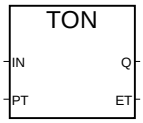
Opis:
Zliczanie (integer) od 0 w górę do danej wartości 1 po 1
lub od danej wartości w dół 0 1 po 1

(* Program FBD wykorzystujący blok "CTUD" *)




```
(* Reprezentacja ST: Można przypuszczać, iż R_TRIG1 i R_TRIG2 są dwoma instancjami
bloku R_TRIG, a CTUD1 jest instancją bloku CTUD *)
CTUD1(R_TRIG1(add_elt), R_TRIG2(sub_elt), reset_cmd, load_cmd,100);
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;

(* Reprezentacja IL: *)
LD      add_elt
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTUD1.cu
LD      sub_elt
ST      R_TRIG2.clk
CAL     R_TRIG2
LD      R_TRIG2.Q
ST      CTUD1.cd
LD      reset_cmd
ST      CTUD1.reset
LD      load_cmd
ST      CTUD1.load
LD      100
ST      CTUD1.pv
CAL     CTUD1
LD      CTUD1.QU
ST      full
LD      CTUD1.QD
ST      empty
LD      CTUD1.CV
ST      nb_elt
```

TON



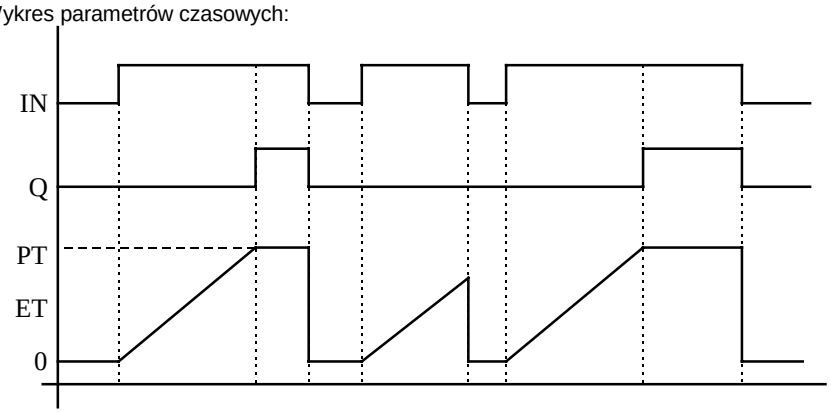
Argumenty:

IN	BOO	Zbocze narastające uruchamia zwiększanie wartości licznika wewnętrznego Zbocze opadające zatrzymuje i zeruje licznik wewnętrzny
PT	TMR	wartość maksymalna czasu zliczania
Q	BOO	Jeśli PRAWDA, gdy upłyną maksymalny czasu zliczania
ET	TMR	bieżąca wartość licznika czasu

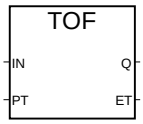
Opis:

Zwiększa licznik wewnętrzny do zadanej wartości.

Wykres parametrów czasowych:



TOF



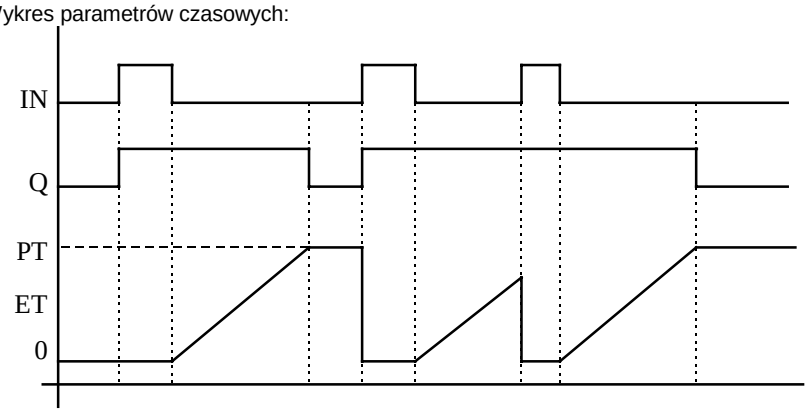
Argumenty:

IN	BOO	Zbocze narastające uruchamia zwiększanie wartości licznika wewnętrznego Zbocze opadające zatrzymuje i zeruje licznik wewnętrzny
PT	TMR	wartość maksymalna czasu zliczania

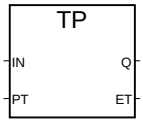
Q	BOO	Jeśli PRAWDA, gdy maksymalny czasu zliczania nie upłynął
ET	TMR	bieżąca wartość licznika czasu

Opis:
Zwiększa licznik wewnętrzny do zadanej wartości.

Wykres parametrów czasowych:



TP

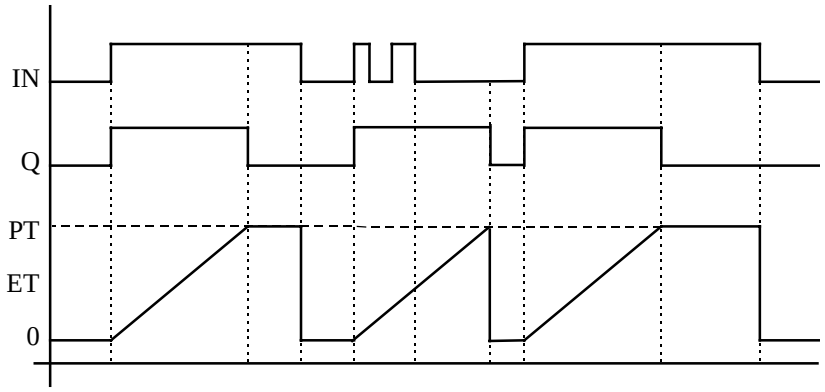


Argumenty:

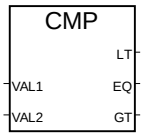
IN	BOO	Zbocze narastające uruchamia zwiększanie wartości licznika wewnętrznego (jeśli jeszcze nie się zwiększa) Jeśli wartością jest FAŁSZ i tylko wtedy, gdy czas licznika jest wykorzystany, to licznik wewnętrzny jest zerowany Żadna zmiana IN podczas zliczania nie odnosi skutku
PT	TMR	wartość maksymalna czasu zliczania
Q	BOO	Jeśli PRAWDA: licznik zlicza
ET	TMR	bieżąca wartość licznika czasu

Opis:
Zwiększa licznik wewnętrzny do zadanej wartości.

Wykres parametrów czasowych:



CMP



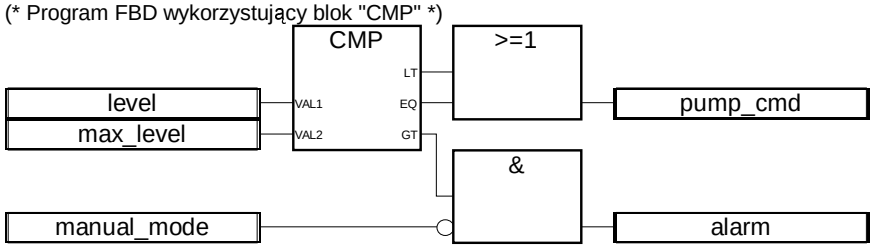
Argumenty:

VAL1	INT	dowolna wartość analogowa integer ze znakiem
VAL2	INT	dowolna wartość analogowa integer ze znakiem
LT	BOO	PRAWDA, jeśli val1 jest mniejsza niż val2
EQ	BOO	PRAWDA, jeśli val1 jest równa val2
GT	BOO	PRAWDA, jeśli val1 jest większa niż val2

Opis:

Porównuje dwie wartości: stwierdza, czy są one równe, czy też pierwsza jest większa czy mniejsza od drugiej.

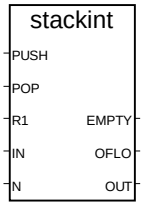
(* Program FBD wykorzystujący blok "CMP" *)




```
(* Reprezentacja ST: Można przypuszczać, iż CMP1 jest instancją bloku CMP *)
CMP1(level, max_level);
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT AND NOT(manual_mode);
```

```
(* Reprezentacja IL: *)
LD      level
ST      CMP1.val1
LD      max_level
ST      CMP1.val2
CAL     CMP1
LD      CMP1.LT
OR      CMP1.EQ
ST      pump_cmd
LD      CMP1.GT
ANDN    manual_mode
ST      alarm
```

STACKINT



Argumenty:

PUSH	BOO	komenda PUSH (tylko na zboczu narastającym) kładzie na stosie wartość IN
POP	BOO	komenda POP (tylko na zboczu narastającym) zdejmuje ze stosu ostatnią wstawioną wartość (wierzch stosu)
R1	BOO	usuwa całą zawartość stosu
IN	INT	wstawiana wartość
N	INT	rozmiar stosu definiowany przez aplikację
EMPTY	BOO	PRAWDA, jeśli stos jest pusty
OFLO	BOO	przepełnienie: PRAWDA, jeśli stos jest pełny
OUT	INT	wartość na wierzchu stosu

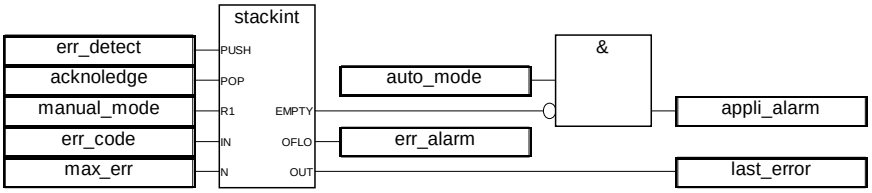
Opis:

Zarządzanie stosem zmiennych integer.

Blok funkcyjny "STACKINT" obsługuje wykrywanie zbocza narastającego zarówno dla komendy PUSH jak i POP. Maksymalny rozmiar stosu to **128**. Rozmiar stosu **N** definiowany przez aplikację nie może być mniejszy niż 1 ani większy niż 128.

Należy zauważyć, iż wartość OFLO jest ważna tylko po usunięciu zawartości stosu (R1 została ustawiona przynajmniej raz na wartość PRAWDA i z powrotem na FAŁSZ).

(* Program FBD wykorzystujący blok "STACKINT": zarządzanie błędami *)




```
(* Reprezentacja ST: Można przypuszczać, iż STACKINT1 jest instancją bloku STACKINT *)
STACKINT1(err_detect, acknoledge, manual_mode, err_code, max_err);
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;

(* Reprezentacja IL: *)
LD      err_detect
ST      STACKINT1.push
LD      acknoledge
ST      STACKINT1.pop
LD      manual_mode
ST      STACKINT1.r1
LD      err_code
ST      STACKINT1.IN
LD      max_err
ST      STACKINT1.N
CAL     STACKINT1
LD      auto_mode
ANDN    STACKINT1.empty
ST      appli_alarm
LD      STACKINT1.OFLO
ST      err_alarm
LD      STACKINT1.OUT
ST      last_error
```

AVERAGE

Argumenty:

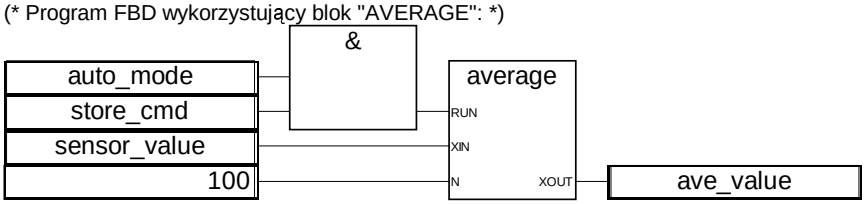
RUN	BOO	PRAWDA=uruchom / FAŁSZ=zeruj
XIN	REAL	dowolna zmienna analogowa typu real
N	INT	ilość próbek określona przez aplikację
XOUT	REAL	obliczanie średniej wartości XIN

Opis:

Przechowuje wartość w każdym cyklu i oblicza wartość średnią wszystkich przechowywanych dotychczas wartości. Przechowywanych jest tylko ostatnie N wartości .

Ilość próbek **N** nie może przekraczać **128**.
Jeśli komenda "**RUN**" ma wartość FAŁSZ (zerowanie), wartość wyjściowa jest równa wartości wejściowej.
Kiedy osiągane jest maksimum przechowywanych wartości N, pierwsza przechowywana wartość jest usuwana przez ostatnią wartość.

(* Program FBD wykorzystujący blok "AVERAGE": *)

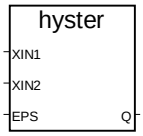


(* Reprezentacja ST: AVERAGE1 to instancja bloku AVERAGE *)
AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;

(* Reprezentacja IL: *)

LD	auto_mode
AND	store_cmd
ST	AVERAGE1.run
LD	sensor_value
ST	AVERAGE1.xin
LD	100
ST	AVERAGE1.N
CAL	AVERAGE1
LD	AVERAGE1.XOUT
ST	ave_value

HYSTER

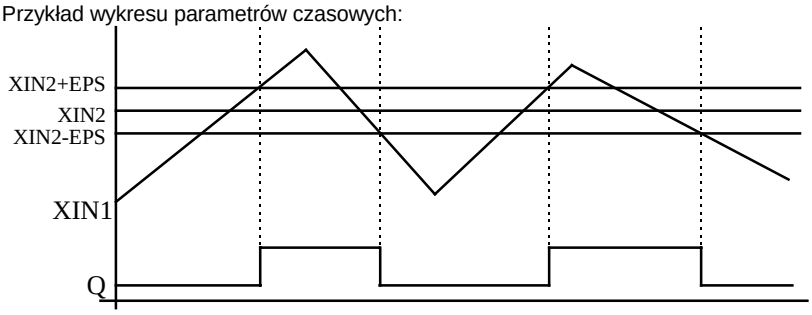


Argumenty:

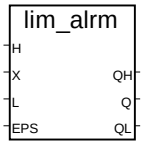
XIN1	REAL	wejściowa dowolna wartość analogowa real
XIN2	REAL	wartość oczekiwana; blok testuje, czy XIN1 przekroczyło XIN2+EPS
EPS	REAL	wartość histerezy(odchylania) (musi być większa od zera)
Q	BOO	PRAWDA, jeśli XIN1 przekroczyło XIN2+EPS i nie jest poniżej XIN2-EPS

Opis:
Histereza wartości typu real dla górnej granicy.

Przykład wykresu parametrów czasowych:



LIM_ALRM

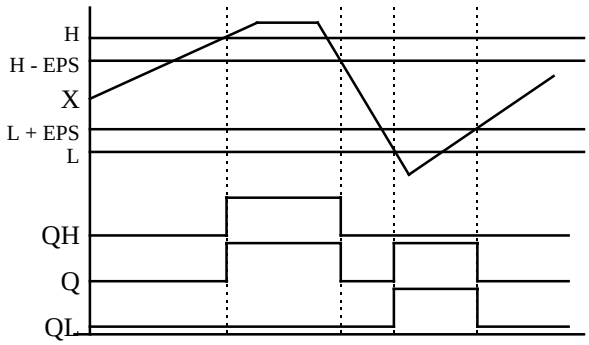


Argumenty:

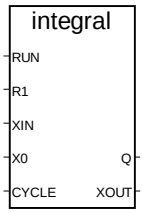
H	REAL	wartość granicy górnej
X	REAL	wejście: dowolna wartość analogowa real
L	REAL	wartość granicy dolnej
EPS	REAL	wartość histerezy (musi być większa od zera)
QH	BOO	alarm "górny": PRAWDA, jeśli X znajduje się ponad granicą górną H
Q	BOO	alarm wyjścia: PRAWDA, jeśli X znajduje się poza granicami
QL	BOO	alarm "dolny": PRAWDA, jeśli X znajduje się poniżej granicy dolnej L

Opis:
Histereza wartości real dla granicy górnej i dolnej .

Histereza jest stosowana na granicy górnej i dolnej. Delta histerezy zastosowana do granicy górnej lub dolnej jest połową parametru EPS. Poniżej przedstawiono przykład wykresu parametrów czasowych:



INTEGRAL



Argumenty:

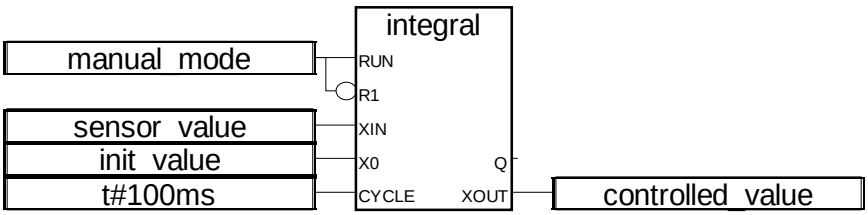
RUN	BOO	tryb: PRAWDA=całkowanie / FAŁSZ=wstrzymanie
R1	BOO	kasowanie zerowania
XIN	REAL	wejście: dowolna wartość analogowa typu real
X0	REAL	wartość początkowa
CYCLE	TMR	okres próbkowania
Q	BOO	Nie R1
XOUT	REAL	wyjście scałkowane

Opis:

Całkowanie wartości typu real.

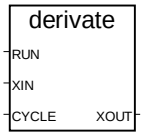
Jeśli wartość parametru "**CYCLE**" jest mniejsza od czasu wykonywania cyklu aplikacji ISaGRAF, okres próbkowania jest czasem cyklu aplikacji.

(* Program FBD wykorzystujący blok "INTEGRAL": *)




```
(* Reprezentacja ST: INTEGRAL1 to instancja bloku INTEGRAL *)
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value, init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;

(* Reprezentacja IL: *)
LD      manual_mode
ST      INTEGRAL1.run
STN     INTEGRAL1.R1
LD      sensor_value
ST      INTEGRAL1.XIN
LD      init_value
ST      INTEGRAL1.X0
LD      t#100ms
ST      INTEGRAL1.CYCLE
CAL     INTEGRAL1
LD      INTEGRAL1.XOUT
ST      controlled_value
```

DERIVATE



Argumenty:

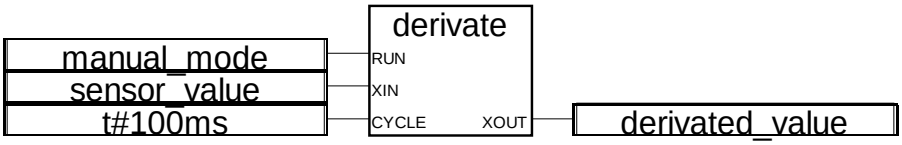
RUN	BOO	tryb: PRAWDA=normalny / FAŁSZ=zerowanie
XIN	REAL	wejście: dowolna wartość analogowa typu real
CYCLE	TMR	okres próbkowania
XOUT	REAL	wyjście różniczki

Opis:

Różniczkowanie wartości rzeczywistej.

Jeśli wartość parametru "**CYCLE**" jest mniejsza od czasu cyklu aplikacji ISaGRAF, okres próbkowania jest czasem cyklu aplikacji.

(* Program FBD wykorzystujący blok "DERIVATE": *)




```
(* Reprezentacja ST: DERIVATE1 to instancja bloku DERIVATE *)
DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;

(* Reprezentacja IL: *)
LD      manual_mode
ST      DERIVATE1.run
LD      sensor_value
ST      DERIVATE1.XIN
LD      t#100ms
ST      DERIVATE1.CYCLE
CAL     DERIVATE1
LD      DERIVATE1.XOUT
ST      derivated_value
```

BLINK

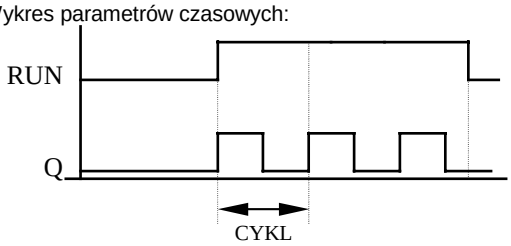
Argumenty:

RUN	BOO	bramkowanie; tryb: PRAWDA=przepuszczanie / FAŁSZ=wyłączenie (Q:=FAŁSZ)
CYCLE	TMR	okres przebiegu
Q	BOO	wyjściowy sygnał generatora

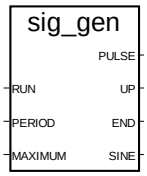
Opis:

Generuje sygnał prostokątny.

Wykres parametrów czasowych:



SIG_GEN



Argumenty:

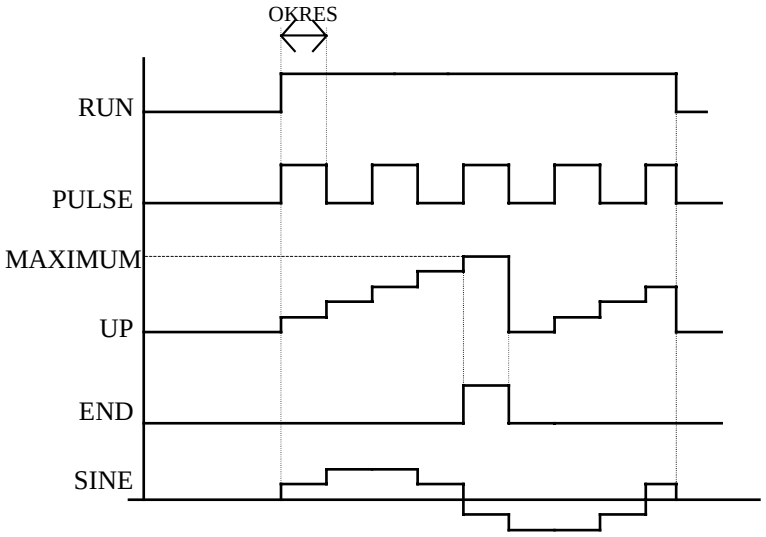
RUN	BOO	bramkowanie; tryb: PRAWDA=przepuszczanie / FAŁSZ=wyłączenie (wyjścia=FAŁSZ)
PERIOD	TMR	okres próbkowania
MAXIMUM	INT	maksymalna wartość zliczania
PULSE	BOO	przebieg prostokątny (negacja po każdej próbce)
UP	INT	zliczanie w górę, zwiększany po każdej próbce
END	BOO	PRAWDA po zakończeniu zliczanie w górę
SINE	REAL	sygnał sinusoidalny (okres = czas trwania zliczania)

Opis:

Generuje sygnały: binarny - przebieg prostokątny; integer - zliczanie w górę; real - przebieg sinusoidalny.

Gdy licznik osiąga wartość maksymalną, zliczanie rozpoczyna się od 0 (zera). Tak więc END ma wartość PRAWDA jedynie podczas 1 próbki.

Wykres parametrów czasowych:



B.9.3 Standardowe funkcje

Oto standardowe funkcje obsługiwane przez system ISaGRAF. Funkcje te są predefiniowane i nie muszą być deklarowane w bibliotece.

Matematyczne:	ABS	Wartość bezwzględna
	EXPT	Wykładnik
	LOG	Logarytm
	POW	Potęga
	SQRT	Pierwiastek kwadratowy
	TRUNC	Zaokrąglenie
Trygonometryczne:	ACOS	Arcus cosinus
	ASIN	Arcus sinus
	ATAN	Arcus tangens
	COS	Cosinus
	SIN	Sinus
	TAN	Tangens
Sterowanie rejestrem:	ROL	Przesunięcie cykliczne w lewo
	ROR	Przesunięcie cykliczne w prawo
	SHL	Przesunięcie w lewo
	SHR	Przesunięcie w prawo
Operacje na danych:	MIN	Minimum
	MAX	Maksimum
	LIMIT	Ogranicznik
	MOD	Modulo
	MUX4	Multiplekser 4 wejściowy
	MUX8	Multiplekser 8 wejściowy
	ODD	Test nieparzystości
	RAND	Wartość losowa
	SEL	Selektor binarny
Konwersja danych:	ASCII	Znak → kod ASCII
	CHAR	Kod ASCII → Znak
Operacje na ciągach		
	DELETE	Usuń podłańcuch
	INSERT	Wstaw łańcuch
	FIND	Znajdź podłańcuch
	MLEN	Pobierz długość łańcucha
	LEFT	Wycinanie części lewej łańcucha
	MID	Wycinanie części środkowej łańcucha
	REPLACE	Zamiana podłańcucha
	RIGHT	Wycinanie części prawej łańcucha
	DAY_TIME	Czas/data systemu
Operacje na tablicach:		
	ARCREATE	Tworzenie tablicy wartości typu integer
	ARREAD	Odczyt elementu tablicy

	ARWRITE	Zapis elementu tablicy

Obsługa pliku binarnego:

F_ROPEN	Otwieranie pliku binarnego (czytanie)
F_WOPEN	Otwieranie pliku binarnego (zapis)
F_CLOSE	Zamknij plik binarny
F_EOF	Test końca pliku binarnego
FA_READ	Odczytanie wartości analogowej z pliku binarnego
FA_WRITE	Zapis wartości analogowej do pliku binarnego
FM_READ	Odczyt komunikatu z pliku binarnego
FM_WRITE	Zapis komunikatu do pliku binarnego

ABS

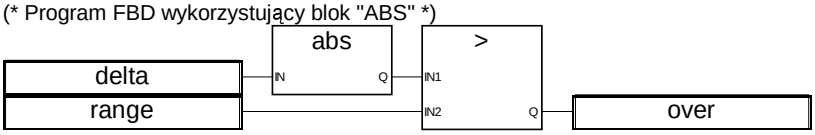
Argumenty:

IN	REAL	dowolna wartość analogowa typu real ze znakiem
Q	REAL	wartość bezwzględna (zawsze dodatnia)

Opis:

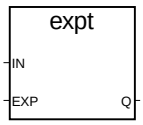
Podaje wartość bezwzględną (dodatnią) wartości typu real.

(* Program FBD wykorzystujący blok "ABS" *)




```
(* Reprezentacja ST: *)
over := (ABS (delta) > range);

(* Reprezentacja IL: *)
LD      delta
ABS
GT      range
ST      over
```

EXPT



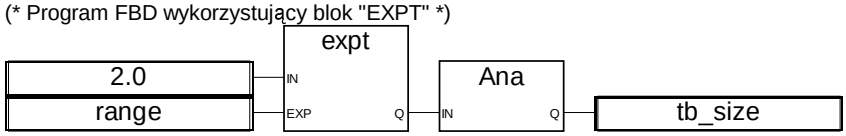
Argumenty:

IN	REAL	dowolna wartość analogowa typu real ze znakiem
EXP	INT	wykładnik analogowy typu integer
Q	REAL	(IN ^{EXP})

Opis:

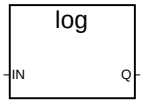
Podaje wynik operacji: (podstawa ^{wykładnik}) w postaci liczby typu real, gdzie 'podstawa', to argument pierwszy, a 'wykładnik' to argument drugi.

(* Program FBD wykorzystujący blok "EXPT" *)




```
(* Reprezentacja ST: *)
tb_size := ANA (EXPT (2.0, range) );

(* Reprezentacja IL: *)
LD      2.0
EXPT    range
ANA
ST      tb_size
```

LOG



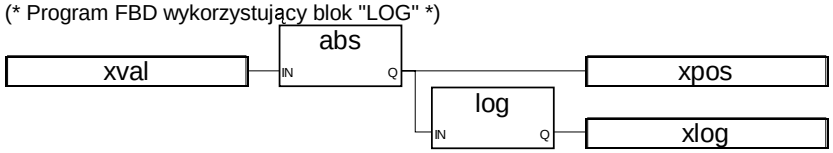
Argumenty:

IN	REAL	musi być większy od zera
Q	REAL	logarytm (podstawa 10) wartości wejściowej

Opis:

Oblicza logarytm (o podstawie 10) z wartości typu real.

(* Program FBD wykorzystujący blok "LOG" *)



```
(* Reprezentacja ST: *)
xpos := ABS (xval);
xlog := LOG (xpos);

(* Reprezentacja IL: *)
LD          xval
ABS
ST          xpos
LOG
ST          xlog
```

POW

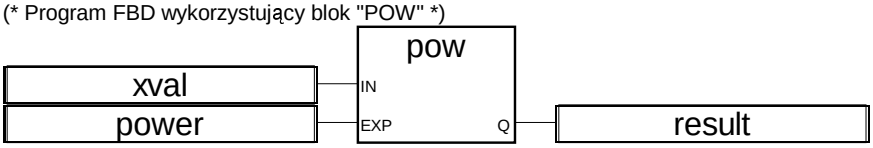
Argumenty:

IN	REAL	liczba analogowa typu real do podniesienia do potęgi
EXP	REAL	potęga (wykładnik)
Q	REAL	(IN ^{EXP})
		1.0 jeśli IN nie wynosi 0.0 i EXP wynosi 0.0
		0.0 jeśli IN wynosi 0.0 i EXP jest ujemny
		0.0 jeśli zarówno IN jak i EXP wynoszą 0.0
		0.0 jeśli IN jest ujemny, a Y nie jest wartością typu integer

Opis:

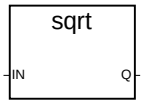
Podaje wynik operacji (podstawa ^{wykładnik}) w postaci liczby real, gdzie 'podstawa', to argument pierwszy, a 'wykładnik' - drugi. Wykładnik to wartość typu real.

(* Program FBD wykorzystujący blok "POW" *)




```
(* Reprezentacja ST: *)
wynik := POW (xval, power);

(* Reprezentacja IL: *)
LD          xval
POW         power
ST          wynik
```

SQRT



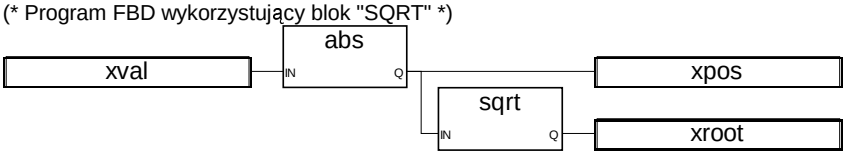
Argumenty:

IN	REAL	musi być większy od lub równy zeru
Q	REAL	pierwiastek kwadratowy wartości wejściowej

Opis:

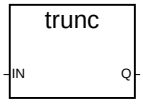
Oblicza pierwiastek kwadratowy wartości typu real.

(* Program FBD wykorzystujący blok "SQRT" *)




```
(* Reprezentacja ST: *)
xpos := ABS (xval);
xroot := SQRT (xpos);

(* Reprezentacja IL: *)
LD      xval
ABS
ST      xpos
SQRT
ST      xrout
```

TRUNC



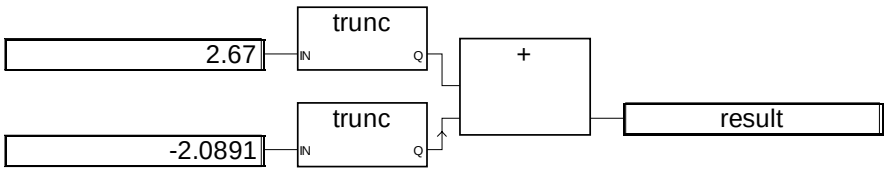
Argumenty:

IN	REAL	dowolna wartość analogowa typu real
Q	REAL	jeśli IN>0, największa wartość integer nie mniejsza lub równa wartości wejściowej jeśli IN<0, najmniejsza wartość integer nie większa lub równa wartości wejściowej

Opis:

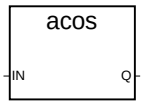
Obcina wartość typu real do uzyskania tylko części typu integer

(* Program FBD wykorzystujący blok "TRUNC" *)




```
(* Reprezentacja ST: *)
wynik := TRUNC (+2.67) + TRUNC (-2.0891);
(* oznacza: wynik := 2.0 + (-2.0) := 0.0; *)

(* Reprezentacja IL: *)
LD          2.67
TRUNC
ST          temporary   (* wynik tymczasowy pierwszego TRUNC *)
LD          -2.0891
TRUNC
ADD         temporary
ST          wynik
```

ACOS



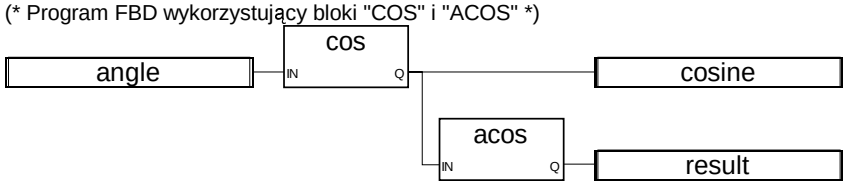
Argumenty:

IN	REAL	musi być ustawiony [-1.0 .. +1.0]
Q	REAL	arcus cosinus wartości wejściowej (w przedziale [0.0 .. PI]) = 0.0 dla wejścia nieprawidłowego

Opis:

Oblicza arcus cosinus wartości typu real.

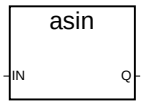
```
(* Program FBD wykorzystujący bloki "COS" i "ACOS" *)
```




```
(* Reprezentacja ST: *)
cosine := COS (angle);
wynik := ACOS (cosine); (* wynik jest równy wartości kąta *)

(* Reprezentacja IL: *)
LD          angle
```

```
COS
ST          cosine
ACOS
ST          wynik
```

ASIN

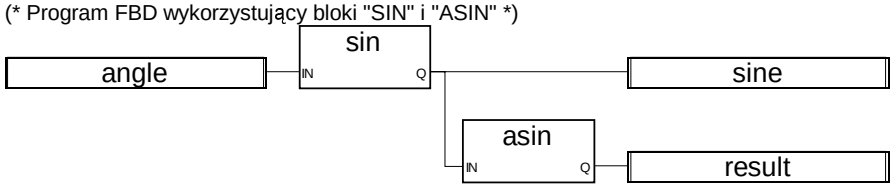


Argumenty:

IN	REAL	musi być ustawiony [-1.0 .. +1.0]
Q	REAL	arcus sinus wartości wejściowej (w przedziale [-PI/2 .. +PI/2]) = 0.0 dla wejścia nieprawidłowego

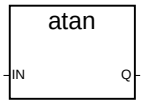
Opis:
Oblicza arcus sinus wartości typu real.

(* Program FBD wykorzystujący bloki "SIN" i "ASIN" *)




```
(* Reprezentacja ST: *)
sine := SIN (angle);
wynik := ASIN (sine); (* wynik jest równy wartości kąta *)

(* Reprezentacja IL: *)
LD          angle
SIN
ST          sine
ASIN
ST          wynik
```

ATAN

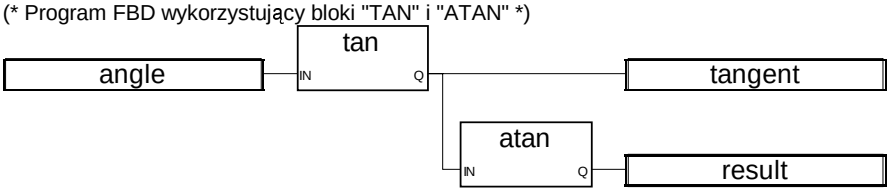


Argumenty:

IN	REAL	dowolna wartość analogowa typu real
Q	REAL	arcus tangens wartości wejściowej (w przedziale [-PI/2 .. +PI/2]) = 0.0 dla wejścia nieprawidłowego

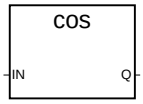
Opis:
Oblicza arcus tangens wartości typu real.

(* Program FBD wykorzystujący bloki "TAN" i "ATAN" *)




```
(* Reprezentacja ST: *)
tangent := TAN (angle);
wynik := ATAN (tangent); (* wynik jest równy wartości kąta *)

(* Reprezentacja IL: *)
LD          angle
TAN
ST          tangent
ATAN
ST          wynik
```

COS

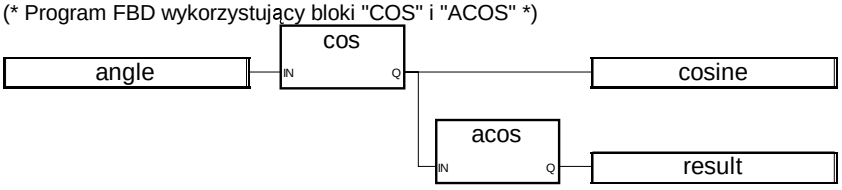


Argumenty:

IN	REAL	dowolna wartość analogowa typu real
Q	REAL	cosinus wartości wejściowej (w przedziale [-1.0 .. +1.0])

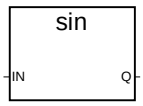
Opis:
Oblicza cosinus wartości typu real.

(* Program FBD wykorzystujący bloki "COS" i "ACOS" *)



```
(* Reprezentacja ST: *)
cosine := COS (angle);
wynik := ACOS (cosine); (* wynik jest równy wartości kąta *)

(* Reprezentacja IL: *)
LD          angle
COS
ST          cosine
ACOS
ST          wynik
```

SIN



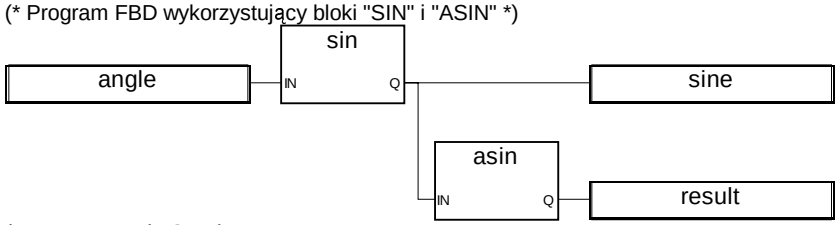
Argumenty:

IN	REAL	dowolna wartość analogowa typu real
Q	REAL	sinus wartości wejściowej (w przedziale [-1.0 .. +1.0])

Opis:

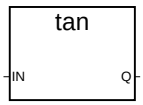
Oblicza sinus wartości typu real.

(* Program FBD wykorzystujący bloki "SIN" i "ASIN" *)




```
(* Reprezentacja ST: *)
sine := SIN (angle);
wynik := ASIN (sine); (* wynik jest równy wartości kąta *)

(* Reprezentacja IL: *)
LD          angle
SIN
ST          sine
ASIN
ST          wynik
```

TAN



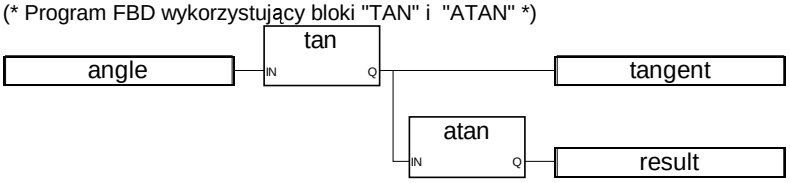
Argumenty:

IN	REAL	nie może być równy PI/2 modulo PI
Q	REAL	tangens wartości wejściowej = 1E+38 dla wejścia nieprawidłowego

Opis:

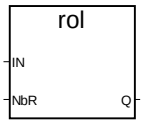
Oblicza tangens wartości typu real.

(* Program FBD wykorzystujący bloki "TAN" i "ATAN" *)




```
(* Reprezentacja ST: *)
tangent := TAN (angle);
wynik := ATAN (tangent); (* wynik jest równy wartości kąta *)

(* Reprezentacja IL: *)
LD      angle
TAN
ST      tangent
ATAN
ST      wynik
```

ROL

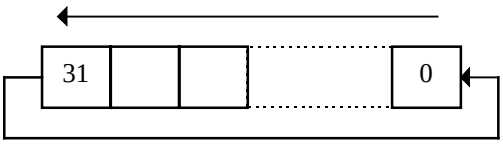


Argumenty:

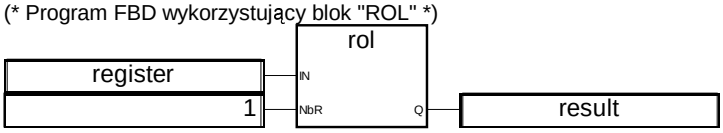
IN	INT	dowolna wartość analogowa typu integer
NbR	INT	liczba przesunięć (w przedziale [1..31])
Q	INT	wartość po przesunięciu w lewo bez zmiany, jeśli NbR <= 0

Opis:

Przesuwa cyklicznie bity wartości integer w lewo. Operacja wykonywana jest na 32 bitach:

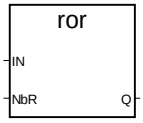


(* Program FBD wykorzystujący blok "ROL" *)




```
(* Reprezentacja ST: *)
wynik := ROL (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* wynik    = 2#1001_1010_0110_1010*)

(* Reprezentacja IL: *)
LD          register
ROL         1
ST          wynik
```

ROR

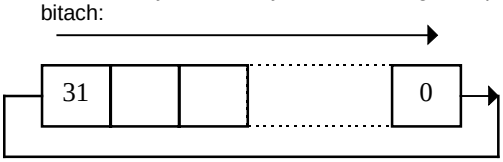


Argumenty:

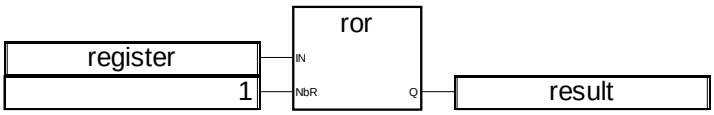
IN	INT	dowolna wartość analogowa typu integer
NbR	INT	liczba przesunięć (w przedziale [1..31])
Q	INT	wartość po przesunięciu w prawo bez zmiany, jeśli NbR <= 0

Opis:

Przesuwa cyklicznie bity wartości integer w prawo. Operacja wykonywana jest na 32 bitach:

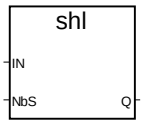


(* Program FBD wykorzystujący blok "ROR" *)




```
(* Reprezentacja ST: *)
wynik := ROR (register, 1);
(* register = 2#0100_1101_0011_0101 *)
(* wynik    = 2#1010_0110_1001_1010 *)

(* Reprezentacja IL: *)
LD          register
ROR         1
ST          wynik
```

SHL

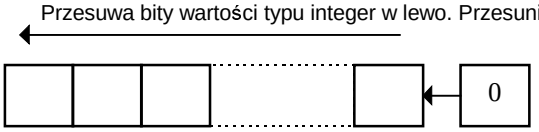


Argumenty:

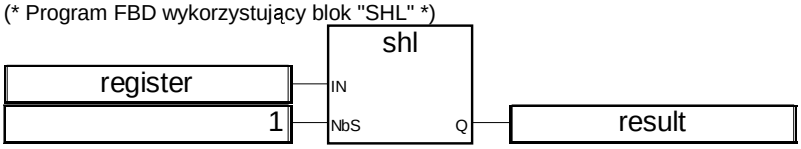
IN	INT	dowolna wartość analogowa integer
NbS	INT	liczba przesunięć o 1 bit (w przedziale [1..31])
Q	INT	wartość po przesunięciu w lewo bez zmiany, jeśli NbS <= 0 0 jest wprowadzane na miejsce najmniej znaczącego bitu

Opis:

Przesuwa bity wartości typu integer w lewo. Przesunięcie wykonywane jest a 32 bitach:



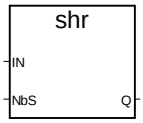
(* Program FBD wykorzystujący blok "SHL" *)




```
(* Reprezentacja ST: *)
wynik := SHL (register,1);
(* register = 2#0100_1101_0011_0101 *)
(* wynik    = 2#1001_1010_0110_1010 *)

(* Reprezentacja IL: *)
```

```
LD      register
SHL     1
ST      wynik
```

SHR

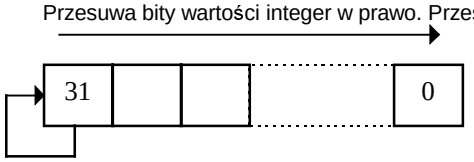


Argumenty:

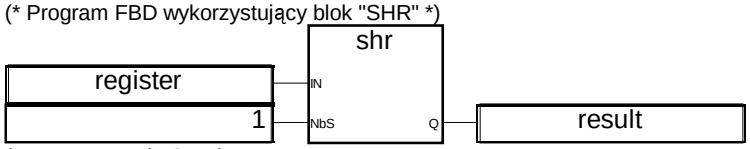
IN	INT	dowolna wartość analogowa integer
NbS	INT	liczba przesunięć o 1 bit (w przedziale [1..31])
Q	INT	wartość po przesunięciu w prawo bez zmiany, jeśli NbS <= 0 przy każdym przesunięciu najbardziej znaczący bit jest kopiowany

Opis:

Przesuwa bity wartości integer w prawo. Przesunięcie wykonywane jest na 32 bitach:

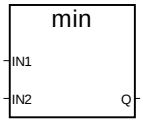


(* Program FBD wykorzystujący blok "SHR" *)




```
(* Reprezentacja ST: *)
wynik := SHR (register,1);
(* register = 2#1100_1101_0011_0101 *)
(* wynik    = 2#1110_0110_1001_1010 *)

(* Reprezentacja IL: *)
LD      register
SHR     1
ST      wynik
```

MIN



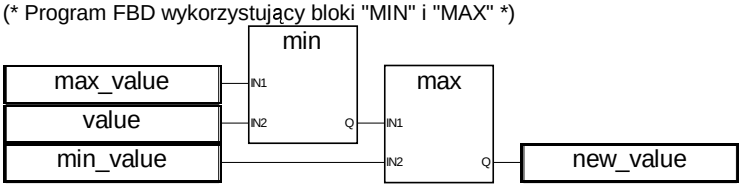
Argumenty:

IN1	INT	dowolna wartość analogowa typu integer ze znakiem
IN2	INT	(nie może być typu REAL)
Q	INT	minimum obu wartości wejściowych

Opis:

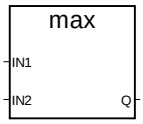
Podaje wartość minimalną dwóch wartości typu integer .

(* Program FBD wykorzystujący bloki "MIN" i "MAX" *)




```
(* Reprezentacja ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* ogranicza wartość ustawioną [min_value..max_value] *)

(* Reprezentacja IL: *)
LD      max_value
MIN     value
MAX     min_value
ST      new_value
```

MAX



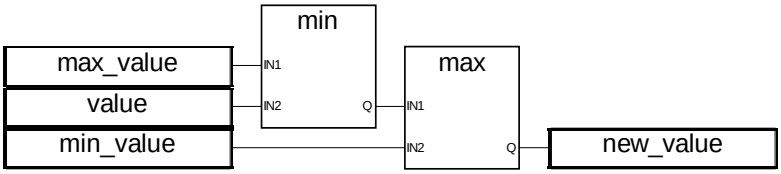
Argumenty:

IN1	INT	dowolna wartość analogowa typu integer ze znakiem
IN2	INT	(nie może być typu REAL)
Q	INT	maksimum obu wartości wejściowych

Opis:

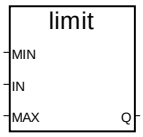
Podaje maksymalną wartość z dwóch wartości typu integer.

(* Program FBD wykorzystujący bloki "MIN" i "MAX" *)




```
(* Reprezentacja ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* ogranicza wartość ustawioną [min_value..max_value] *)

(* Reprezentacja IL: *)
LD      max_value
MIN     value
MAX     min_value
ST      new_value
```

LIMIT



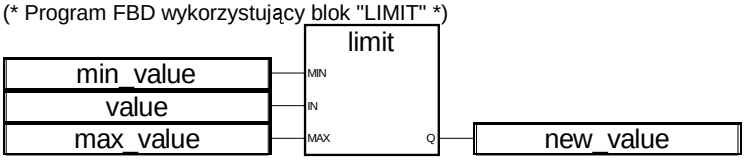
Argumenty:

MIN	INT	minimalna dopuszczalna wartość
IN	INT	dowolna wartość analogowa typu integer ze znakiem
MAX	INT	maksymalna dopuszczalna wartość
Q	INT	wartość wejściowa ograniczona do dopuszczalnego zakresu

Opis:

Ogranicza wartość typu integer do danego przedziału. Wartość jest zachowana, jeśli znajduje się ona pomiędzy minimum a maksimum; wartość jest zmieniana na wartość maksymalną, jeśli jest od niej większa; wartość jest zmieniana na minimum, jeśli jest mniejsza od niego mniejsza.

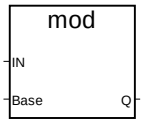
(* Program FBD wykorzystujący blok "LIMIT" *)




```
(* Reprezentacja ST: *)
new_value := LIMIT (min_value, value, max_value);
(*ogranicza wartość ustawioną [min_value..max_value] *)

(* Reprezentacja IL: *)
```

```
LD      min_value
LIMIT   value, max_value
ST      new_value
```

MOD



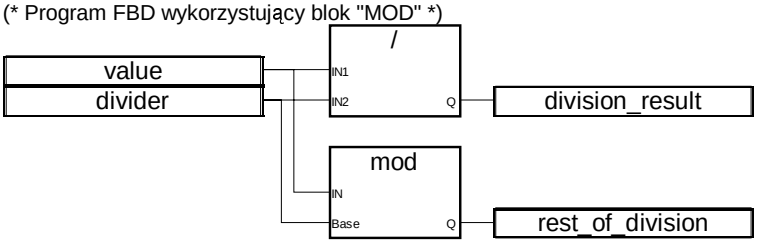
Argumenty:

IN	INT	dowolna wartość analogowa typu integer ze znakiem
Base	INT	musi być większy od zera
Q	INT	obliczenie modulo (wejściem jest podstawa MOD) zwraca -1 jeśli podstawa <= 0

Opis:

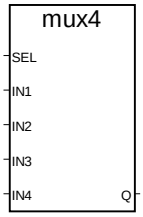
Oblicza modulo wartości typu integer (reszta z dzielenia).

(* Program FBD wykorzystujący blok "MOD" *)




```
(* Reprezentacja ST: *)
division_wynik := (value / divider); (* dzielenie typu integer *)
rest_of_division := MOD (value, divider); (* reszta z dzielenia *)

(* Reprezentacja IL: *)
LD      value
DIV     divider
ST      division_wynik
LD      value
MOD     divider
ST      rest_of_division
```

MUX4



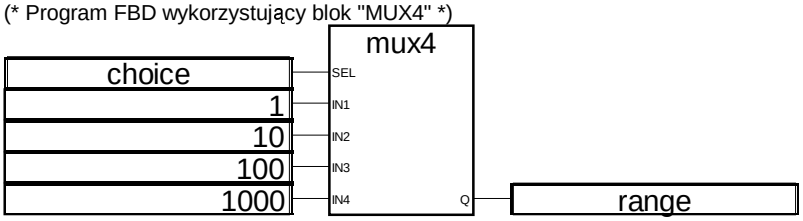
Argumenty:

SEL	INT	wartość integer selektora (musi być w przedziale [0..3])
IN1..IN4	INT	dowolne wartości analogowe typu integer
Q	INT	= value1 jeśli SEL = 0 = value2 jeśli SEL = 1 = value3 jeśli SEL = 2 = value4 jeśli SEL = 3 = 0 dla wszystkich pozostałych wartości selektora

Opis:

Multiplekser o 4 wejściach: wybiera jedną wartość spośród 4 wartości integer.

(* Program FBD wykorzystujący blok "MUX4" *)

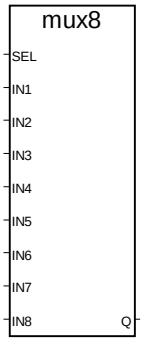


(* Reprezentacja ST: *)
range := MUX4 (choice, 1, 10, 100, 1000);
(* wybór spomiędzy 4 wstępnie zdefiniowanych zakresów, np. jeśli wybrane zostanie 1, to zakresem jest 10 *)

(* Reprezentacja IL: *)

```
LD      choice
MUX4    1,10,100,1000
ST      range
```

MUX8



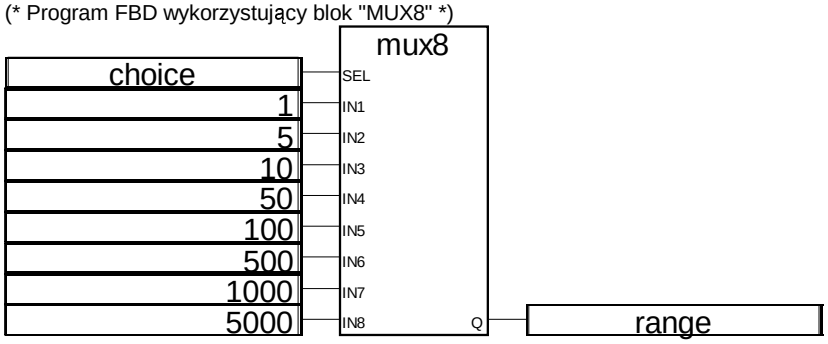
Argumenty:

SEL	INT	wartość integer selektora (musi być w przedziale [0..7])
IN1..IN8	INT	dowolne wartości analogowe typu integer
Q	INT	= value1 jeśli selektor = 0 = value2 jeśli selektor = 1 ... = value8 jeśli selektor = 7 = 0 dla wszystkich pozostałych wartości selektora

Opis:

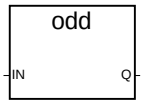
Multiplekser o 8 wejściach: wybiera wartość spośród 8 wartości integer.

(* Program FBD wykorzystujący blok "MUX8" *)




```
(* Reprezentacja ST: *)
range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);
(*wybór spomiędzy 8 wstępnie zdefiniowanych zakresów, np. jeśli wybrane zostanie 3, to
zakresem jest 50 *)

(* Reprezentacja IL: *)
LD      choice
MUX8    1,5,10,50,100,500,1000,5000
ST      range
```

ODD



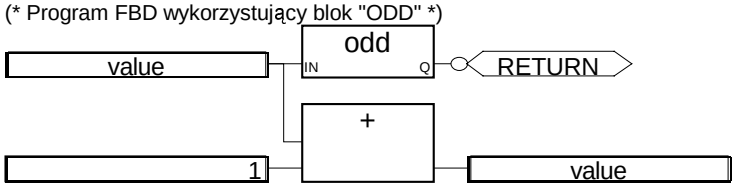
Argumenty:

IN	INT	wartość analogowa typu integer ze znakiem
Q	BOO	PRAWDA, jeśli wartość wejściowa jest nieparzysta FAŁSZ, jeśli wartość wejściowa jest parzysta

Opis:

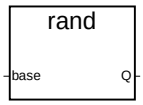
Testuje nieparzystość wartości integer: wynik jest wartością binarną PRAWDA, gdy liczbą jest nieparzysta.

(* Program FBD wykorzystujący blok "ODD" *)




```
(* Reprezentacja ST: *)
If Not (ODD (value)) Then Return; End_if;
value := value + 1;
(* sprawia, że wartość jest zawsze parzysta *)

(* Reprezentacja IL: *)
LD          value
ODD
RETNC
LD          value
ADD         1
ST          value
```

RAND



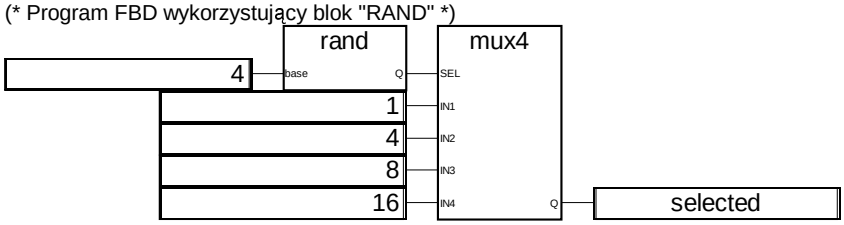
Argumenty:

base	INT	definiuje górny zakres przedziału
Q	INT	ustawia wartość losową [0..base-1]

Opis:

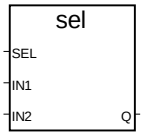
Podaje losową wartość integer z przedziału [0,base-1].

(* Program FBD wykorzystujący blok "RAND" *)




```
(* Reprezentacja ST: *)
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );
(*
losowy wybór 1 z 4 wstępnie zdefiniowanych wartości
wartość wyniku wywołania RAND ustawiana jest w przedziale [0..3],
więc 'wybrany' wynik MUX4, uzyskuje 'losowo' wartość
1 jeśli 0 jest wynikiem RAND,
lub 4 jeśli 1 jest wynikiem RAND,
lub 8 jeśli 2 jest wynikiem RAND,
lub 16 jeśli 3 jest wynikiem RAND,
*)

(* Reprezentacja IL: *)
LD          4
RAND
MUX4        1,4,8,16
ST          selected
```

SEL



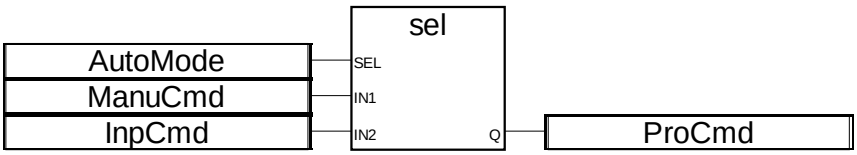
Argumenty:

SEL	BOO	wskazuje wybraną wartość
IN1, IN2	INT	dowolne wartości analogowe typu integer
Q	INT	= wartość 1, jeśli SEL = FAŁSZ = wartość 2, jeśli SEL = PRAWDA

Opis:

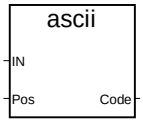
Selektor binarny: wybiera jedną wartość z 2 wartości integer.

(* Program FBD wykorzystujący blok "SEL" *)




```
(* Reprezentacja ST: *)
ProCmd := SEL (AutoMode, ManuCmd, InpCmd);
(* wybór komendy procesu *)

(* Reprezentacja IL: *)
LD          AutoMode
SEL         ManuCmd,InpCmd
ST          ProCmd
```

ASCII



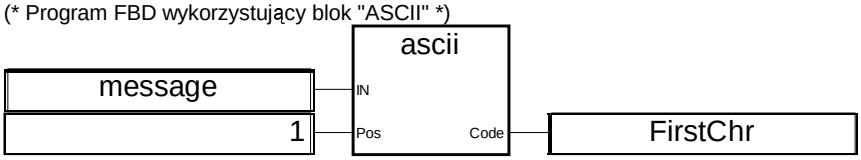
Argumenty:

IN	MSG	dowolny łańcuch nie-pusty
Pos	INT	pozycja wybranego znaku w przedziale [1.. len] (len jest długością komunikatu IN)
Code	INT	kod wybranego znaku (w przedziale [0 .. 255]) daje w rezultacie 0, jeśli Pos znajduje się poza łańcuchem

Opis:

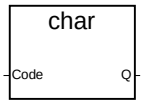
Podaje kod ASCII jednego znaku w łańcuchu znakowym.

```
(* Program FBD wykorzystujący blok "ASCII" *)
```




```
(* Reprezentacja ST: *)
FirstChr := ASCII (message, 1);
(* FirstChr to kod Ascii pierwszego znaku w łańcuchu znaków *)

(* Reprezentacja IL: *)
LD          message
ASCII       1
ST          FirstChr
```

CHAR



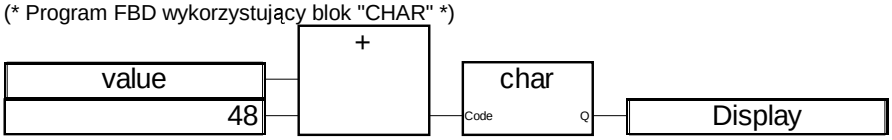
Argumenty:

Code	INT	kod z przedziału [0 .. 255]
Q	MSG	łańcuch jednoznakowy znak posiada kod ASCII podany na wejściu Code (ASCII kod jest stosowany modulo 256)

Opis:

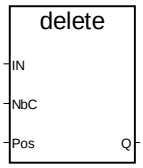
Podaje łańcuch jednoznakowy komunikatu z danego kodu ASCII.

(* Program FBD wykorzystujący blok "CHAR" *)




```
(* Reprezentacja ST: *)
Display := CHAR ( value + 48 );
(* wartość jest z przedziału [0..9] *)
(* 48 to kod ascii znaku '0' *)
(* wynikiem jest łańcuch jednoznakowy od '0' do '9' *)

(* Reprezentacja IL: *)
LD      value
ADD     48
CHAR
ST      Display
```

DELETE



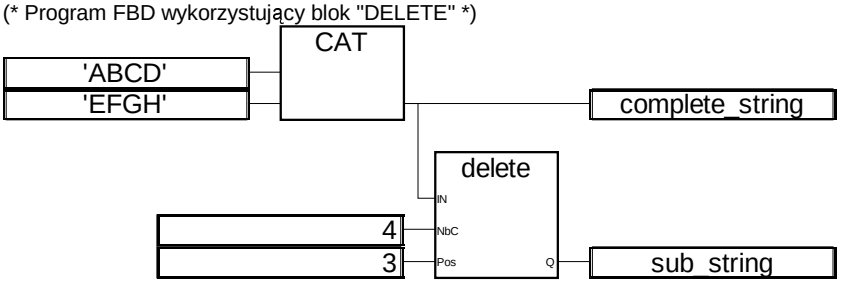
Argumenty:

IN	MSG	dowolny nie-pusty komunikat
NbC	INT	liczba znaków do usunięcia
Pos	INT	pozycja pierwszego usuwanego znaku (pierwszy znak łańcucha zajmuje pozycję 1)
Q	MSG	zmodyfikowany łańcuch łańcuch pusty, jeśli Pos < 1

łańcuch pierwotny, jeśli Pos > długość łańcucha IN
łańcuch pierwotny, jeśli NbC <= 0

Opis:
Usuwa część znaków z komunikatu.

(* Program FBD wykorzystujący blok "DELETE" *)

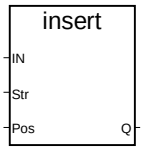


(* Reprezentacja ST: *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string to 'ABCDEFGH' *)
sub_string := DELETE (complete_string, 4, 3); (* sub_string to 'ABGH' *)

(* Reprezentacja IL: *)

LD	'ABCD'
ADD	'EFGH'
ST	complete_string
DELETE	4,3
ST	sub_string

INSERT

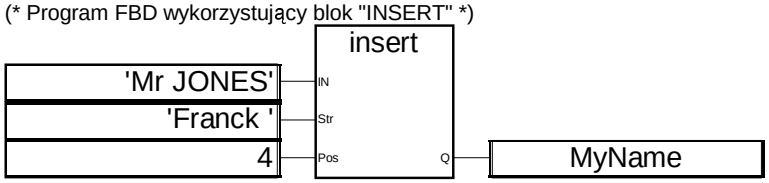


Argumenty:

IN	MSG	łańcuch pierwotny
Str	MSG	łańcuch do wstawienia
Pos	INT	pozycja wstawiania wstawianie jest dokonywane przed daną pozycją (pierwszą prawidłową pozycją jest 1)
Q	MSG	zmodyfikowany łańcuch łańcuch pusty, jeśli Pos <= 0 łączenie obu ciągów, jeśli Pos jest większa od długości łańcucha IN

Opis:
Wstawia podłańcuch do komunikatu we wskazanym miejscu.

(* Program FBD wykorzystujący blok "INSERT" *)

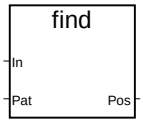


(* Reprezentacja ST: *)
MyName := INSERT ('Mr JONES', 'Frank ', 4);
(* MyName to 'Mr Frank JONES' *)

(* Reprezentacja IL: *)

LD	'Mr JONES'
INSERT	'Frank ',4
ST	MyName

FIND



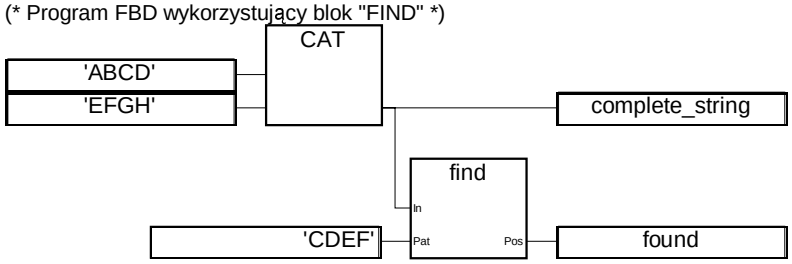
Argumenty:

In	MSG	dowolny łańcuch znaków komunikatu
Pat	MSG	dowolny łańcuch nie-pusty (Wzór)
Pos	INT	= 0 jeśli podłańcuch Pat nie zostanie znaleziony = pozycja pierwszego znaku pierwszego wystąpienia podłańcucha Pat (pierwszą pozycją jest 1) funkcja ta **uwzględnia wielkość liter**

Opis:

Znajduje podłańcuch w łańcuchu znakowym komunikatu. Podaje pozycję podłańcucha w danym łańcuchu.

(* Program FBD wykorzystujący blok "FIND" *)



(* Reprezentacja ST: *)

```
complete_string := 'ABCD' + 'EFGH'; (* complete_string to 'ABCDEFGH' *)
found := FIND (complete_string, 'CDEF'); (* found to 3 *)

(* Reprezentacja IL: *)
LD          'ABCD'
ADD         'EFGH'
ST          complete_string
FIND        'CDEF'
ST          found
```

MLEN

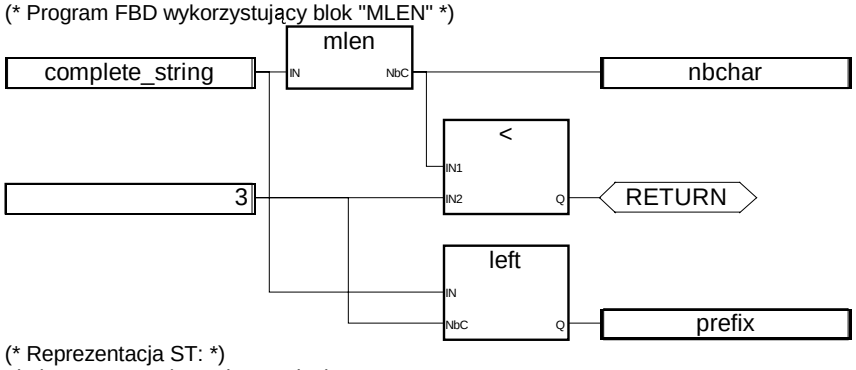
Argumenty:

IN	MSG	dowolny łańcuch znaków komunikatu
NbC	INT	liczba znaków w łańcuchu IN

Opis:

Podaje długość komunikatu.

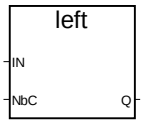
(* Program FBD wykorzystujący blok "MLEN" *)




```
(* Reprezentacja ST: *)
nbchar := MLEN (complete_string);
If (nbchar < 3) Then Return; End_if;
prefix := LEFT (complete_string, 3);
(* program ten wyciąga 3 znaki z lewej strony łańcucha i wstawia wynik w prefiksie zmiennej
komunikatu
jeśli długość łańcucha jest mniejsza niż 3 znaki, to nic się nie dzieje *)

(* Reprezentacja IL: *)
LD          complete_string
MLEN
ST          nbchar
```

```
LT          3
RETC
LD          complete_string
LEFT        3
ST          prefix
```

LEFT



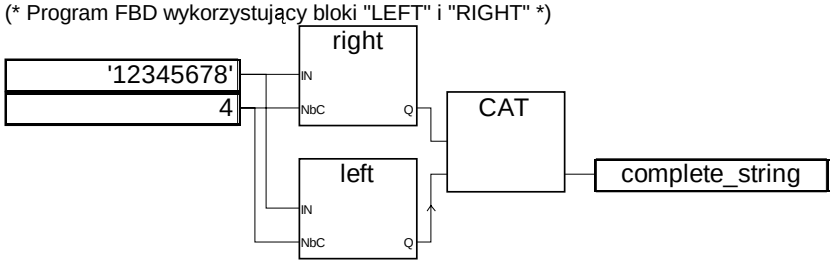
Argumenty:

IN	MSG	dowolny łańcuch nie-pusty
NbC	INT	liczba znaków do wyciągnięcia nie może być większa niż długość łańcucha IN
Q	MSG	lewa część łańcucha IN (jego długość = NbC) łańcuch pusty, jeśli NbC <= 0 kompletny łańcuch, IN jeśli NbC >= długość łańcucha IN

Opis:

Wyciąga lewą część łańcucha komunikatu. Podana jest liczba znaków do wyciągnięcia.

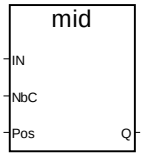
(* Program FBD wykorzystujący bloki "LEFT" i "RIGHT" *)




```
(* Reprezentacja ST: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string to '56781234'
wartością wynikającą z wywołania RIGHT jest '5678'
wartością wynikającą z wywołania LEFT jest '1234'
*)

(* Reprezentacja IL: Najpierw wykonywane jest wywołanie LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string  (* wynik pośredni *)
LD          '12345678'
RIGHT       4
ADD         sub_string
```

```
ST          complete_string
```

MID



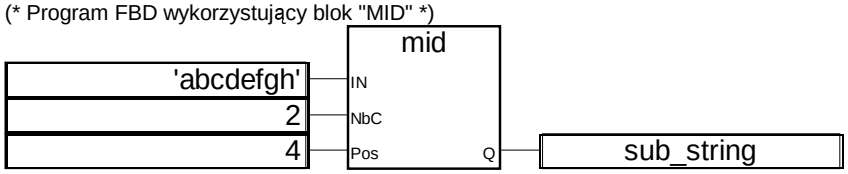
Argumenty:

IN	MSG	dowolny łańcuch nie-pusty
NbC	INT	Liczba znaków do wydobycia nie może być większa niż długość łańcucha IN
Pos	INT	pozycja pierwszego znaku podłańcucha jest wskazywana przez Pos (pierwszą prawidłową pozycją jest 1)
Q	MSG	środkowa część łańcucha (jego długość = NbC) łańcuch pusty, jeśli parametry nie są prawidłowe

Opis:

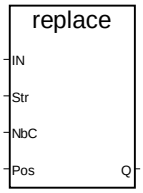
Wyciąga część znaków z komunikatu. Podana jest ilość znaków i pozycja pierwszego znaku do wyciągnięcia.

(* Program FBD wykorzystujący blok "MID" *)




```
(* Reprezentacja ST: *)
sub_string := MID ('abcdefgh', 2, 4);
(* sub_string to 'de' *)

(* Reprezentacja IL: *)
LD          'abcdefgh'
MID         2,4
ST          sub_string
```

REPLACE



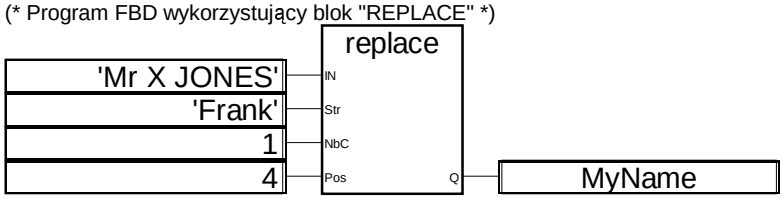
Argumenty:

IN	MSG	dowolny łańcuch
Str	MSG	łańcuch do wstawienia (zastępujący znaki NbC)
NbC	INT	ilość znaków do usunięcia
Pos	INT	pozycja pierwszego zmodyfikowanego znaku (pierwszą prawidłową pozycją jest 1)
Q	MSG	zmodyfikowany łańcuch: - znaki NbC w pozycji Pos są usuwane - następnie w tej pozycji jest wstawiany podłańcuch Str zwraca łańcuch pusty, jeśli Pos <= 0 zwraca ciągi połączone (IN+Str), jeśli wartość Pos jest większa od długości łańcucha IN zwraca łańcuch pierwotny IN, jeśli NbC <= 0

Opis:

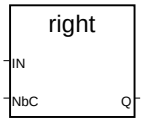
Zastępuje część łańcucha komunikatu nowym łańcuchem znaków.

(* Program FBD wykorzystujący blok "REPLACE" *)




```
(* Reprezentacja ST: *)
MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);
(* MyName ma postać 'Mr Frank JONES' *)

(* Reprezentacja IL: *)
LD          'Mr X JONES'
REPLACE     'Frank',1,4
ST          MyName
```

RIGHT



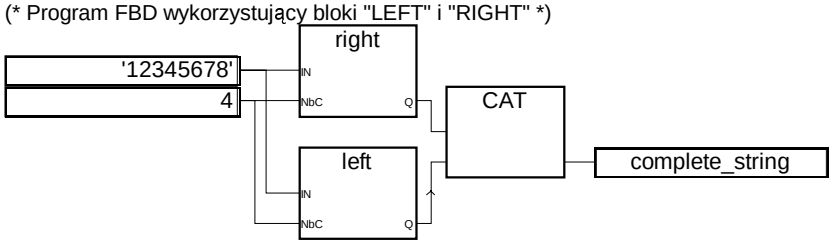
Argumenty:

IN	MSG	dowolny łańcuch nie-pusty
NbC	INT	nie może być większy niż długość łańcucha IN
Q	MSG	prawa część łańcucha (długość = NbC) łańcuch pusty, jeśli NbC <= 0 kompletny łańcuch, jeśli NbC >= długość łańcucha

Opis:

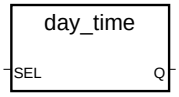
Wyciąga prawą część łańcucha znaków komunikatu. Podana jest ilość znaków do wyciągnięcia.

(* Program FBD wykorzystujący bloki "LEFT" i "RIGHT" *)




```
(* Reprezentacja ST: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string to '56781234'
wartością otrzymywaną z wywołania RIGHT jest '5678'
wartością otrzymywaną z wywołania LEFT jest '1234'
*)

(* Reprezentacja IL: Pierwszym wykonywanym wywołaniem jest LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string  (* wynik pośredni *)
LD          '12345678'
RIGHT       4
ADD         sub_string
ST          complete_string
```

DAY_TIME



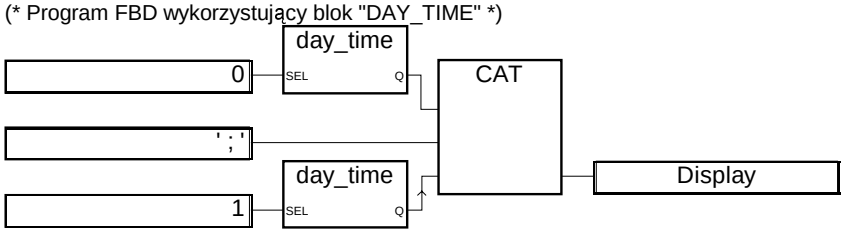
Argumenty:

SEL	INT	wybór wyjścia 0= pobierz aktualną datę 1= pobierz aktualny czas 2= pobierz dzień tygodnia
Q	MSG	czas/data wyrażona łańcuchem znaków 'RRRR/MM/DD' jeśli SEL = 0 'GG:MM:SS' jeśli SEL = 1 nazwa dnia, jeśli SEL = 2 (np. 'Poniedziałek')

Opis:

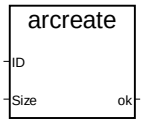
Podaje datę i czas dobowy w formie komunikatu.

(* Program FBD wykorzystujący blok "DAY_TIME" *)




```
(* Reprezentacja ST: *)
Display := Day_Time (0) + ' ; ' + Day_Time (1);
(* Tekst jest wyświetlany w formacie: 'RRRR/MM/DD ; GG:MM:SS' *)

(* Reprezentacja IL: Pierwszym wykonywanym wywołanie, jest day_time(1) *)
LD          1
DAY_TIME
ST          hour_str    (* wynik pośredni *)
LD          0
DAY_TIME
ADD         ' ; '
ADD         hour_str
ST          Display
```

ARCREATE



Argumenty:

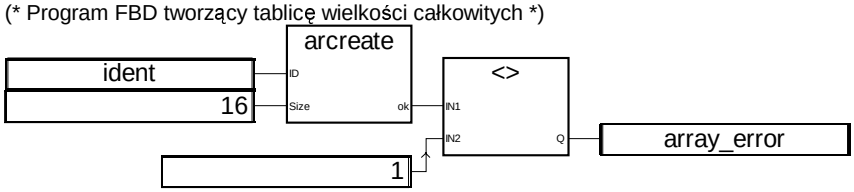
ID	INT	identyfikator tablicy (musi być z przedziału [0..15])
Size	INT	ilość elementów tablicy
ok	INT	status wykonania: **1** = jeśli z powodzeniem stworzono tablicę

2 = nieprawidłowy identyfikator tablicy lub już utworzona tablica
3 = nieprawidłowy rozmiar
4 = niewystarczająca ilość pamięci

Opis:
Tworzenie tablicy wartości typu integer.

Uwaga: W aplikacji jest najwyżej **16** tablic. Tablice zawierają wartości **analogowe** typu **integer** . Po wykonywaniu dynamicznej alokacji pamięci, funkcja ta może spowodować błąd systemowy, jeśli wielkość tablicy jest zbyt bliska ilości dostępnej pamięci.

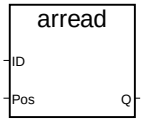
(* Program FBD tworzący tablicę wielkości całkowitych *)



(* Reprezentacja ST: *)
array_error := (ARCREATE (ident, 16) <> 1));

(* Reprezentacja IL: *)
LD ident
ARCREATE 16
NE 1
ST array_error

ARREAD

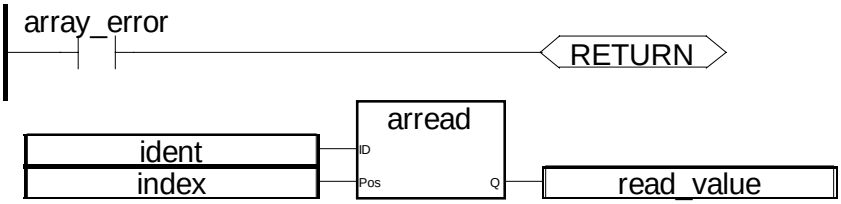


Argumenty:

ID	INT	identyfikator tablicy (musi być w przedziale [0..15])
Pos	INT	pozycja elementu w tablicy musi być w przedziale [0 .. rozmiar-1]
value	INT	wartość odczytywanego elementu 0 jeśli argumenty nie są prawidłowe

Opis:
Odczytuje element z tablicy typu integer .

(* Program FBD wykorzystujący bloki zarządzania tablicą *)




```
(* Reprezentacja ST: *)
If (array_error) Then Return; End_if;
read_value := ARREAD (ident, index);
(* array_error pochodzi z wywołania ARCREATE *)

(* Reprezentacja IL: *)
LD          array_error
RETC
LD          ident
ARREAD      index
ST          read_value
```

ARWRITE

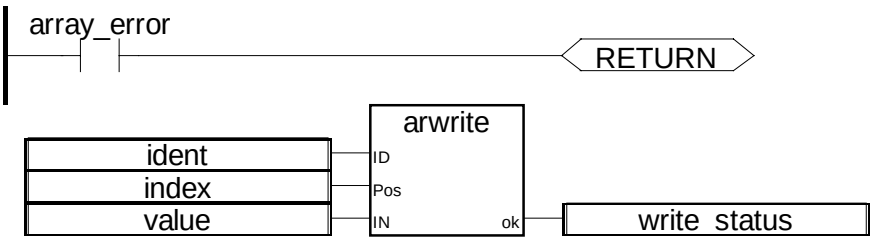
Argumenty:

ID	INT	identyfikator tablicy (musi być w przedziale [0..15])
Pos	INT	pozycja elementu tablicy musi być w przedziale [0 .. rozmiar-1]
IN	INT	nowa wartość elementu
ok	INT	status wykonania **1** = zapis wykonany pomyślnie **2** = nieprawidłowy identyfikator tablicy **3** = nieprawidłowy indeks

Opis:

Zachowuje (zapisuje) wartość w tablicy wartości typu integer.

(* Program FBD wykorzystujący bloki zarządzania tablicą *)




```
(* Reprezentacja ST: *)
If (array_error) Then Return; End_if;
write_status := ARWRITE (Ident, Index, value);
(* array_error pochodzi z wywołania ARCREATE *)

(* Reprezentacja IL: *)
LD          array_error
RETC
LD          ident
ARWRITE     index,value
ST          write_status
```

F_ROPEN

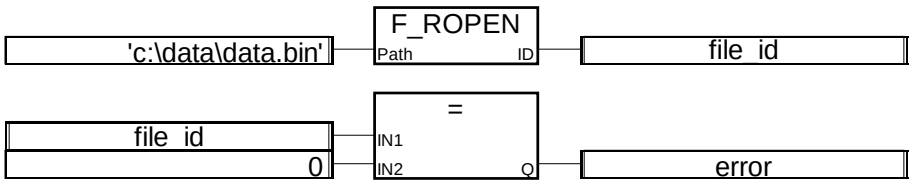
Argumenty:

Path	MSG	nazwa pliku Może zawierać ścieżkę dostępu do pliku z użyciem symbolu \ lub / do określenia katalogu. Dla ułatwienia przenoszalności aplikacji przyjmuje się, że znaki / i \ są równoważne.
ID	INT	numer pliku 0 jeśli, wystąpi błąd: plik nie istnieje.

Opis:

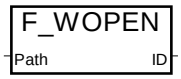
Otwiera plik binarny w trybie do odczytu. Do użycia z FX_READ i F_CLOSE.
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

(* Program FBD wykorzystujący bloki zarządzania plikiem *)




```
(* Reprezentacja ST: *)
file_id := F_ROPEN('c:\data \data.bin');
error := (file_id=0);

(* Reprezentacja IL: *)
LD          'c:\data\data.bin'
F_ROPEN
ST          file_id
EQ          0
ST          error
```

F_WOPEN



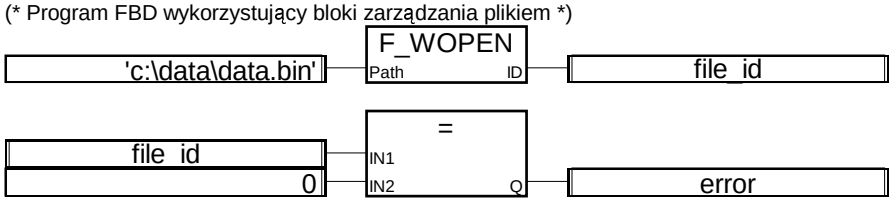
Argumenty:

Path	MSG	nazwa pliku Może zawierać ścieżkę dostępu do pliku z użyciem symbolu \ lub / do określenia katalogu. Dla ułatwienia przenoszalności aplikacji przyjmuje się, że / i \ są równoważne.
ID	INT	numer pliku 0 jeśli wystąpi błąd. Jeśli plik już istnieje, zostaje zapisany nowymi wartościami.

Opis:

Otwiera plik binarny w trybie do zapisu. Do użycia z FX_READ i F_CLOSE.
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

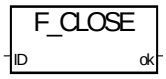
(* Program FBD wykorzystujący bloki zarządzania plikiem *)




```
(* Reprezentacja ST: *)
file_id := F_WOPEN('c:\data \data.bin');
```

```
error := (file_id=0);

(* Reprezentacja IL: *)
LD              'c:\data\data.bin'
F_WOPEN
ST              file_id
EQ              0
ST              error
```

F_CLOSE



Argumenty:

ID	INT	Numer pliku: zwracany przez F_ROPEN lub F_WOPEN.
ok	BOO	zwracany status PRAWDA, jeśli zamknięcie pliku jest prawidłowe FAŁSZ, jeśli wystąpił błąd

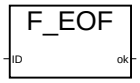
Opis:

Zamyka plik binarny otwarty przy pomocy funkcji F_ROPEN lub F_WOPEN.
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

(* Program FBD wykorzystujący bloki zarządzania plikiem *)

```
(* Reprezentacja ST: *)
file_id := F_ROPEN('data.bin');
ok := F_CLOSE(file_id);

(* Reprezentacja IL: *)
LD              'data.bin'
F_ROPEN
ST              file_id
F_CLOSE                 (* file_id znajduje się już w wyniku bieżącym IL *)
ST              ok
```

F_EOF



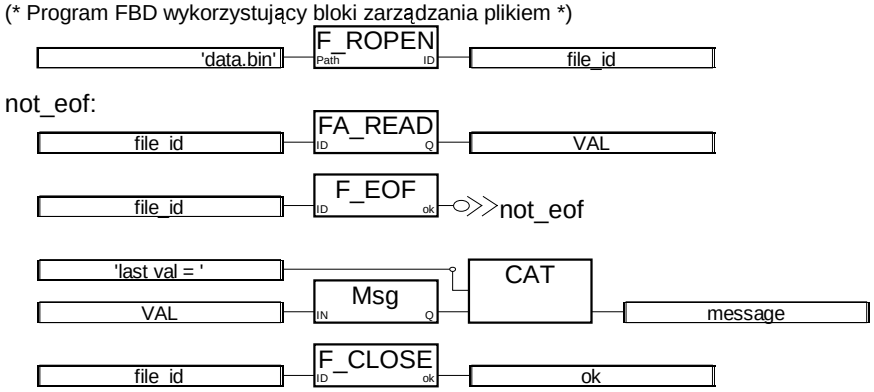
Argumenty:

ID	INT	numer pliku: zwracany przez F_ROPEN lub F_WOPEN.
ok	BOO	wskaźnik końca pliku PRAWDA, jeśli w ostatnim wywołaniu procedury odczytu lub zapisu osiągnięty został koniec pliku. W przypadku FM_READ ostatni komunikat odczytany z pliku może nie być prawidłowy, jeśli ostatni znak nie jest znakiem końca łańcucha.

Opis:

Sprawdza, czy osiągnięty został koniec pliku.
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

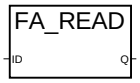
(* Program FBD wykorzystujący bloki zarządzania plikiem *)




```
(* Reprezentacja ST: *)
file_id := F_ROPEN('data.bin');
WHILE not(F_EOF(file_id))
        VAL := FA_READ(file_id);
END_WHILE;
MESSAGE := 'last val = ' + msg(VAL);
ok := F_CLOSE(file_id);

(* Reprezentacja IL: *)
            LD      'data.bin'
            F_ROPEN
            ST      file_id
            LD      file_id
            F_EOF
            JMPC    END_OF_FILE
NOT_EOF:    LD      file_id
```

```
                FA_READ
                ST          VAL
                LD          file_id
                F_EOF
                JMPNC       NOT_EOF  (* jeśli nie jest to koniec pliku, odczytuj dalej *)
END_OF_FILE:    LD          VAL
                MSG
                ST          val_msg     (* konwersja VAL na komunikat *)
                LD          'last val = '
                ADD         val_msg
                ST          MESSAGE
                LD          file_id
                F_CLOSE
                ST          ok
```

FA_READ



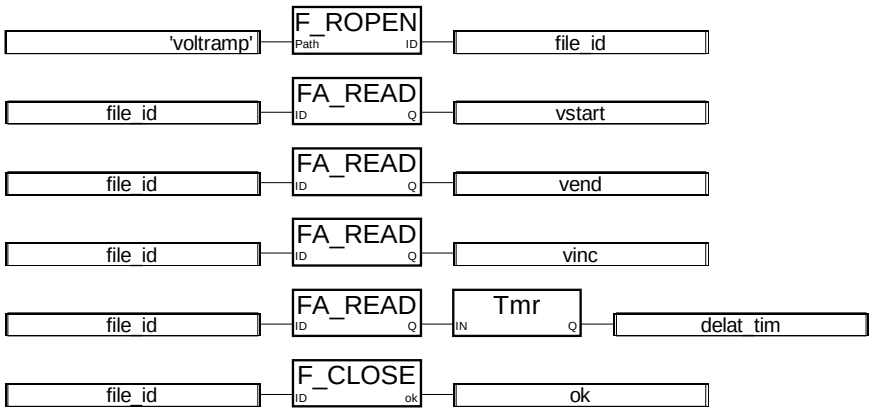
Argumenty:

ID	INT	numer pliku: zwracany przez F_ROPEN.
Q	INT	wartość analogowa typu integer odczytywana z pliku

Opis:

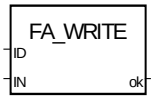
Odczytuje zmienne ANALOGOWE z pliku binarnego. Do użycia z F_ROPEN i F_CLOSE.
Procedura ta uzyskuje dostęp sekwencyjny do pliku z poprzedniej pozycji.
Pierwsze wywołanie po F_ROPEN odczytuje pierwsze 4 bajty pliku,
każde wywołanie umieszcza (na stosie) wskaźnik odczytu.
Aby sprawdzić, czy osiągnięty został koniec pliku, należy zastosować F_EOF.
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

(* Program FBD wykorzystujący bloki zarządzania plikiem *)




```
(* Reprezentacja ST: *)
file_id := F_ROPEN('voltramp.bin');
vstart := FA_READ(file_id);
vend := FA_READ(file_id);
vinc := FA_READ(file_id);
delta_tim := tmr(FA_READ(file_id));
ok := F_CLOSE(file_id);

(* Reprezentacja IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
        FA_READ             (* odczyt vstart *)
        ST      vstart
        LD      file_id
        FA_READ             (* odczyt vend *)
        ST      vend
        LD      file_id
        FA_READ             (* odczyt vinc *)
        ST      vinc
        LD      file_id
        FA_READ             (* odczyt delta_tim *)
        TMR                 (* konwersja do typu timer *)
        ST      delta_tim
        LD      file_id
        F_CLOSE
        ST      ok
```

FA_WRITE



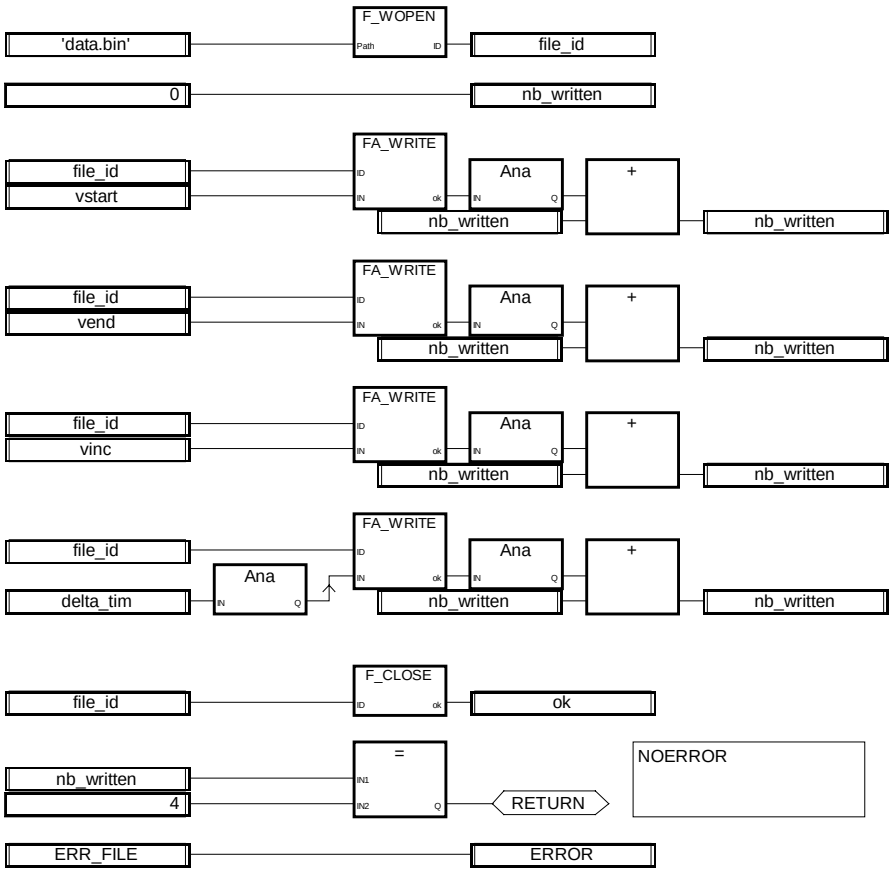
Argumenty:

ID	INT	numer pliku: zwracany przez F_WOPEN.
IN	INT	wartość analogowa typu integer do zapisania w pliku
OK	BOO	status wykonania: PRAWDA, jeśli zapis prawidłowy

Opis:

Zapisuje zmienne ANALOGOWE z pliku binarnego.
Procedura ta uzyskuje dostęp sekwencyjny do pliku z poprzedniej pozycji.
Pierwsze wywołanie po F_WOPEN zapisuje pierwsze 4 bajty pliku,
każde wywołanie umieszcza (na stosie) wskaźnik zapisu.
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

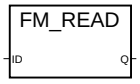
(* Program FBD *)




```
(* Reprezentacja ST: *)
file_id := F_WOPEN('voltramp.bin');
nb_written  := 0;
nb_written := nb_written + ana(FA_WRITE(file_id,vstart));
nb_written := nb_written + ana(FA_WRITE(file_id,vend));
nb_written := nb_written + ana(FA_WRITE(file_id,vinc));
nb_written := nb_written + ana(FA_WRITE(file_id,ana(delta_tim)));
ok := F_CLOSE(file_id);
IF ( nb_written <> 4) THEN
        ERROR := ERR_FILE;
END_IF;

(* Reprezentacja IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
```

```
LD          0
ST          nb_written
LD          file_id       (* zapis vstart *)
FA_WRITE    vstart
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (* zapis vend *)
FA_WRITE    vend
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (* zapis vinc *)
FA_WRITE    vinc
ANA
ADD         nb_written
LD                        (* zapis delta_tim *)
ANA                       (* konwertuj wartości całkowitej *)
ST          ana_delta_tim
LD          file_id
FA_WRITE    ana_delta_tim
ANA
ADD         nb_written
ST          nb_written
F_CLOSE
ST          ok
LD          nb_written
EQ          4
RETC                      (* zwrot jeśli równe 4 *)
LD          ERR_FILE      (* w przeciwny wypadku - błąd *)
ST          ERROR
```

FM_READ



Argumenty:

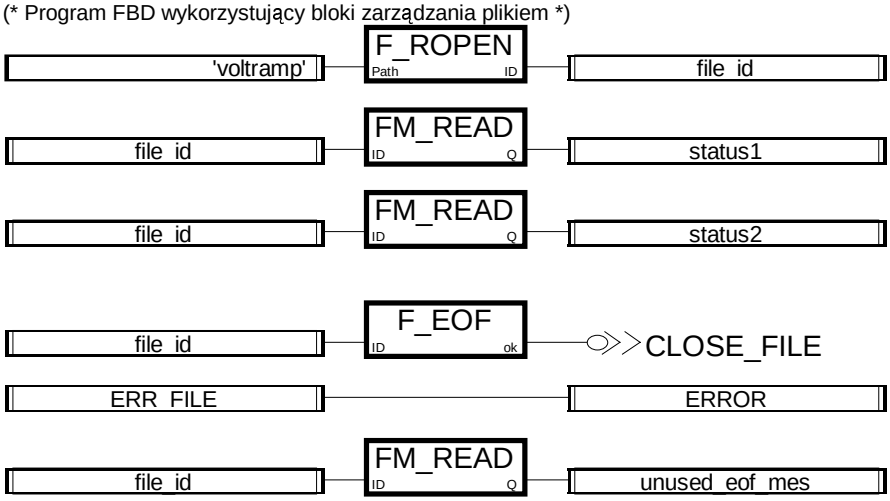
ID	INT	numer pliku: zwracany przez F_ROPEN.
Q	MSG	wartość komunikatu odczytywana z pliku

Opis:

Odczytuje zmienne typu KOMUNIKAT z pliku binarnego.
Do użycia z F_ROPEN i F_CLOSE.
Procedura ta uzyskuje dostęp sekwencyjny do pliku z poprzedniej pozycji.
Pierwsze wywołanie po F_ROPEN odczytuje pierwsze 4 bajty pliku,
każde wywołanie umieszcza (na stosie) wskaźnik odczytu.
Łańcuch zakończony jest znakiem pustym (0), znakiem końca wiersza (‘\n’) lub powrotu (‘\r’);

Aby sprawdzić, czy osiągnięty został koniec pliku, należy użyć F_EOF
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

(* Program FBD wykorzystujący bloki zarządzania plikiem *)

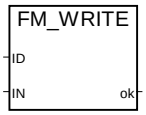


CLOSE_FILE:

```
(* Reprezentacja ST: *)
file_id := F_ROPEN('voltramp.bin');
status1 := FM_READ(file_id);
status2 := FM_READ(file_id);
IF (F_EOF(file_id)) THEN
        ERROR := ERR_FILE;
        unused_eof_mes := FM_READ(file_id);
END_IF;
ok := F_CLOSE(file_id);

(* Reprezentacja IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
        FM_READ         (* odczyt status1 *)
        ST      status1
        LD      file_id
        FM_READ         (* odczyt status2 *)
        ST      status2
        LD      file_id
        F_EOF
        JMPNC   CLOSE_FILE (* jeśli koniec pliku, skok nie jest wykonywany *)
```

```
            LD          ERR_FILE
            ST          ERROR
            LD          file_id
            FM_READ             (* odczyt unused_eof_mes *)
            ST          unused_eof_mes
CLOSE_FILE  LD          file_id
            F_CLOSE
            ST          ok
```

FM_WRITE



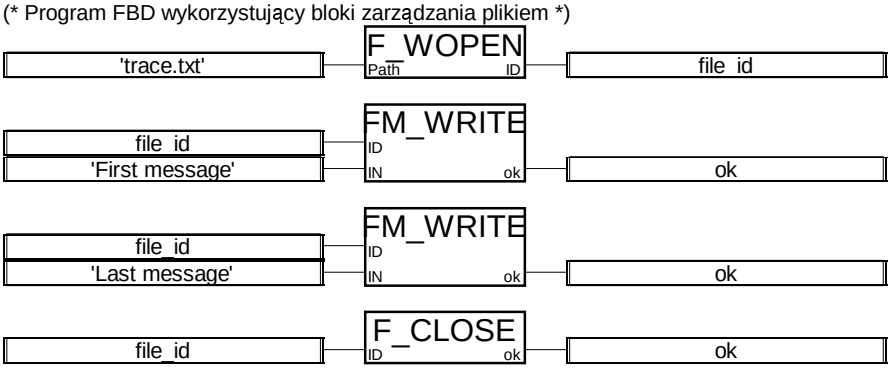
Argumenty:

ID	INT	numer pliku: zwracany przez F_WOPEN.
IN	MSG	wartość komunikatu do zapisania w pliku
ok	BOO	status wykonania PRAWDA, jeśli zapis wykonany pomyślnie

Opis:

Zapisuje zmienne typu KOMUNIKAT do pliku binarnego.
Do użycia z F_WOPEN i F_CLOSE.
Komunikat jest zapisywany w pliku w formie łańcucha zakończonego znakiem pustym
Procedura ta uzyskuje dostęp sekwencyjny do pliku z poprzedniej pozycji.
Pierwsze wywołanie po F_WOPEN zapisuje pierwszy łańcuch do pliku,
każde wywołanie umieszcza (na stosie) wskaźnik zapisu.
Funkcja ta nie jest obsługiwana w symulatorze ISaGRAF.

(* Program FBD wykorzystujący bloki zarządzania plikiem *)




```
(* Reprezentacja ST: *)
file_id := F_WOPEN('trace.txt');
```

```
ok := FM_WRITE(file_id,'First message');
ok := FM_WRITE(file_id,'Last message');
ok := F_CLOSE(file_id);

(* Reprezentacja IL: *)
                LD          'trace.txt'
                F_WOPEN
                ST          file_id
                FM_WRITE'First message'      (* zapis pierwszego komunikatu *)
                ST          ok
                LD          file_id
                FM_WRITE'Last message'       (* zapis drugiego komunikatu *)
                ST          ok
                LD          file_id
                F_CLOSE
                ST          ok
```

C. Podręcznik użytkownika oprogramowania wbudowanego

C.1 Wprowadzenie

Oprogramowanie wbudowane ISaGRAF jest przeznaczone do wykonywania w czasie rzeczywistym, na płycie lub komputerze przemysłowym, aplikacji ISaGRAF zgodnie z następującym dobrze znanym schematem:

Cykl pracy oprogramowania wbudowanego polega na zeskanowaniu fizycznych stanów na wejściach sterowanego procesu, przetworzeniu danych z aplikacji zgodnie z programami aplikacyjnymi pakietu ISaGRAF[1] i na uaktualnieniu fizycznych stanów na wyjściach.

- Pierwsza część tego rozdziału opisuje sposób uruchomienia określonej docelowej implementacji ISaGRAF pod DOS, OS-9, VxWorks i NT. W każdym przypadku użytkownik znajdzie wskazówki wykonania oprogramowania wbudowanego. Dalej zostaną podane informacje o specjalnych funkcjach takich, jak: uruchomienie oprogramowania wbudowanego przy włączaniu, obsługa błędów, postępowanie ogólne, itp.
- Druga część jest poświęcona funkcjom użytkowym języka C, metodzie implementacji bloków funkcji i funkcji konwersji mających na celu wzbogacenie działania oprogramowania wbudowanego.
- Trzecia część dostarcza informacje o realizacji Modbus i ISaGRAF. Zawiera opis formatu ramek różnych kodów funkcji.
- Czwarta część opisuje niektóre narzędzia służące do obsługi awarii zasilania i ponownego uruchomienia oprogramowania wbudowanego.

[1] Przyjęto, że użytkownik zapoznał się z oprogramowaniem pakietu ISaGRAF

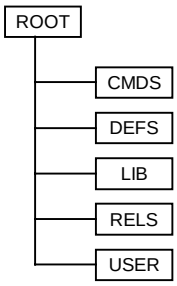
C.2 Instalacja

Do zainstalowania programu potrzeba około 1 megabajta wolnego miejsca na dysku w komputerze użytkownika. Dostarczony na dysku install.bat zainstaluje wszystkie potrzebne dla określonej platformy pliki.

Na przykład: a:\install a: c:*ścieżka*

zainstaluje pliki z napędu dyskowego a: na dysk c: do katalogu określonego *ścieżką*.

Jest stosowana następująca architektura katalogów:



Katalog ROOT zawiera niektóre narzędzia i pliki readme
Katalog CMDS zawiera pliki wykonywalne
Katalog DEFS zawiera pliki z definicją nagłówków
Katalog LIB zawiera biblioteki
Katalog RELS zawiera pliki (pośrednie) przemieszczalne
Katalog USER zawiera procedury użytkowe ‘C’ dla funkcji, bloków funkcyjnych i funkcji konwersji C (pliki źródłowe i pliki nagłówkowe)

Potem należy po prostu uruchomić zainstalowaną platformę.

C.3 Uruchomienie oprogramowania wbudowanego ISaGRAF pod DOS

C.3.1 Wykonywanie ISaGRAF: ISA.EXE

Pod MS-DOS oprogramowanie wbudowane pracuje tak, jak pojedynczy program ISA.EXE. Aby go uruchomić należy wykonać komendę pomocy isa -? z katalogu CMDS. W tym systemie operacje mogą mieć istotne znaczenie. Na przykład, w celu zagwarantowania wysokiej wydajności zaleca się nie przeciążanie łącza komunikacyjnego. Oprogramowanie wbudowane nie blokuje wykonywania procedur standardowych sterowanych przerwaniami.

- ***Łącze komunikacyjne i konfiguracja: opcja -t***

Do komunikowania się z debagerem oprogramowanie wbudowane wykorzystuje łącze szeregowe. Nazwa portu jest określana przy użyciu opcji -t. Ponieważ interfejs komunikacyjny jest przeznaczony do pracy na każdym urządzeniu, więc w zależności od wersji BIOS może być stosowany port COM1, COM2 lub COM3.

Brak wartości domyślnej: Jeśli ta opcja nie zostanie użyta, to nie będzie możliwa komunikacja z oprogramowaniem wbudowanym. W takim wypadku może zostać wyświetlony błąd numer 7.

Jeśli oprogramowanie wbudowane ISaGRAF pracuje pod systemem DOS, to nie będzie możliwa komunikacja za pośrednictwem łącza Ethernet. O specjalną wersję programu należy poprosić swojego dostawcę.

Parametry komunikacji muszą być ustawione przed uruchomieniem ISaGRAF, aby użytkownik mógł całkowicie swobodnie wybrać te, które są mu potrzebne. Stosując debagera pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry komunikacji (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom oprogramowania wbudowanego.

Przykład:
MODE COM1:9600,N,8,1
Ustawia parametry komunikacji na następujące wartości:
- Szybkość transmisji 9600 bodów
- brak kontroli parzystości
- dane 8 bitowe
- 1 bit stopu

Należy zwrócić uwagę, że dla pewnych wersji BIOS domyślne ustawienie szybkości transmisji w pakiecie ISaGRAF na 19200 bodów nie jest dozwolone.
Firma CJ oferuje program usługowy ISAMOD.EXE do ustawiania parametrów pakietu:
ISAMOD COM1 jest równoważny MODE COM1:19200,N,8,1

- Numer sterownika: opcja -s

Opcja ta określa numer sterownika oprogramowania wbudowanego. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Numer sterownika jest używany w protokole komunikacyjnym. Służy głownie do odróżniania od siebie poszczególnych sterowników, gdy uruchomiono więcej niż jedno oprogramowanie wbudowane. Korzystając z debagera pakietu ISaGRAF należy się upewnić, czy numer sterownika w pakiecie ISaGRAF odpowiada numerowi tego sterownika w oprogramowaniu wbudowanym (patrz Podręcznik użytkownika: Zarządzanie programami).

Wartość domyślna: Domyślnym numerem sterownika jest 1 (tak, jak w pakiecie ISaGRAF)

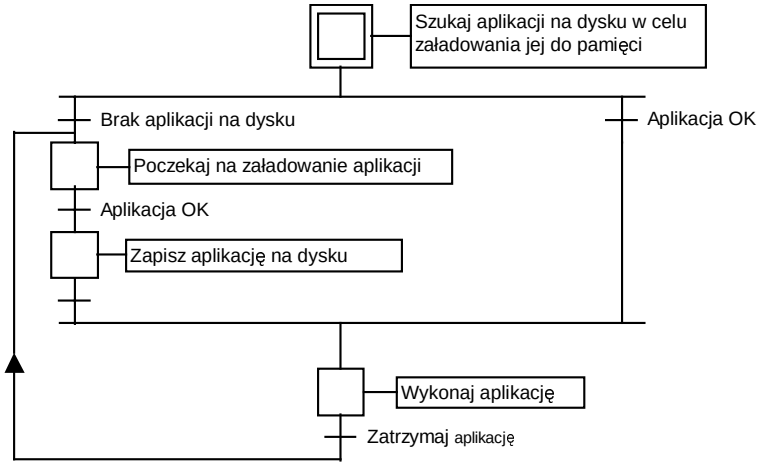
- Przykłady:

isamod COM1 Ustawia COM1 na 19200 bodów, bez kontroli parzystości, przyjmując 8 bitów danych i 1 bit stopu.
isa -t=COM1 Uruchamia oprogramowanie wbudowane z domyślnym numerem sterownika (1) i z COM1 jako portem komunikacyjnym.
isa -s=3 -t=COM1 Uruchamia oprogramowanie wbudowane z numerem sterownika 3 i z COM1 jako portem komunikacyjnym.

C.3.2 Własności specjalne

- ***Uruchomienie ISaGRAF***

Po uruchomieniu oprogramowania wbudowanego jest wykonany następujący algorytm.



- **Definicje**

Kod aplikacji jest bazą binarnych danych generowanych i ładowanych przez pakiet ISaGRAF a potem wykonywanych przez oprogramowanie wbudowane. Może być uzupełniony tablicą symboli.

Tablica symboli aplikacji jest bazą danych ASCII generowanych i załadowanych przez pakiet ISaGRAF. Jest ona ogniwem łączącym obiekty symboliczne i wewnętrzne obiekty sterownika. Jest potrzebna w tym oprogramowaniu wyłącznie do obsługi określonych symboli użytkownika. Więcej informacji o tablicy symboli podano w Podręczniku użytkownika: Zaawansowane techniki programowania.

- **Kopia zapasowa aplikacji**

Ładowanie nowej aplikacji z debagera pakietu ISaGRAF do sterownika spowoduje zapamiętanie kodu aplikacji w bieżącym katalogu oprogramowania pod nazwą:

ISAx1 Kopia zapasowa kodu aplikacji ISaGRAF
(gdzie x jest numerem sterownika)

Ponadto, jeśli uprzednio została wczytana tablica symboli aplikacji, to zostanie ona także zapamiętana w bieżącym katalogu oprogramowania pod nazwą:

ISAx6 Kopia zapasowa symboli aplikacji ISaGRAF
(gdzie x jest numerem sterownika)

Po uruchomieniu oprogramowania wbudowanego ISaGRAF pliki z kodem aplikacji i symbolami aplikacji są wyszukiwane w bieżącym katalogu i ładowane do pamięci. Jeśli nie jest dostępny żaden plik z symbolami, to sterownik rozpoczyna wykonywanie kodu aplikacji bez ładowania symboli.
Jeśli natomiast nie jest dostępny żaden kod, to sterownik czeka na załadowanie aplikacji.

Jeśli pakiet ISaGRAF pracuje na tym samym komputerze, to aby uruchomić oprogramowanie wbudowane z określoną aplikacją, bez stosowania łącza debagera, można pliki skopiować bezpośrednio do bieżącego katalogu sterownika na tym samym dysku, lub skorzystać z dysku elastycznego. Jeśli w urządzeniu z oprogramowaniem wbudowanym nie ma dysku, to można zastosować dysk wirtualny.

Jeśli pakiet ISaGRAF jest zainstalowany w standardowym katalogu \ISAWIN, to plikiem aplikacji projektu MYPROJ jest:

\ISAWIN\APL\MYPROJ\appli.x8m

a plikiem z symbolami aplikacji projektu MYPROJ jest:

\ISAWIN\APL\MYPROJ\appli.tst

Przykład:
Jeśli z katalogu, w którym jest zainstalowane isa.exe zostanie podana następująca komenda:

copy \ISAWIN\APL\MYPROJ\appli.x8m isa11

to isa.exe znajdzie i wykona aplikację ‘myproj’.

Wszystkie te komendy mogą zostać pogrupowane na przykład w pliku wsadowym a potem uruchomione z menu narzędzi pakietu (patrz Podręcznik użytkownika: Zarządzanie programami).

- Obsługa błędów i komunikaty

Oprogramowanie wbudowane ISaGRAF posiada funkcję wykrywania błędów. W załączniku podano listę błędów i ich opis.

Wykrywanie błędów odbywa się w następujący sposób:
Na błąd składa się numer błędu i argument wysyłany do standardowej procedury ISaGRAF przeznaczonej do obsługi błędów

Jeśli w opcjach tworzenia aplikacji w pakiecie ISaGRAF jest ustawiona flaga wykrywania błędów, to błąd zostanie obsłużony. W przeciwnym razie informacja jest tracona i obsługa błędów kończy się.

Podczas obsługi:
Numer błędu (wartość dziesiętna) i argument (wartość szesnastkowa) są wyświetlane na domyślnym wyjściu stdout
Numer błędu i argument są umieszczane w cyklicznym buforze błędów FIFO do wykorzystania w późniejszym czasie. Rozmiar bufora błędów jest ustawiany w pakiecie w opcjach tworzenia aplikacji. Po zapełnieniu się bufora, każdy nowo wprowadzony błąd kasuje błąd najdłużej przechowywany w buforze.
Błędy mogą być wybierane albo z debagera albo z wykonywanej aplikacji przy użyciu wywołania SYSTEM (patrz Podręcznik użytkownika).

Gdy debager wykryje błąd, to w oknie błędu zostaje wyświetlony komunikat opisujący ten błąd. W zależności od stanu aplikacji (wykonywana czy nie) debager może wyświetlić w kwadratowych nawiasach nazwę obiektu (zmiennej lub programu), w którym ten błąd wystąpił lub numer błędu (wartość dziesiętna) postaci [x] o znaczeniu różnym dla każdego błędu.

Po uruchomieniu oprogramowania wbudowanego i wykryciu błędu na domyślnym wyjściu stdout jest wyświetlany komunikat powitalny i wartości błędu. Jeśli nie ma potrzeby wyświetlania tych danych w standardowym kanale wyjściowym, można zastosować komendę przekierowującą taką, jak:

isa -t=COM1 -s=1 **>NUL**

- Zegar systemowy

Ponieważ oprogramowanie wbudowane ISaGRAF jest przeznaczone do pracy pod każdym systemem, to wzorcem czasu służącym zarówno do synchronizacji cyklu jak i do zmiany zmiennych typu timer jest standardowy takt wynoszący około 55 milisekund.

Dlatego nie jest możliwe uzyskanie wyższej dokładności zmiennych timera niż 55 ms. Z tego samego powodu określony czas trwania cyklu, który jest mniejszy lub równy 55 ms i różny od zera, wygeneruje przepełnienie czasu cyklu (błąd 62) i cykl nie zostanie wyzwolony.

Zaletą nie modyfikowania taktu systemu jest niezakłócanie przez wykonywanie ISaGRAF rezydentnej aplikacji lub funkcji C i bloków funkcyjnych włączonych do aplikacji.

Jeśli stosowana przez użytkownika aplikacja wymaga większej dokładności, należy poprosić dostawcę o wersję specjalną.

- Przyciski wyjścia

Testując aplikację w warunkach nieprzemysłowych na komputerze osobistym użytkownik może chcieć zatrzymać ISaGRAF: odbywa się to przez naciśnięcie złożonej kombinacji przycisków, która ma zapobiec niespodziewanym zatrzymaniom. Tą sekwencją przycisków jest:

shift + ctrl + alt

Oczywiście, jeśli naciśnięcie przycisków nie powinno zatrzymać aplikacji przemysłowej, należy przygotować odpowiednie zabezpieczenie blokujące.

Pewnym niebezpiecznym efektem ubocznym takich szybkich przerwań jest nie wykonywanie zamknięcia interfejsu karty We/Wy. Dlatego właściwym sposobem zatrzymania oprogramowania wbudowanego ISaGRAF jest:
- zatrzymanie aplikacji z debagera (spowoduje to zamknięcie kart We/Wy)
- zatrzymanie oprogramowania wbudowanego ISaGRAF z klawiatury.

- Wielkość aplikacji

Ponieważ oprogramowanie wbudowane ISaGRAF pod MS-DOS jest przeznaczone do pracy w czasie rzeczywistym z procesorem Intel, więc maksymalna wielkość struktury danych wynosi 64K. Dlatego ładowany przez pakiet kod aplikacji nie powinien przekroczyć tego limitu. W niektórych bardzo rzadkich przypadkach wewnętrzna struktura wygenerowana przez ISaGRAF może przekroczyć ten limit i po załadowaniu zniszczyć aplikację. Ponadto cała dostępna pamięć jest ograniczona do 640K pamięci konwencjonalnej.

Jeśli stosowana przez użytkownika aplikacja wymaga większej pojemności pamięci, należy poprosić dostawcę o wersję specjalną.

C.4 Uruchomienie oprogramowania wbudowanego ISaGRAF pod OS9

C.4.1 Wykonywanie jednozadaniowej implementacji ISaGRAF: isa

Oprogramowanie wbudowane ISaGRAF może być wykonywany jako pojedyncze zadanie. Jednak w takiej konfiguracji operacje mogą mieć znaczenie krytyczne. Na przykład, w celu zagwarantowania wysokiej wydajności zaleca się nie przeciążanie łącza komunikacyjnego. W systemie wielozadaniowym OS-9 można na tej samej jednostce centralnej wykonywać różne implementacje oprogramowania wbudowanego ISaGRAF w trybie jednozadaniowym o ile numery sterowników i porty komunikacyjne różnią się.
Ta jednozadaniowa implementacja została zaprojektowana głównie dla ubogiej platformy sprzętowej takiej, jak tanie płyty lub komputery osobiste MS-DOS lub w celu wykonania prototypu dla nowej platformy. Dlatego zalecana jest wielozadaniowa implementacja oprogramowania wbudowanego.
Jednozadaniowa wersja oprogramowania wbudowanego umożliwia pracę w tle lub wykonywanie standardowych procedur sterowanych przerwaniami.

- ***Łącze komunikacyjne i konfiguracja: opcja -t***

Jednozadaniowa wersja oprogramowania wbudowanego ISaGRAF komunikuje się z debagerem przez łącze szeregowe. Nazwa deskryptora jest określana opcją -t.

Brak wartości domyślnej: Jeśli ta opcja nie jest używana, to nie będzie możliwa komunikacja z oprogramowaniem wbudowanym. W takim wypadku może zostać wyświetlony błąd numer 7.

Z jednozadaniową wersją oprogramowania wbudowanego nie można komunikować się za pośrednictwem łącza Ethernet.
Łącze szeregowe jest otwarte w trybie przesyłania danych binarnych (bez znaków sterujących, bez XON/XOFF). Inne parametry komunikacji muszą zostać ustawione przed uruchomieniem ISaGRAF, aby użytkownik mógł całkowicie swobodnie wybrać potrzebne mu parametry. Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry komunikacji (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom oprogramowania wbudowanego.
Przykład:
xmode /t0 baud=19200
Ustawia szybkość transmisji na 19200 bodów w urządzeniu /t0.

- Numer sterownika: opcja -s

Ta opcja określa numer sterownika oprogramowania wbudowanego. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Numer sterownika jest używany w protokole komunikacyjnym. Służy do odróżniania od siebie poszczególnych implementacji oprogramowania wbudowanego, gdy uruchomiono ich więcej niż jedną. Korzystając z debagera pakietu ISaGRAF należy się upewnić, czy numer sterownika pakietu odpowiada odnośnemu parametrowi aktualnej implementacji oprogramowania wbudowanego (patrz Podręcznik użytkownika: Zarządzanie programami).

Wartość domyślna: Domyślnym numerem sterownika jest 1 (tak, jak w pakiecie ISaGRAF)

- Przykłady:

isa -t=/t0 Uruchamia jednozadaniowe oprogramowanie wbudowane ISaGRAF z domyślnym numer sterownika (1) i z /t0 jako portem komunikacyjnym.
isa -s=3 -t=/t1 Uruchamia jednozadaniowe oprogramowanie wbudowane ISaGRAF z numer sterownika 3 i z /t1 jako portem komunikacyjnym.
isa -t=/t0 i
isa -s=3 -t=/t1 Uruchamia dwie jednozadaniowe implementacje oprogramowania wbudowanego ISaGRAF – jedną z domyślnym numer sterownika (1) i z /t0 jako portem komunikacyjnym a drugą z numerem sterownika 3 i z /t1 jako portem komunikacyjnym.

C.4.2 Wykonywanie wielozadaniowej implementacji ISaGRAF: isaker, isatst, isanet

Aby poprawić czas reakcji jądra oprogramowania wbudowanego ISaGRAF i łącza komunikacyjnego, podzielono oprogramowanie wbudowane na dwa zadania oddzielając zadanie komunikacji (isatst lub isanet) od zadania wykonywania aplikacji (zadanie jądra isaker).
Taka architektura jest bardziej elastyczna. Umożliwia użytkownikowi wykonywanie więcej niż jednego zadania komunikacji związanego z tym samym zadaniem jądra lub na wykonywanie do 4 jąder z tym samym zadaniem komunikacji. Ułatwia to konsolidację łącza do wizualizacji procesu i łącza debagera pakietu ISaGRAF w tej samej aplikacji lub wykorzystanie jednego łącza dla aż 4 różnych aplikacji poprzez ten sam fizyczny port.

Zadania jądra i komunikacji są od siebie niezależne i mogą być oddzielnie wybierane. Wymagane jest jedynie rozpoczęcie najpierw zadania (zadań) jądra, aby przygotować środowisko pracy systemu i włączyć do niego zadanie (zadania) komunikacji.

Wielozadaniowa implementacja oprogramowania wbudowanego ISaGRAF umożliwia pracę w tle lub wykonywanie standardowych procedur sterowanych przerwaniami.

C.4.2.1 Wykonywanie zadania jądra: isaker

- ***Numer sterownika: opcja -s***

Ta opcja określa numer sterownika jądra oprogramowania wbudowanego. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Numer sterownika jest używany w protokole komunikacyjnym i w związanych z jądrem zadaniach komunikacji . Służy do odróżniania od siebie sterowników oprogramowania wbudowanego, gdy uruchomiono ich więcej niż jeden.

Wartość domyślna: Domyślnym numerem sterownika jest 1 (tak, jak w pakiecie ISaGRAF)

C.4.2.2 Wykonywanie zadania komunikacji szeregowej: isatst

- ***Łącze komunikacyjne i konfiguracja: opcja -t***

Zadanie komunikacji isatst stosuje łącze szeregowe do komunikacji z debagerem. Nazwa deskryptora jest określona opcją -t.

Brak wartości domyślnej: Jeśli ta opcja nie jest używana, to nie będzie możliwa komunikacja z oprogramowaniem wbudowanym. W takim wypadku może zostać wyświetlony błąd numer 7.

Z implementacją isatst nie można komunikować się za pośrednictwem łącza Ethernet.

Łącze szeregowe jest otwarte w trybie przesyłania danych binarnych (bez znaków sterujących, bez XON/XOFF). Inne parametry komunikacji muszą zostać ustawione przed uruchomieniem ISaGRAF, aby użytkownik mógł całkowicie swobodnie wybrać potrzebne mu parametry. Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry komunikacji (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom oprogramowania wbudowanego.

Przykład:
xmode /t0 baud=19200
Ustawia szybkość transmisji na 19200 bodów w urządzeniu /t0.

- Numer sterownika: opcja -s

Ta opcja określa numer(y) sterownika jądra oprogramowania wbudowanego, z którym jest związane zadanie komunikacji. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Opcję można powtórzy aż 4 razy i dołączyć 4 różne jądra. Numer sterownika jest używany w protokole komunikacyjnym. Służy do odróżniania od siebie poszczególnych sterowników, gdy uruchomiono ich więcej niż jeden. Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry komunikacji (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom aktualnego oprogramowania wbudowanego (zadania jądra i komunikacji).

Wartość domyślna: Domyślnym numerem sterownika jest 1 (tak, jak w pakiecie ISaGRAF)

- Numer logiczny zadania komunikacji: opcja -c

Ta opcja określa numer logiczny zadania komunikacji. Służy do jednoczesnego zarządzania więcej niż jednym zadaniem. Może być liczbą z przedziału 1 do 255 i musi mieć inną wartość dla każdego zadania.

Wartość domyślna: Stosowana jest ostatnio określona opcja -s. Wartość domyślna zapewnia kompatybilność z poprzednimi wersjami ISaGRAF (3.0).

C.4.2.3 Wykonywanie zadanie komunikacji poprzez Ethernet: isanet

- ***Łącze komunikacyjne i konfiguracja: opcja -t***

Zadanie komunikacji isanet komunikuje się z debagerem poprzez standardowe łącze Ethernet. Numer portu jest określony opcją -t.

Brak wartości domyślnej: Jeśli ta opcja nie jest używana, to nie będzie możliwa komunikacja z oprogramowaniem wbudowanym. W takim wypadku może zostać wyświetlony błąd numer 7.

Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry komunikacji (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom oprogramowania wbudowanego.

Z punktu widzenia ISaGRAF oprogramowanie wbudowane pracujące pod OS-9 jest serwerem a debager jest klientem łączącym się z określonym numerem portu.
Przed rozpoczęciem pierwszej sesji debagera w sieci Ethernet należy się upewnić, czy karta Ethernet pod OS-9 jest dobrze skonfigurowana. W tym celu można na przykład wysłać do systemu OS-9 komendę ping.

Numer sterownika: opcja -s

Ta opcja określa numer sterownika jądra oprogramowania wbudowanego, z którym związane jest zadanie komunikacji. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Opcję można powtórzy aż 4 razy i dołączyć 4 różne jądra. Numer sterownika jest używany w protokole komunikacyjnym. Służy do odróżniania od siebie implementacji oprogramowania wbudowanego, gdy uruchomiono ich więcej niż jedną. Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry komunikacji (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom aktualnego oprogramowania wbudowanego (zadania jądra i komunikacji).

Wartość domyślna: Domyślnym numerem sterownika jest 1 (tak, jak w pakiecie ISaGRAF)

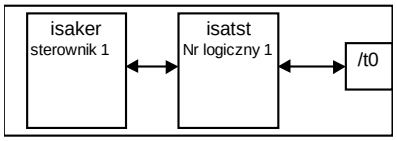
Numer logiczny zadania komunikacji: opcja -c

Ta opcja określa numer logiczny zadania komunikacji. Służy do jednoczesnego zarządzania więcej niż jednym zadaniem. Może być liczbą z przedziału 1 do 255 i musi mieć inną wartość dla każdego zadania.

Wartość domyślna: Stosowana jest ostatnio określona opcja -s. Wartość domyślna zapewnia kompatybilność z poprzednimi wersjami ISaGRAF (3.0).

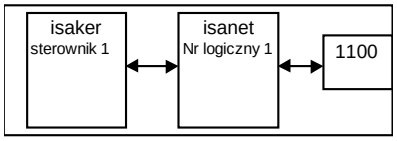
C.4.2.4 Przykłady:

isaker i
isatst -t=/t0



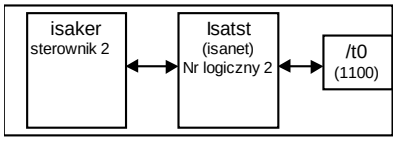
Start:
Zadanie jądra ISaGRAF o domyślnym numerze sterownika (1).
Zadanie komunikacji szeregowej ISaGRAF w porcie /t0, związane ze sterownikiem o numerze (1), oznaczone domyślnym numerem logicznym (ostatnio określony numer sterownika = numer domyślny = 1).

isaker i
isanet -t=1100



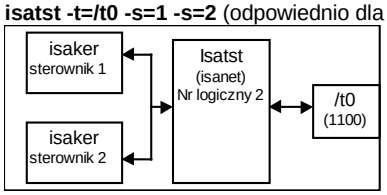
Start:
Zadanie jądra ISaGRAF o domyślnym numerze sterownika (1).
Zadanie komunikacji ISaGRAF poprzez Ethernet w porcie o numerze 1100, związane z sterownikiem o domyślnym numerze (1), oznaczone domyślnym numerem logicznym (ostatnio określony numer sterownika= numer domyślny = 1).

isaker -s=2 i
isatst -t=/t0 -s=2 (odpowiednio dla isanet -t=1100 -s=2)



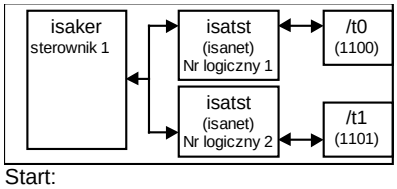
Start:
Zadanie jądra ISaGRAF o numerze sterownika 2.
Zadanie komunikacji szeregowej (Ethernet), w porcie /t0 (o numerze 1100), związane ze sterownikiem o numerze 2, oznaczone domyślnym numerem logicznym (ostatnio określony numer sterownika = numer domyślny = 2).

Isaker -s=1 i
isaker -s=2 i
isatst -t=/t0 -s=1 -s=2 (odpowiednio dla isanet -t=1100 -s=1 -s=2)



Start:
Zadanie jądra ISaGRAF o numerze sterownika 1.
Zadanie jądra ISaGRAF o numerze sterownika 2.
Zadanie komunikacji szeregowej (Ethernet), w porcie /t0 (o numerze 1100), związane ze sterownikami o numerach 1 i 2, oznaczone domyślnym numerem logicznym (ostatnio określony numer sterownika = 2).

Isaker -s=1 i
isatst -t=/t0 -s=1 -c=1 (odpowiednio dla isanet -t=1100 -s=1 -c=1) i
isatst -t=/t1 -s=1 -c=2 (odpowiednio dla isanet -t=1101 -s=1 -c=2)



Start:
Zadanie jądra ISaGRAF z numerem sterownika 1.
Zadanie komunikacji szeregowej (Ethernet), w porcie /t0 (o numerze 1100), związane ze sterownikiem o numerze 1, oznaczone domyślnym numerem logicznym 1.
Zadanie komunikacji szeregowej (Ethernet), w porcie /t1 (o numerze 1101), związane ze sterownikiem o numerze 1, oznaczone numerem logicznym 2.
Uwaga:
Można łączyć zadania komunikacji szeregowej i poprzez Ethernet.

C.4.3 Własności specjalne

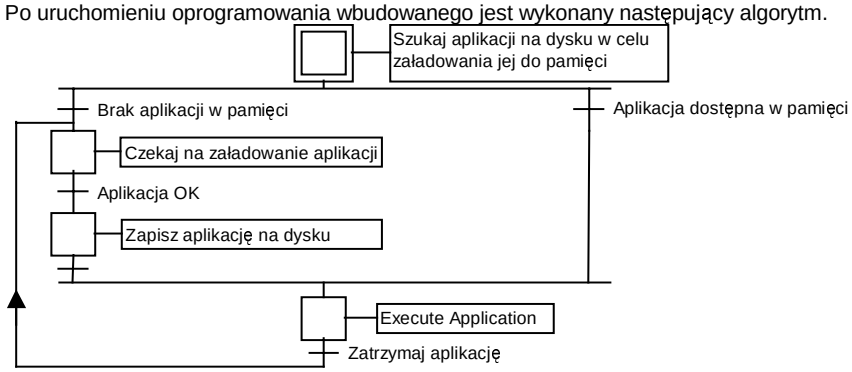
- ***Łącze komunikacyjne***

Ponieważ Serial Character Manager systemu OS-9 jest narzędziem bardzo elastycznym, więc można wykorzystać niemal każde obsługiwane przez Microware urządzenie do transmisji dwukierunkowej:

Przykład:
Łącze szeregowe może być sieciową ścieżką do fizycznego portu zlokalizowanego w innej jednostce centralnej. Następnie można zastosować opcję -t na przykład następująco: -t=/nr/MASTER/t0
Jeśli łącze komunikacyjne korzysta z portu w jednostce centralnej o nazwie MASTER w sieci ramnet, to fizycznie stosowanym portem jest /t0.

- Uruchomienie ISaGRAF

Po uruchomieniu oprogramowania wbudowanego jest wykonany następujący algorytm.



Definicje

Kod aplikacji jest bazą binarnych danych generowanych i ładowanych przez pakiet ISaGRAF a potem wykonywanych przez oprogramowanie wbudowane. Może być uzupełniony tablicą symboli.
Tablica symboli aplikacji jest bazą danych ASCII generowanych i ładowanych przez pakiet ISaGRAF. Jest ona ogniwem łączącym obiekty symboliczne i wewnętrzne obiekty oprogramowania wbudowanego. Jest potrzebna w oprogramowaniu wbudowanym wyłącznie do obsługi określonych symboli użytkownika. Więcej informacji o tablicy symboli podano w Podręczniku użytkownika: Zaawansowane techniki programowania.
Obiekty i aplikacje ISaGRAF pod OS-9
Nazwa każdego ogólnego obiektu ISaGRAF rozpoczyna się od '**ISAxn**', gdzie **x** jest numerem sterownika jądra a **n** numerem segmentu pamięci o specjalnym znaczeniu, oprócz **ISAy3**, gdzie **y** jest numerem logicznym zadania komunikacji w implementacji wielozadaniowej.
W jednostce centralnej mogą jednocześnie pracować różne aplikacje (zadania jądra i komunikacji), o ile odpowiadają im różne numery sterowników i różne numery logiczne zadań komunikacji. Niemniej jednak wykonując różne aplikacje użytkownik musi ostrożnie postępować z pewnymi powszechnie udostępnianymi obiektami aplikacji, jak karty We/Wy. Na przykład różne aplikacje (jądra) mogą korzystać z odrębnych kart fizycznych o ile w sterowniku We/Wy nie jest wprowadzony pewnego rodzaju serwer lub semafor We/Wy.

Nazwy obiektów OS-9:
Pliki dyskowe:

ISAx1	Kopia zapasowa kodu aplikacji ISaGRAF
ISAx6	Kopia zapasowa symbolu aplikacji ISaGRAF

Moduły pamięci:

ISAx0	Dane systemowe o jądrze ISaGRAF
ISAx1	Kod aplikacji ISaGRAF
ISAx2	Działająca w czasie rzeczywistym baza danych jądra ISaGRAF
ISAy3	Bufor ISaGRAF do wymiany danych o transmisji
ISAx4	Kod aplikacji ISaGRAF do modyfikacji bezpośredniej 1
ISAx5	Kod aplikacji ISaGRAF do modyfikacji bezpośredniej 2
ISAx6	Symbol aplikacji ISaGRAF

Dlatego użytkownik musi starać się nie stosować tych samych nazw obiektów.
Kopia zapasowa aplikacji
Ładowanie nowej aplikacji z debagera pakietu ISaGRAF do oprogramowania wbudowanego spowoduje zapamiętanie kodu aplikacji w bieżącym katalogu oprogramowania wbudowanego pod nazwą:

ISAx1	Kopia zapasowa kodu aplikacji ISaGRAF (gdzie x jest numerem sterownika)

Ponadto, jeśli uprzednio została wczytana tablica symboli aplikacji, to zostanie ona także zapamiętana w bieżącym katalogu oprogramowania wbudowanego pod nazwą:

ISAx6	Kopia zapasowa symboli aplikacji ISaGRAF (gdzie x jest numerem sterownika)

Po uruchomieniu oprogramowania wbudowanego ISaGRAF pliki z kodem aplikacji i symbolami aplikacji są wyszukiwane w bieżącym katalogu i ładowane do pamięci jako moduły danych o takich samych nazwach.

Następnie, jeśli w pamięci nie ma tablicy symboli, to sterownik rozpoczyna wykonywanie kodu aplikacji bez żadnych symboli. Jeśli natomiast w pamięci nie ma żadnego kodu, to sterownik czeka na załadowanie aplikacji.

Aby uruchomić oprogramowanie wbudowane z określoną aplikacją przy włączaniu i bez stosowania łącza debagera można zastosować dwa sposoby:

- pierwszy sposób może polegać na bezpośrednim skopiowaniu plików do bieżącego katalogu oprogramowania wbudowanego na dysku z komputera PC, na którym jest zainstalowany pakiet ISaGRAF, przy pomocy narzędzi do przysyłania plików. Dla ułatwienia tych czynności można zastosować menu narzędzi pakietu ISaGRAF (patrz Podręcznik użytkownika: Zarządzanie programami).

- Drugi sposób może polegać na zapamiętaniu, przy pomocy własnych narzędzi, kodu aplikacji (a w razie potrzeby także tablicy symboli aplikacji) w pamięci trwałej (takiej, jak PROM lub EPROM) z plików z komputera PC, na którym jest zainstalowany pakiet ISaGRAF.

Potem, jeśli trzeba (na przykład z powodu szybszego dostępu lub zarządzania pułapkami), można przy uruchamianiu systemu załadować przy pomocy własnych narzędzi kod aplikacji (i jeśli trzeba tablicę symboli aplikacji) z PROM do RAM w postaci modułu danych **ISAx1** (i jeśli trzeba **ISAx6**).

OSTRZEŻENIE:
W debagerze ISaGRAF zarządzanie pułapkami nie będzie wykonywane prawidłowo, jeśli moduł z kodem aplikacji nie może być zapisywany. Nie jest to problemem, ponieważ aplikacja jest normalnie poddawana gruntownemu uprzedniemu testowaniu.

Jeśli pakiet ISaGRAF jest zainstalowany w standardowym katalogu \ISAWIN, to w komputerze PC:
plikiem zawierającym kod aplikacji projektu MYPROJ jest:
\ISAWIN\APL\MYPROJ\appli.x6m (odpowiadający isax1 w oprogramowaniu wbudowanym),
plikiem zawierającym symbole aplikacji projektu MYPROJ jest:
\ISAWIN\APL\MYPROJ\appli.tst (odpowiadający isax6 w oprogramowaniu wbudowanym).

- Obsługa błędów i komunikaty

Oprogramowanie wbudowane ISaGRAF łączy w sobie funkcję wykrywania błędów. W załączniku podano listę błędów ostrzegawczych i ich opis.

Wykrywanie błędów odbywa się w następujący sposób:
Na błąd składa się numer błędu i argument wysyłany do standardowej procedury ISaGRAF do obsługi błędów
Jeśli w opcjach tworzenia aplikacji pakietu ISaGRAF jest ustawiona flaga wykrywania błędów, to błąd zostanie obsłużony. W przeciwnym razie informacje jest tracona i obsługa błędów kończy się.

Podczas obsługi:
Numer błędu (wartość dziesiętna) i argument (wartość szesnastkowa) są wyświetlane na domyślnym wyjściu stdout
Numer błędu i argument są umieszczane w cyklicznym buforze błędów FIFO w celu wykorzystania ich w późniejszym czasie. Rozmiar bufora błędów jest ustawiany w opcjach

tworzenia aplikacji. Po zapełnieniu się bufora, każdy nowo wprowadzony błąd kasuje błąd najdłużej przechowywany w buforze.
Błędy mogą być wyciągane albo z debagera albo z wykonywanej aplikacji przy użyciu wywołania SYSTEM (patrz Podręcznik użytkownika).

Gdy debager wykryje błąd, to w oknie błędu zostaje wyświetlony komunikat opisujący ten błąd. W zależności od stanu aplikacji (wykonywana czy nie) debager może wyświetlić w kwadratowych nawiasach nazwę obiektu (zmiennej lub programu), w którym ten błąd wystąpił lub numer błędu (wartość dziesiętna) postaci [x], który ma różne znaczenie dla każdego błędu.

Po uruchomieniu oprogramowania wbudowanego i wykryciu błędu na domyślnym wyjściu stdout jest wyświetlany komunikat powitalny i wartości błędu. Jeśli nie ma potrzeby wyświetlania tych danych w standardowym kanale wyjściowym, można zastosować komendę przekierowującą taką, jak:

prog_name [options] **>>>/nil**

- Czas trwania cyklu, realizacja zadań i priorytety

Pod koniec jednego cyklu ISaGRAF, a przed rozpoczęciem następnego jest wykonywany następujący algorytm:
Jeśli są określone parametry czasowe cyklu (z pakietu ISaGRAF: patrz Podręcznik użytkownika: Zarządzanie programami), to przez pozostały czas jednostka centralna nie będzie sterowana (określony czas cyklu – czas wykonywania bieżącej aplikacji). Jeśli ten pozostały okres czasu jest ujemny, to zostanie wygenerowane przepełnienie i jednostka centralna nie będzie sterowana przez 1 takt w celu wymuszenia wykonywania zadań systemowych.

Jeśli taktowanie cyklu nie jest określone lub jeśli pozostały czas jest mniejszy lub równy 1 taktowi lub jest równy zeru, to jednostka centralna nie będzie sterowana przez 1 takt, aby wymusić wykonywanie zadań systemowych.

Dokładność taktowania oprogramowania wbudowanego odpowiada jednemu taktowi systemu OS-9.

Taktowanie z określoną częstotliwością jest powszechnie stosowane do wyzwalania cykli lub do wykorzystania jednostki centralnej do innych zadań wykonywanych pod systemem OS-9.

Jeśli łączem komunikacyjnym nie są przesyłane żadne dane, to zadanie komunikacji jest w stanie czuwania. W odpowiedniej chwili za pośrednictwem protokołu typu pytanie / odpowiedź zadanie otrzymuje z jądra informacje o pracującej aplikacji. Zadanie komunikacja wysyła do jądra pytanie. Pod koniec cyklu jądro przekazuje odpowiedź do zadania komunikacji (aby mieć zsynchronizowany obraz aplikacji).

Zadania ISaGRAF nie modyfikują nadanego im priorytetu. Użytkownik ma swobodę nadawania priorytetów w zależności od opisanego powyżej sposobu realizacji zadania ISaGRAF i ogólnych wymagań aplikacji.
Na przykład, aby mieć pewność, że ISaGRAF nie zostanie wyparty przez zadanie o niższym priorytecie, można modyfikować takie parametry zarządzania zadaniami pod OS-9, jak **MIN_AGE** i **MAX_AGE**.

- Praca z terminala

Protokół komunikacyjny oprogramowania wbudowanego rozpoznaje sekwencje 3 znaków powrotu karetki ($0D) a potem rozpoczyna realizację nakładki OS-9 korzystając z łącza

szeregowego. Dzięki temu przy pomocy łącza szeregowego nakładka OS-9 jest natychmiast na terminalu.

Przykład:
Z komputera PC:
Zamknij debager ISaGRAF.
Rozpocznij sesję z terminala pod Windows (grupa akcesoriów) z właściwymi parametrami komunikacyjnymi
Naciśnij 3 razy znak powrotu karetki

Nastąpi zalogowanie się użytkownika w nakładce OS-9
Aby zakończyć pracę z terminala podaj polecenie **logout**.

OSTRZEŻENIE:
Sesję z terminala należy zawsze poprawnie zamykać przy użyciu logout i nic ponadto. W przeciwnym razie następne połączenie z pakietem ISaGRAF nie zakończy się pomyślnie.

C.5 Uruchomienie oprogramowania wbudowanego ISaGRAF pod VxWorks

Praca jednej lub kilku implementacji oprogramowania wbudowanego ISaGRAF wymaga wykonania kilku komend pod systemem VxWorks system, w celu ustawienia parametrów konfiguracji i ostatecznego poskładania oprogramowania. Wszystkie te komendy mogą być zainicjowane z pliku skryptowego. Są one opisane w następnych rozdziałach.

C.5.1 Menedżer zasobów systemu: isassr.o

Ten moduł jest niezbędny w każdej konfiguracji oprogramowania wbudowanego ISaGRAF i musi być ładowany jako pierwszy. Umożliwia zarządzanie zasobami systemu, w którym ma pracować kilka implementacji oprogramowania wbudowanego.

C.5.2 Typowe własności isa.o, isakerse.o i isakeret.o

Aby uruchomić ISaGRAF należy załadować jeden z następujących modułów

isa.o: umożliwia uruchomienie jednozadaniowych implementacji oprogramowania wbudowanego ISaGRAF (tylko łącze do komunikacji szeregowej).

Isakerse.o: umożliwia uruchomienie wielozadaniowych implementacji oprogramowania wbudowanego ISaGRAF (tylko łącze do komunikacji szeregowej).

Isakeret.o: umożliwia uruchomienie wielozadaniowych implementacji oprogramowania wbudowanego ISaGRAF (łącze do komunikacji szeregowej i/lub Ethernet)

Szczegółowy opis modułów podano w następnych rozdziałach.

Konfiguracja łącza do komunikacji szeregowej

Do komunikacji z debagerem oprogramowanie wbudowane ISaGRAF wykorzystuje zasadniczo łącze szeregowe. Uruchomiony ISaGRAF nie wykonuje żadnych operacji konfigurujących określone łącze szeregowe. W ten sposób użytkownik może całkowicie swobodnie ustawiać potrzebne mu parametry. Konieczna jest jednak funkcja binarnego przesyłania danych (funkcja RAW). Do jej realizacji służy podprogram *ISAMOD ()*.

```
uchar ISAMOD
  (
  char *desc,        /* Nazwa łącza szeregowego */
  uint32 baudrate    /* Szybkość transmisji     */
  )
```

Opis:
Konfiguruje określone łącze szeregowe do binarnego przesyłania danych z określoną szybkością.

wartość zwracana:
0 jeśli wszystko odbywa się pomyślnie, BAD_RET jeśli wystąpią błędy

Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry komunikacji (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom oprogramowania wbudowanego.

- Częstotliwość pracy zegara systemowego

Zmienna globalna CLKRATE (uint32) wymaga wstępnego ustawienia na częstotliwość zegara systemu VxWorks. W tym celu można zastosować:

CLKRATE = sysClkRateGet ()

Wartością domyślną CLKRATE jest 60 Hz.

C.5.3 Wykonywanie jednozadaniowej implementacji ISaGRAF: isa.o

ISaGRAF target może być wykonywany jako pojedyncze zadanie. Jednak w takiej konfiguracji operacje mogą mieć znaczenie krytyczne. Na przykład w celu zagwarantowania wysokiej wydajności zaleca się nie przeciążanie łącza komunikacyjnego. W systemie wielozadaniowym VxWorks można w tej samej jednostce centralnej wykonywać różne implementacje oprogramowania wbudowanego ISaGRAF w trybie jednozadaniowym, o ile ich numery sterowników i porty komunikacyjne różnią się.
Ta jednozadaniowa implementacja została zaprojektowana głównie dla ubogiej platformy sprzętowej takiej, jak tanie płyty lub komputery osobiste MS-DOS lub w celu wykonania prototypu dla nowej platformy. Dlatego zalecana jest wielozadaniowa implementacja oprogramowania wbudowanego ISaGRAF.
Jednozadaniowa implementacja oprogramowania wbudowanego ISaGRAF umożliwia pracę w tle lub wykonywanie standardowych procedur sterowanych przerwaniami.

- ***Rejestracja sterownika***

Oprogramowanie wbudowane ISaGRAF jest oznaczane numerem sterownika. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Numer sterownika jest używany w protokole komunikacyjnym. Służy do odróżniania od siebie poszczególnych implementacji oprogramowania wbudowanego, gdy uruchomiono ich więcej niż jedną. Dlatego przed uruchomieniem zadania ISaGRAF należy zarejestrować numer sterownika. Temu celowi służy podprogram *isa_register_slave()*.

```
uchar isa_register_slave
  (
  uchar slave    /* numer sterownika */
  )
```

Opis:
Wprowadza nowy numer sterownika do systemu zarządzania pracą wielu implementacji oprogramowania wbudowanego.

wartość zwracana:
0 jeśli wszystko przebiega pomyślnie, BAD_RET jeśli wystąpią błędy

- Zapamiętywanie kopii zapasowej aplikacji

Zmienna globalna TSK_FUNIT (char *) może wstępnie zawierać ścieżkę do jednostki pamięci, w której będzie zapisana kopia zapasowa aplikacji. Do tworzenia kopii zapasowej oprogramowanie wbudowane ISaGRAF stosuje po prostu standardowe procedury zarządzania plikami fopen, fread, fwrite, fclose.
Wartością domyślną jest pusty ciąg ("") oznaczający brak jednostki pamięci.

Przykład:
TSK_FUNIT = "*host name:*/C:/ISaGRAF/target/apl/"

Określa jednostkę pamięci jako ISaGRAF\target\apl\ na dysku C: w komputerze *host_name* , Jest to katalog, w którym na zostać zapisana kopia zapasowa aplikacji. Nie wolno zapomnieć o ostatniej kresce ukośnej, gdyż w przeciwnym razie kopia zapasowa zostanie zapisana w katalogu ISaGRAF\target\ a pliki będą miały w nazwie przedrostek apl.

Zmienna ta może zostać ustawiona na różne jednostki pamięci dla poszczególnych implementacji oprogramowania wbudowanego przed ich poskładaniem.
Więcej szczegółowych informacji o plikach zapasowych aplikacji podano w rozdziale Własności specjalne; Kopia zapasowa aplikacji.

- Kontrola końca cyklu

Zmienna TSK_NBTCKSCHED (uint 32) może być ustawiona na wartość określającą opóźnienie taktu, jakie jest stosowane przez oprogramowanie wbudowane ISaGRAF na koniec cyklu.
Domyślną wartością jest 0 (tak samo, jak dla szeregowania zadań priorytetowych).
Zmienna ta może zostać ustawiona na wartość inną dla poszczególnych implementacji oprogramowania wbudowanego przed ich zainstalowaniem.
Więcej szczegółowych informacji podano w rozdziale Własności specjalne; Czas trwania cyklu, realizacja zadań i priorytety.

- Składanie oprogramowania wbudowanego ISaGRAF

Po ustawieniu konfiguracji ostatni krok polega na skompletowaniu oprogramowania wbudowanego ISaGRAF: isa_main.

```
uchar isa_main
(
uchar slave,        /* Numer sterownika       */
char *com           /* Nazwa łącza szeregowego */
)
```

Opis:
Rozpoczyna zadanie wykonania oprogramowania wbudowanego ISaGRAF.

wartość zwracana:
będzie to liczba różna od zera jeśli wystąpią błędy.

Numer sterownika jest taki sam, jaki omówiono w rozdziale o rejestrowaniu sterownika.
Można uruchomić więcej niż jedną implementację oprogramowania wbudowanego, o ile odpowiadające im numery sterowników i porty komunikacyjne są różne.
Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry sterownika (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom aktualnego oprogramowania wbudowanego.

- Przykład

Przykład przedstawia sposób uruchomienia jednozadaniowej implementacji oprogramowania wbudowanego ISaGRAF o numerze sterownika 1 i łączu szeregowym /tyCo/1.
Katalogiem bieżącym będzie ten, w którym jest zainstalowane oprogramowanie wbudowane.

Załaduj moduł isassr.o
ld < RELS/isassr.o

Załaduj moduł isa.o
ld < CMDS/isa.o

Konfiguracja łącza do komunikacji szeregowej
ISAMOD ("/tyCo/1", 19200)

Częstotliwość zegara systemowego
CLKRATE = sysClkRateGet ()

Numer sterownika
isa_register_slave (1)

Jednostka pamięci do zapisywania plików (można pominąć, gdyż jest ustawiona domyślnie)
TSK_FUNIT = ""

Kontrola końca cyklu (można pominąć, gdyż jest ustawiona domyślnie)
TSK_NBTCKSCHED = 0

Składanie oprogramowania wbudowanego ISaGRAF
sp (isa_main, 1, "/tyCo/1")

C.5.4 Wykonywanie wielozadaniowej implementacji ISaGRAF: isakerse.o i isakeret.o

Aby poprawić czas reakcji jądra oprogramowania wbudowanego ISaGRAF i łącza komunikacyjnego, podzielono oprogramowanie wbudowane na dwa zadania oddzielając zadanie komunikacyjne (isatst lub isanet) od wykonywania aplikacji (zadanie jądra).
Taka architektura jest bardziej elastyczna. Umożliwia użytkownikowi wykonywanie więcej niż jednego zadania komunikacji związanego z tym samym zadaniem jądra lub na wykonywanie do 4 jąder z tym samym zadaniem komunikacji. Ułatwia to integrację łącza do wizualizacji procesu i łącza debagera pakietu ISaGRAF w tej samej aplikacji lub jednego łącza dla aż 4 różnych aplikacji poprzez ten sam fizyczny port.

Zadania jądra i komunikacji są od siebie niezależne i mogą być oddzielnie składane. Wymagane jest jedynie rozpoczęcie najpierw zadania (zadań) jądra, aby przygotować środowisko pracy systemu i włączyć do niego zadanie (zadania) komunikacji.

Wielozadaniowa implementacja oprogramowania wbudowanego ISaGRAF umożliwia pracę w tle lub wykonywanie standardowych procedur sterowanych przerwaniami.

W zależności od możliwości sprzętu komunikacyjnego proponuje się dwa moduły:
- Jądro i łącze szeregowe: isakerse.o
Ten moduł umożliwia rozpoczęcia zadania (zadań) jądra i zadania (zadań) komunikacji szeregowej.
- Jądro i łącze szeregowe i/lub łącze Ethernet: isakeret.o
Ten moduł umożliwia rozpoczęcia zadania (zadań) jądra i zadania (zadań) komunikacji szeregowej i/lub poprzez Ethernet.

ISaGRAF uruchamia się z modułu isakerse.o i isakeret.o w taki sam sposób, z tym wyjątkiem, że dla isakeret.o nazwę łącza szeregowego lub nazwę portu dla łącza Ethernet można określić w formie parametrycznej: tst_main_ex podczas uruchamiania zadania (zadań) komunikacji (patrz poniżej).

Z punktu widzenia ISaGRAF oprogramowanie wbudowane pracujące pod VxWorks jest serwerem a debager jest klientem łączącym się z określonym numerem portu.

- ***Rejestracja jądra (jąder)***

Jądro ISaGRAF jest oznaczane numerem odpowiadającego mu sterownika. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Numer sterownika jest używany w protokole komunikacyjnym i poprzez zadania komunikacji związane z jądrem. Służy do odróżniania od siebie poszczególnych implementacji oprogramowania wbudowanego, gdy uruchomiono ich więcej niż jedną. Dlatego przed uruchomieniem zadania (zadań) jądra ISaGRAF należy je zarejestrować. Temu celowi służy podprogram *isa_register_slave()*.

```
uchar isa_register_slave
  (
  uchar slave    /* numer sterownika */
  )
```

Opis:
Wprowadza nowy numer sterownika do systemu zarządzania pracą wielu realizacji oprogramowania wbudowanego.

wartość zwracana:
0 jeśli wszystko odbywa się pomyślnie, BAD_RET jeśli wystąpią błędy

- Rejestracja zadania (zadań) komunikacji

Zadanie komunikacji ISaGRAF jest oznaczone numerem logicznym. Służy do jednoczesnego zarządzania więcej niż jednym zadaniem komunikacji. Może być liczbą z przedziału od 1 do 255, różną dla każdego zadania. Dlatego zanim zadanie komunikacji ISaGRAF rozpocznie się, musi zostać zarejestrowane. W tym celu jest dostarczony podprogram *isa_register_com()*.

```
uchar isa_register_com
(
uchar com_id          /* identyf. zadania kom. */
)
```

Opis:
Rejestruje nowe zadanie komunikacji w systemie zarządzania pracą wielu implementacji oprogramowania wbudowanego.

wartość zwracana:
0 jeśli wszystko odbywa się pomyślnie, BAD_RET jeśli wystąpią błędy

- Zapamiętywanie kopii zapasowej aplikacji

Zmienna globalna TSK_FUNIT (char *) może wstępnie przyjąć wartość ciągu zawierającego ścieżkę do jednostki pamięci, w której będzie zapisana kopia zapasowa aplikacji. Do tworzenia kopii zapasowej oprogramowanie wbudowane ISaGRAF stosuje po prostu standardowe procedury zarządzania plikami fopen, fread, fwrite, fclose.
Wartością domyślną jest pusty ciąg ("") oznaczający brak jednostki pamięci.

Przykład:
TSK_FUNIT = "*host name:*/C:/ISaGRAF/target/apl/"

Określa jednostkę pamięci ISaGRAF\target\apl\ na dysku C: w komputerze *host_name*. Jest to katalog do zapisania kopii zapasowej aplikacji. Nie wolno zapomnieć o ostatniej kresce ukośnej, gdyż w przeciwnym razie kopia zapasowa zostanie zapisana w katalogu ISaGRAF\target\ a pliki będą miały w nazwie przedrostek apl.

Zmienna ta może zostać ustawiona na różne jednostki pamięci dla poszczególnych implementacji oprogramowania wbudowanego zanim zostaną one poskładane.
Więcej szczegółowych informacji o plikach zapasowych aplikacji podano w rozdziale Własności specjalne; Kopia zapasowa aplikacji.

- Kontrola końca cyklu

Zmienna TSK_NBTCKSCHED (uint 32) może być ustawiona na wartość określającą opóźnienie taktu, jakie jest stosowane przez oprogramowanie wbudowane ISaGRAF na koniec cyklu.
Domyślną wartością jest 0 (tak samo, jak dla szeregowania zadań priorytetowych).
Zmienna ta może zostać ustawiona na wartość inną dla poszczególnych jąder zanim jądra te zostaną poskładane.
Więcej szczegółowych informacji podano w rozdziale Własności specjalne; Czas trwania cyklu, realizacja zadań i priorytety.

- Składanie jądra ISaGRAF

Po dokonaniu konfiguracji ostatni krok polega na skompletowaniu jądra (jąder) ISaGRAF: isa_main.

```
uchar isa_main
(
uchar slave,        /* Numer sterownika        */
char *com           /* NIE UŻYWANY Może pozostać ciąg pusty */
)
```

Opis:
Uruchamia jądro ISaGRAF

wartość zwracana:
będzie to liczba różna od zera jeśli wystąpią błędy

Numer sterownika jest taki sam, jaki omówiono w rozdziale o rejestrowaniu sterownika.
Można uruchomić więcej niż jedno jądro, o ile odpowiadające im numery sterowników są różne.

- Składanie zadania komunikacji ISaGRAF

Po dokonaniu konfiguracji ostatni krok polega na skompletowaniu zadania (zadań) komunikacji) ISaGRAF: tst_main_ex.

```
uchar tst_main_ex
(
char *com,          /* Nazwa urządzenia komunikacyjnego */
uchar *slave,       /* Lokalizacja pola 4 bajtowego określającego sterownik jądra, do którego należy dołączyć zadanie*/
uchar com_id        /* identyfikator zadania komunikacji */
)
```

Opis:
Uruchamia zadanie komunikacji ISaGRAF

wartość zwracana:
będzie to liczba różna od zera jeśli wystąpią błędy

To 4 bajtowe pole określa sterowniki jądra, z którym jest związane zadanie komunikacji. Jeśli potrzeba mniej niż 4 sterowniki, to pole musi zostać uzupełnione zerami. Po uruchomieniu zadania to pole nie jest już dłużej potrzebne.
Nazwa urządzenia komunikacyjnego odpowiada nazwie łącza komunikacyjnego.
Można uruchomić więcej niż jedno zadanie komunikacji, o ile identyfikatory zadań są różne.
Stosując debager pakietu ISaGRAF należy się upewnić, czy ustawione tam parametry łącza komunikacyjnego (patrz Podręcznik użytkownika: Zarządzanie programami) odpowiadają parametrom oprogramowania wbudowanego (zadaniom jądra i komunikacji).

Przykład:
Przykład ten pokazuje sposób uruchomienia:
zadania jądra ISaGRAF o numerze sterownika 1;
zadania komunikacji ISaGRAF oznaczonego identyfikatorem 1, związanego ze sterownikiem jądra o numerze 1 i z urządzeniem /tyCo/1 jako łączem szeregowym;
zadania komunikacji ISaGRAF oznaczonego identyfikatorem 2, związanego ze sterownikiem jądra o numerze 1 o z portem o numerze 1100 jako łączem komunikacyjnym Ethernet.
Bieżącym katalogiem w komputerze jest ten, w którym jest zainstalowane oprogramowanie wbudowane.

załaduj moduł isassr.o
ld < RELS/isassr.o

załaduj moduł isakeret.o (można załadować isakerse.o wtedy, gdy nie jest potrzebne łącze komunikacyjne Ethernet)
ld < CMDS/isakeret.o

konfiguracja łącza do komunikacji szeregowej
ISAMOD ("/tyCo/1", 19200)

częstotliwość zegara systemowego
CLKRATE = sysClkRateGet ()

rejestracja sterownika
isa_register_slave (1)

rejestracja portów komunikacyjnych
isa_register_com (1)
isa_register_com (2)

jednostka pamięci dla plików (można pominąć ponieważ jest ustawiana domyślne)
TSK_FUNIT = ""

kontrola końca cyklu (można pominąć ponieważ jest ustawiana domyślne)
TSK_NBTCKSCHED = 0

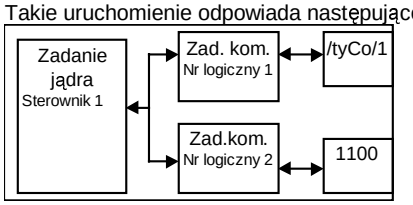
składanie jądra ISaGRAF
sp (isa_main, 1, "")

łącze sterownika, zadania komunikacji

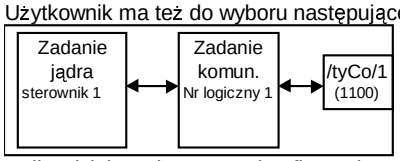
SlavesLink = 0x01000000

składanie zadania komunikacji ISaGRAF
sp (tst_main_ex, "/tyCo/1", &SlavesLink, 1)
sp (tst_main_ex, "1100", &SlavesLink, 2)

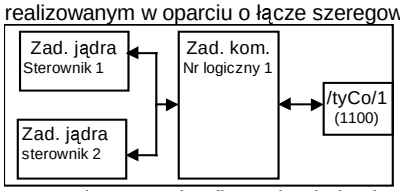
Takie uruchomienie odpowiada następującemu rysunkowi



Użytkownik ma też do wyboru następujące konfiguracje podstawowe.



Najbardziej podstawowa konfiguracja obejmuje jądro związane z zadaniem komunikacji realizowanym w oparciu o łącze szeregowe (Ethernet).

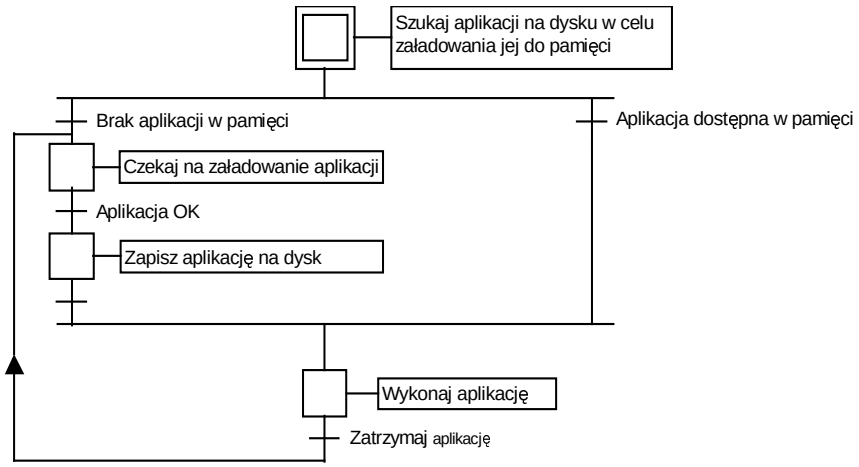


Inna podstawowa konfiguracja obejmuje 2 jądra związane z zadaniem komunikacji w oparciu o łącze szeregowe (Ethernet). W tym przypadku SlavesLink = 0x01020000.

C.5.5 Własności specjalne

Uruchomienie ISaGRAF

Po uruchomieniu oprogramowania wbudowanego jest wykonywany następujący algorytm.



- **Definicje**

Kod aplikacji jest bazą binarnych danych generowanych i ładowanych przez pakiet ISaGRAF a potem wykonywanych przez oprogramowanie wbudowane. Może być uzupełniony tablicą symboli.
Tablica symboli aplikacji jest bazą danych ASCII generowanych i ładowanych przez pakiet ISaGRAF. Jest ona ogniwem łączącym obiekty symboliczne i wewnętrzne obiekty oprogramowania wbudowanego. Jest potrzebna w oprogramowaniu wbudowanym wyłącznie do obsługi określonych symboli użytkownika. Więcej informacji o tablicy symboli podano w Podręczniku użytkownika: Zaawansowane techniki programowania.
Ścieżka do pliku na dysku jest określona w sekwencji uruchomieniowej oprogramowania wbudowanego ISaGRAF przy pomocy globalnej zmiennej TSK_FUNIT (wartość domyślna = "" i oznacza brak jednostki dyskowej)

- **Aplikacje ISaGRAF**

W jednostce centralnej mogą jednocześnie pracować różne aplikacje (zadania jądra i komunikacji), o ile odpowiadają im różne numery sterowników i różne numery logiczne zadań komunikacji. Niemniej jednak wykonując różne aplikacje użytkownik musi ostrożnie postępować z pewnymi powszechnie udostępnianymi obiektami aplikacji, jak karty We/Wy. Na przykład różne aplikacje (jądra) mogą korzystać z odrębnych kart fizycznych o ile w sterowniku We/Wy nie jest wprowadzony pewnego rodzaju serwer lub semafor We/Wy.

- **Kopia zapasowa aplikacji**

Załadowanie nowej aplikacji z debagera pakietu ISaGRAF do oprogramowania wbudowanego spowoduje zapamiętanie (przy pomocy standardowych procedur do zarządzania plikami, jak fopen, itd.) kodu aplikacji pod nazwą:

*ścieżka***ISAx1** Kopia zapasowa kodu aplikacji ISaGRAF
(gdzie x jest numerem sterownika)

Ponadto, jeśli uprzednio została załadowana tablica symboli aplikacji, to zostanie ona także zapamiętana w bieżącym katalogu oprogramowania wbudowanego pod nazwą:

*ścieżka***ISAx6** Kopia zapasowa symboli aplikacji ISaGRAF
(gdzie x jest numerem sterownika)

Ścieżka jest określana w sekwencji uruchomieniowej oprogramowania wbudowanego ISaGRAF przy pomocy zmiennej globalnej TSK_FUNIT. Pusty ciąg ("") będzie oznaczał brak jednostki dyskowej (ustawienie domyślne).

Po uruchomieniu oprogramowania wbudowanego ISaGRAF pliki z kodem aplikacji i symbolami aplikacji są wyszukiwane w bieżącym katalogu i ładowane do pamięci.

Następnie, jeśli w pamięci nie ma tablicy symboli, to sterownik rozpoczyna wykonywanie kodu aplikacji bez żadnych symboli. Jeśli natomiast w pamięci nie ma żadnego kodu, to sterownik czeka na załadowanie aplikacji.

Aby uruchomić oprogramowanie wbudowane z określoną aplikacją zaraz przy włączaniu bez stosowania łącza debagera można zastosować dwa sposoby:

- pierwszy sposób może polegać na bezpośrednim skopiowaniu plików do jednostki pamięci, gdzie ma być zapisana kopia zapasowa aplikacji, z komputera PC, na którym jest zainstalowany pakiet ISaGRAF, przy pomocy narzędzi do przysyłania plików. Dla ułatwienia tych czynności można zastosować menu Narzędzia pakietu ISaGRAF (patrz Podręcznik użytkownika: Zarządzanie programami).

- Drugi sposób może polegać na zapamiętaniu przy pomocy własnych narzędzi kodu aplikacji (a w razie potrzeby także tablicy symboli aplikacji) w pamięci trwałej (takiej, jak PROM lub EPROM) z plików z komputera PC, na którym jest zainstalowany pakiet ISaGRAF.

  Potem, jeśli trzeba (na przykład z powodu szybszego dostępu lub zarządzania pułapkami), można przy uruchamianiu systemu załadować przy pomocy własnych narzędzi kod aplikacji (i jeśli trzeba tablicę symboli aplikacji) z PROM do RAM.
  Następnie podczas uruchamiania ISaGRAF (tuż przed składaniem zadań) można określić adres(y) pamięci, pod którym(i) jest zlokalizowany kod aplikacji (i jeśli trzeba, również tablica symboli aplikacji). Wymaga to ustawienia wartości globalnej zmiennej SSR:
  SSR[x][1].obszar pamięci= *adres lokalizacji kodu aplikacji*

I jeśli trzeba:
SSR[x][6].obszar pamięci = *adres lokalizacji tablicy symboli aplikacji*

Można w tym celu napisać krótką procedurę. Zmienna globalna SSR jest zadeklarowana jako struktura typu str_ssr, która jest zdefiniowana w pliku tasy0ssr.h.

OSTRZEŻENIE:
W debagerze ISaGRAF zarządzanie pułapkami nie będzie wykonywane prawidłowo, jeśli moduł z kodem aplikacji nie jest dostępny do zapisu. Nie jest to problemem, ponieważ aplikacja jest normalnie poddawana gruntownemu uprzedniemu testowaniu.

Jeśli pakiet ISaGRAF jest zainstalowany w standardowym katalogu \ISAWIN, to w komputerze PC:
plikiem zawierającym kod aplikacji projektu MYPROJ jest:
\ISAWIN\APL\MYPROJ\appli.x6m (odpowiadający isax1 w oprogramowaniu wbudowanym),
plikiem zawierającym symbole aplikacji projektu MYPROJ jest:

\ISAWIN\APL\MYPROJ\appli.tst (odpowiadający isax6 w oprogramowaniu wbudowanym).

- Obsługa błędów i komunikaty

Oprogramowanie ISaGRAF target posiada funkcję wykrywania błędów. W załączniku podano listę błędów ostrzegawczych i ich opis.

Wykrywanie błędów odbywa się w następujący sposób:
Na błąd składa się numer błędu i argument wysyłany do standardowej procedury obsługi błędów ISaGRAF
Jeśli w opcjach tworzenia aplikacji w pakiecie ISaGRAF jest ustawiona flaga wykrywania błędów, to błąd zostanie obsłużony. W przeciwnym razie informacje jest tracona i obsługa błędów kończy się.

Podczas obsługi:
Numer błędu (wartość dziesiętna) i argument (wartość szesnastkowa) są wyświetlane na domyślnym wyjściu stdout
Numer błędu i argument są umieszczane w cyklicznym buforze błędów FIFO w celu wykorzystania ich w późniejszym czasie. Rozmiar bufora błędów jest ustawiany w opcjach tworzenia aplikacji w pakiecie. Po zapełnieniu się bufora, każdy nowo wprowadzony błąd kasuje błąd najdłużej przechowywany w buforze.
Błędy mogą być pobierane albo z debagera albo z wykonywanej aplikacji przy użyciu wywołania SYSTEM (patrz Podręcznik użytkownika).

Gdy debager wykryje błąd, to w oknie błędu zostaje wyświetlony komunikat opisujący ten błąd. W zależności od stanu aplikacji (wykonywana czy nie) debager może wyświetlić w kwadratowych nawiasach nazwę obiektu (zmiennej lub programu), w którym ten błąd wystąpił lub numer błędu (wartość dziesiętna) postaci [x], który ma różne znaczenie dla każdego błędu.

Po wykryciu błędu w oprogramowaniu wbudowanym na domyślnym wyjściu stdout są wyświetlane wartości błędu. Dlatego wyświetlaczem można sterować przy użyciu takich procedur standardowych, jak

ioGlobalStdSet()

lub *ioTaskStdSet()*

W tym ostatnim przypadku należy zauważyć, że błędy mogą być generowane albo przez zadania jądra albo zadania komunikacji.

- Czas trwania cyklu, realizacja zadań i priorytety

Pod koniec cyklu ISaGRAF, a przed rozpoczęciem nowego cyklu jest wykonywany następujący algorytm:
Jeśli są określone parametry czasowe cyklu (z pakietu ISaGRAF: patrz Podręcznik użytkownika: Zarządzanie programami), to przez pozostały czas jednostka centralna nie będzie sterowana (określony czas cyklu – czas wykonywania bieżącej aplikacji). Jeśli ten pozostały okres czasu jest ujemny, to zostanie wygenerowane przepełnienie i jednostka centralna nie będzie sterowana przez liczbę taktów określoną w TSK_NBTCKSCHED (zmiennej ustawianej w sekwencji uruchomieniowej ISaGRAF) w celu wymuszenia wykonywania zadań systemowych.

Jeśli parametry czasowe cyklu nie są określone lub jeśli pozostały czas jest mniejszy lub równy 1 taktowi lub jest równy zeru, to jednostka centralna nie będzie sterowana przez liczbę

taktów podanych w TSK_NBTCKSCHED (zmiennej ustawianej w sekwencji uruchomieniowej ISaGRAF), aby wymusić wykonywanie zadań systemowych.

Dokładność parametrów czasowych oprogramowania wbudowanego odpowiada jednemu taktowi systemu VxWorks.

Taktowanie z określoną częstotliwością jest powszechnie stosowane do wyzwalania cykli lub do wykorzystania jednostki centralnej do innych zadań pracujących pod systemem OS-9.

Jeśli łączem komunikacyjnym nie są przesyłane żadne dane, to zadanie komunikacji jest w stanie czuwania. W odpowiedniej chwili za pośrednictwem protokołu typu pytanie / odpowiedź zadanie otrzymuje z jądra informacje o pracującej aplikacji. Zadanie komunikacja wysyła do jądra pytanie. Pod koniec cyklu jądro przekazuje odpowiedź do zadania komunikacji (aby mieć zsynchronizowany obraz aplikacji).

Zadania ISaGRAF nie modyfikują nadanego im priorytetu. Użytkownik ma swobodę nadawania priorytetów w zależności od opisanego powyżej sposobu realizacji zadania ISaGRAF i ogólnych wymagań aplikacji

C.6 Uruchomienie oprogramowania wbudowanego ISaGRAF pod Windows NT

C.6.1 Wykonywanie ISaGRAF

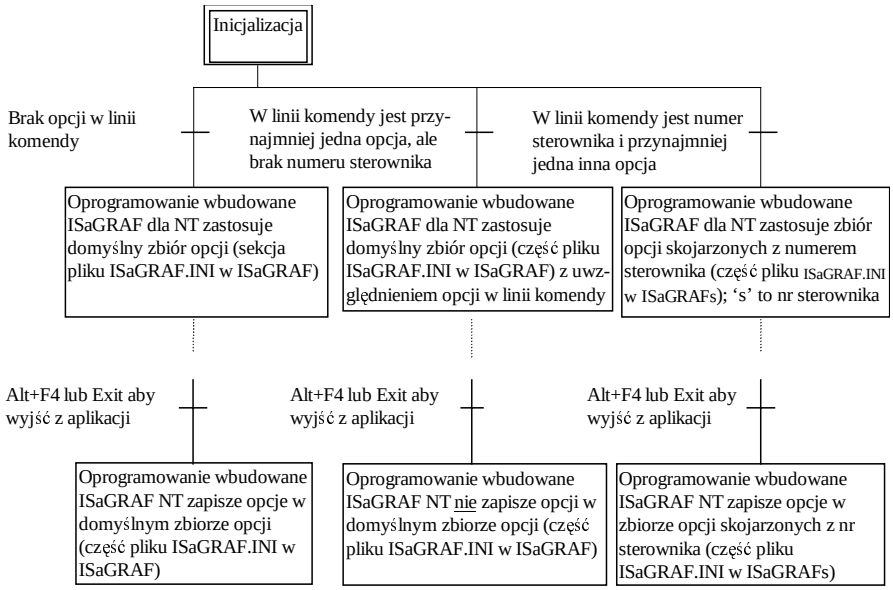
Pod Windows NT oprogramowanie wbudowane pracuje jako pojedynczy program: WISAKER.EXE, który może być uruchamiany kilka razy. Dzięki temu użytkownik może wykonywać jednocześnie dowolną ilość implementacji oprogramowania ISaGRAF NT, ponieważ każdy sterownik ma swój numer.

Oprogramowanie wbudowane umożliwia pracę programów sterowanych przerwaniami.

Oprogramowanie WISAKER jest przeznaczone do pracy pod Windows NT 3.51 lub późniejszą wersją.

C.6.2 Ogólne informacje o opcjach

Opcje są zapamiętywane i wyszukiwane zgodnie z następującym schematem:



Należy zwrócić uwagę, że plik ISAGRAF.INI jest zapamiętany w bieżącym katalogu roboczym.

Numer sterownika: opcja -s

Opcja ta określa numer sterownika oprogramowania wbudowanego. Może to być liczba z przedziału 1 do 255 oprócz liczby 13 ($0D). Numer sterownika jest używany w protokole komunikacyjnym. Służy głownie do odróżniania od siebie poszczególnych sterowników, gdy z pakietem ISaGRAF jest związane więcej niż jedno oprogramowanie wbudowane lub gdy na danym komputerze pracuje więcej niż jedno oprogramowanie wbudowane. Korzystając z debagera pakietu ISaGRAF należy się upewnić, czy ustawiony w pakiecie ISaGRAF numer sterownika odpowiada ustawieniu w oprogramowaniu wbudowanym (patrz Podręcznik użytkownika: Zarządzanie programami).

Wartość domyślna: Domyślnym numerem sterownika jest 1 lub numer podany w pliku ISaGRAF.INI.

Przykład:
WISAKER.EXE -s=2

Interfejs użytkownika: Poniższe okno jest wyświetlane przez komendę "**Options/Slave number**" z głównego okna oprogramowania wbudowanego ISaGRAF NT.

Podaną w tej opcji wartość można zmienić przy użyciu myszy lub klawiszy ze strzałkami (w górę i w dół). Po dokonaniu zmiany należy ponownie uruchomić oprogramowanie wbudowane ISaGRAF NT.

Łącze komunikacyjne i konfiguracja: opcja -t

Do komunikowania się z debagerem oprogramowanie wbudowane wykorzystuje łącze szeregowe lub łącze Ethernet. Nazwa portu jest określana opcją -t. Ponieważ interfejs komunikacyjny jest przeznaczony do pracy na każdym urządzeniu, więc do komunikacji szeregowej mogą być użyte porty COM1, COM2, COM3 lub COM4, a do komunikacji poprzez Ethernet porty o numerach zaczynających się od 1100.

Wartość domyślna: Domyślnym portem komunikacyjnym jest port o numerze 1100 dla Ethernet i COM1 dla komunikacji szeregowej lub port określony w pliku ISaGRAF.INI.

UWAGA: Domyślnym łączem komunikacyjnym jest Ethernet.

Przykłady:
WISAKER -t=COM2
WISAKER -t=1101

Konfiguracja szeregowa:
Niektóre opcje mogą być stosowane tylko wtedy, gdy jest określona opcja -t=COMx.

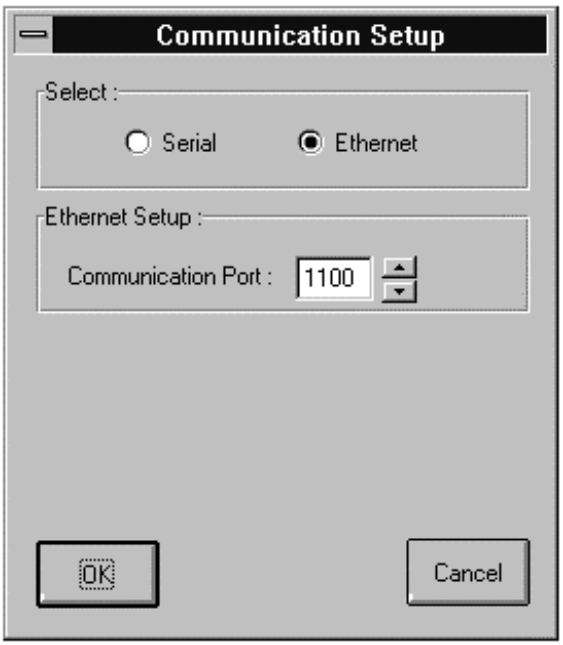
Poniżej są podane opcje konfiguracyjne dla łącza szeregowego:

Opcja	Wartości	Znaczenie
bod	600 1200 2400 4800 9600 **19200**	Szybkość transmisji
parzystość	**n** e o	Brak kontroli parzystości Parzystość Nieparzystość
dane	7 lub **8**	Liczba bitów
stop	**1** lub 2	Długość bitu stopu
przepływ	h **n**	Sterowanie sprzętowe Brak kontroli

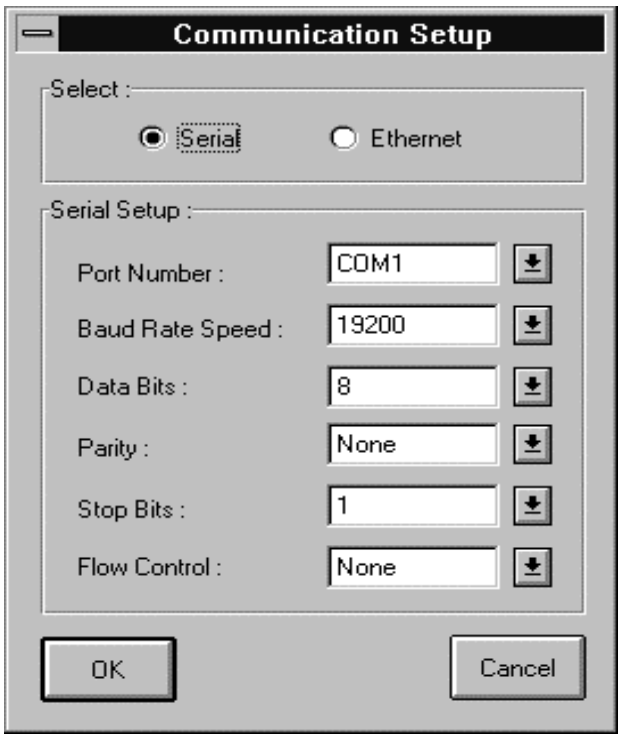
Wartościami domyślnymi są 19200 bodów, brak kontroli parzystości, 8 bitów danych, 1 bit stopu, brak kontroli przepływowej

Przykład:
WISAKER -t=COM1 baud=1200 data=8 parity=n stop=2

Interfejs użytkownika: Poniższe okno jest wyświetlane po komendzie "**Options/Slave number**" z głównego okna oprogramowania wbudowanego ISaGRAF NT.



Można wybrać komunikację szeregową lub komunikację poprzez Ethernet. Komunikacja poprzez Ethernet daje możliwość modyfikowania numeru portu. Numer portu musi być taki sam, jak w specyfikacji Parametry połączenia PC--PLC w pakiecie ISaGRAF.



Po wybraniu komunikacji szeregowej pojawi się konfiguracja. Konfiguracja ta powinna być taka sama, jak w specyfikacji Parametry połączenia PC--PLC w pakiecie ISaGRAF.

Graficzna symulacja kart wirtualnych: opcja -x

Po ustawieniu tej opcji będą symulowane karty zadeklarowane jako wirtualne w edytorze połączeń We/Wy (patrz część A).
Możliwe są dwie wartości: 0 lub 1; 0 oznacza brak symulacji a 1 oznacza włączenie symulacji.

Wartość domyślna: Wartością domyślną jest 0 lub wartość podana w pliku ISaGRAF.INI.

Przykład:
WISAKER -x=1 będzie symulować karty wirtualne,

Interfejs użytkownika: Ta pozycja menu będzie zaznaczana lub odznaczana odzwierciedlając stan opcji. Symulowane karty pojawią się na panelu graficznym.

Priorytet dla oprogramowania wbudowanego ISaGRAF NT: opcja -p

Ponieważ oprogramowanie wbudowane pracuje pod NT, bardzo przydatne jest określenie poziomu priorytetu. Możliwe jest na przykład wykonywanie czasowo krytycznej aplikacji

ISaGRAF pod oprogramowaniem wbudowanym o wyższym priorytecie i równoczesna praca w tle jednej lub kilku implementacji tego oprogramowania wbudowanego o niższych priorytetach.

Możliwymi wartościami są 0, 1, 2 lub 3. 0 oznacza priorytet najwyższy a 3 najniższy.

Przykłady:
WISAKER -p=0
WISAKER -p=1

Interfejs użytkownika: Poniższe okno jest wyświetlane po komendzie "**Options/Priority**" z głównego okna oprogramowania wbudowanego ISaGRAF NT.

Najwyższy priorytet oznacza wykonanie w czasie rzeczywistym a najniższy oznacza wykonanie przy braku obciążenia.
0: Wykonywanie w czasie rzeczywistym
1: Priorytet wysoki
2: Priorytet normalny
3: Wykonanie przy braku obciążenia

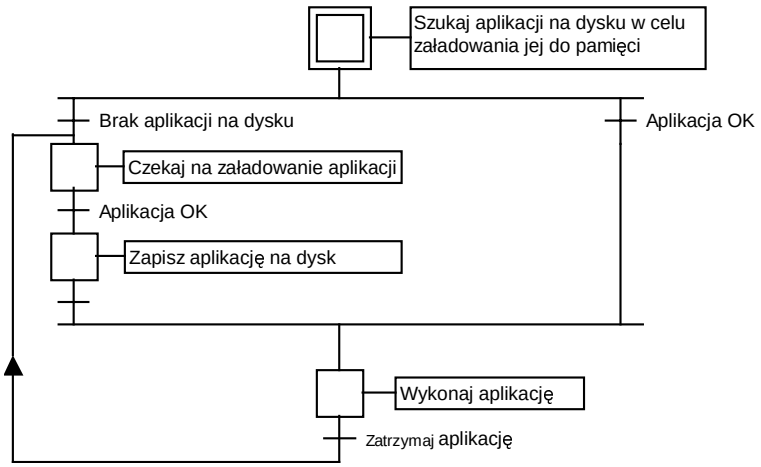
Przykłady:

wisaker -t=COM1 Uruchamia oprogramowanie wbudowane ISaGRAF o domyślnym numerze sterownika (1) i z COM1 jako portem komunikacyjnym.
wisaker -s=3 -t=COM1 Uruchamia oprogramowanie wbudowane ISaGRAF o numerze sterownika 3 i z COM1 jako portem komunikacyjnym.

C.6.3 Własności specjalne

Uruchomienie ISaGRAF

Po uruchomieniu oprogramowania wbudowanego jest wykonywany następujący algorytm.



- **Definicje**

Kod aplikacji jest bazą binarnych danych generowanych i ładowanych przez pakiet ISaGRAf a potem wykonywanych przez oprogramowanie wbudowane. Może być uzupełniony tablicą symboli.
Tablica symboli aplikacji jest bazą danych ASCII generowanych i ładowanych przez pakiet ISaGRAF. Jest ona ogniwem łączącym obiekty symboliczne i wewnętrzne obiekty oprogramowania wbudowanego. Jest potrzebna w oprogramowaniu wbudowanym wyłącznie do obsługi określonych funkcji użytkownika, jak na przykład własności DDE lub symulacji We/Wy przy pomocy funkcji nazw symboli. Więcej informacji o tablicy symboli podano w Podręczniku użytkownika: Zaawansowane techniki programowania.

- **Aplikacje ISaGRAF**

W jednostce centralnej mogą jednocześnie pracować różne aplikacje, o ile odpowiadają im różne numery sterowników i różne numery logiczne zadań komunikacji. Niemniej jednak wykonując różne aplikacje użytkownik musi ostrożnie postępować z pewnymi powszechnie udostępnianymi obiektami aplikacji, jak karty We/Wy. Na przykład różne aplikacje mogą korzystać z odrębnych kart fizycznych o ile w sterowniku We/Wy nie jest wprowadzony pewnego rodzaju serwer lub semafor We/Wy.

- **Kopia zapasowa aplikacji**

Ładowanie nowej aplikacji z debagera pakietu ISaGRAF do oprogramowania wbudowanego spowoduje zapamiętanie kodu aplikacji w bieżącym katalogu oprogramowania pod nazwą:

ISAx1 Kopia zapasowa kodu aplikacji ISaGRAF
(gdzie x jest numerem sterownika)

Ponadto, jeśli uprzednio została załadowana tablica symboli aplikacji, to zostanie ona także zapamiętana w bieżącym katalogu pod nazwą:

ISAx6 Kopia zapasowa symboli aplikacji ISaGRAF
(gdzie x jest numerem sterownika)

Po uruchomieniu oprogramowania wbudowanego ISaGRAF pliki z kodem aplikacji i symbolami aplikacji są wyszukiwane w bieżącym katalogu i ładowane do pamięci. Jeśli nie jest dostępny żaden plik z symbolami, to sterownik rozpoczyna wykonywanie kodu aplikacji bez ładowania symboli.
Jeśli natomiast nie jest dostępny żaden kod, to sterownik czeka na załadowanie aplikacji.

Aby uruchomić oprogramowanie wbudowane z określoną aplikacją, bez stosowania łącza debagera można te pliki bezpośrednio skopiować na dysk, na którym znajduje się bieżący katalog sterownika z tego samego dysku, jeśli pakiet ISaGRAF pracuje na tym samym komputerze lub skorzystać z dysku elastycznego.

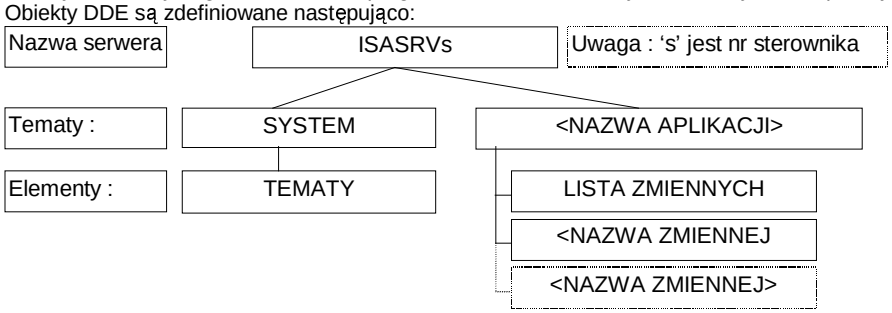
Jeśli pakiet ISaGRAF jest zainstalowany w standardowym katalogu \ISAWIN, to plikiem aplikacji projektu MYPROJ jest:
\ISAWIN\APL\MYPROJ\appli.x8m
a plikiem z symbolami aplikacji projektu MYPROJ jest:
\ISAWIN\APL\MYPROJ\appli.tst

Przykład:
Jeśli z katalogu, w którym jest zainstalowany WISAKER.EXE zostanie podana następująca komenda:
copy \ISAWIN\APL\MYPROJ\appli.x8m isa11
to WISAKER.EXE znajdzie i wykona aplikację 'myproj'.

Wszystkie te komendy mogą zostać pogrupowane na przykład w pliku wsadowym a potem uruchomione z menu Narzędzia pakietu (patrz Podręcznik użytkownika: Zarządzanie programami).

- Opis techniczny DDE

Oprogramowanie wbudowane ISaGRAF NT jest serwerem DDE (Dynamic Data Exchange). Każde oprogramowanie, które może pełnić rolę klienta może zostać połączone z oprogramowaniem wbudowanym w celu wymiany zmiennych. Na przykład MSEXCEL może animować grafikę przy użyciu wartości pochodzących z oprogramowania wbudowanego ISaGRAF za pośrednictwem DDE.
Funkcja DDE wymaga istnienia w oprogramowaniu wbudowanym tabeli symboli aplikacji. Obiekty DDE są zdefiniowane następująco:



« ISASRVs » jest nazwą serwera DDE, 's' jest numerem sterownika.
« SYSTEM » jest standardowym tematem zapewniającym dostęp do elementu « TEMATY »,
« TEMATY » podaje listę aktualnie zdefiniowanych tematów: systemu i nazwy aplikacji pracującej pod oprogramowaniem wbudowanym ISaGRAF NT.
« NAZWA APLIKACJI » jest nazwą aplikacji.

« LISTA ZMIENNYCH » jest listą elementów dostępnych w aktualnym temacie, jest to lista zmiennych, dostęp do których jest możliwy za pośrednictwem DDE.
« NAZWA ZMIENNEJ » jest nazwą zmiennej.

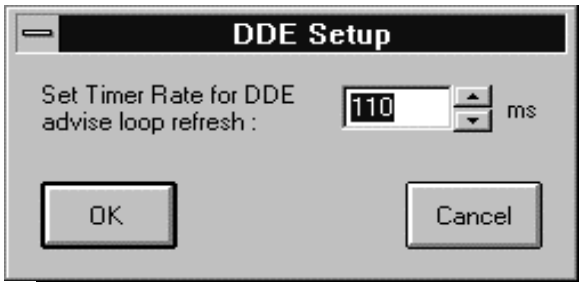
Tempo wymiany informacji poprzez tryb ADVISE LOOP DDE dla oprogramowania wbudowanego ISaGRAF NT: opcja -d
Serwer DDE generalnie pobiera zmienne za każdym razem, gdy je potrzebuje. Może to pochłaniać dużo czasu, jeśli zmiennych jest dużo. Jest natomiast inny tryb nazywany trybem ADVISE LOOP, w którym sam serwer wysyła tylko zmodyfikowane zmienne. W ten sposób komunikacja jest skrócona do minimum i jest efektywna. W tym trybie serwer okresowo przegląda zmienne oznaczone jako zmienione, aby wiedzieć, które powinny zostać wysłane. Okres ten jest określany tempem wymiany informacji poprzez ADVISE LOOP DDE.

Przy użyciu tej opcji można określić szybkość wymiany informacji (w ms) dla ADVISE LOOP DDE.

Wartość domyślna: Wartością domyślną jest 1000 ms lub wartość podana w pliku ISaGRAF.INI

Przykład:
WISAKER -d=100

Interfejs użytkownika: Poniższe okno jest wyświetlane po komendzie "**Options/DDE**" podanej w głównym oknie oprogramowania wbudowanego ISaGRAF NT.



- Obsługa błędów i komunikaty

Oprogramowanie wbudowane ISaGRAF posiada funkcję wykrywania błędów. W załączniku podano listę błędów ostrzegawczych i ich opis.

Wykrywanie błędów odbywa się w następujący sposób:
Na błąd składa się numer błędu i argument wysyłany do standardowej procedury obsługi błędów ISaGRAF
Jeśli w opcjach kompilatora pakietu ISaGRAF jest ustawiona flaga wykrywania błędów, to błąd zostanie obsłużony. W przeciwnym razie informacje jest tracona i obsługa błędów kończy się.

Podczas obsługi:
Numer błędu (wartość dziesiętna) i argument (wartość szesnastkowa) są wyświetlane na wyjściu (w oknie WISAKER.EXE)

Numer błędu i argument są umieszczane w cyklicznym buforze błędów według zasady FIFO do wykorzystania ich w późniejszym czasie. Rozmiar bufora błędów jest ustawiany w opcjach 'Utworzenia kodu aplikacji' pakietu ISaGRAF. Po zapełnieniu się bufora, każdy nowo wprowadzony błąd kasuje błąd najdłużej przechowywany w buforze.
Błędy mogą być wybierać albo z debagera albo z wykonywanej aplikacji przy użyciu wywołania SYSTEM (patrz Podręcznik użytkownika).

Gdy debager wykryje błąd, to w oknie błędu zostaje wyświetlony komunikat opisujący ten błąd. W zależności od stanu aplikacji (wykonywana czy nie) debager może wyświetlić w kwadratowych nawiasach nazwę obiektu (zmiennej lub programu), w którym ten błąd wystąpił lub numer błędu (wartość dziesiętna) postaci [x], który ma różne znaczenie dla każdego błędu.

Po uruchomieniu oprogramowania wbudowanego na wyjściu jest wyświetlany komunikat powitalny. Zawiera on numer sterownika, konfigurację transmisji i nazwę serwera DDE.

- Zegar systemowy

Ponieważ oprogramowanie wbudowane ISaGRAF jest przeznaczone do pracy pod każdym systemem, to wzorcem czasu służącym zarówno do synchronizacji cyklu jak i do zmiany zmiennych typu timer jest standardowy takt wynoszący około 10 milisekund.

Dlatego nie jest możliwe uzyskanie wyższej dokładności zmiennych timera niż 10 ms. Z tego samego powodu określony czas trwania cyklu, który jest mniejszy lub równy 10 ms i różny od zera wygeneruje przepełnienie czasu cyklu (błąd 62). Więcej informacji podano w następnym rozdziale.

Jeśli stosowana przez użytkownika aplikacja wymaga większej dokładności, należy poprosić dostawcę o wersję specjalną.

- Czas trwania cyklu a zachowanie się oprogramowania wbudowanego

Pod koniec jednego cyklu ISaGRAF, a przed rozpoczęciem następnego jest wykonywany następujący algorytm:
Jeśli są określone parametry czasowe cyklu (z pakietu ISaGRAF: patrz Podręcznik użytkownika: Zarządzanie programami), to przez pozostały czas jednostka centralna nie będzie sterowana (określony czas cyklu – czas wykonywania bieżącej aplikacji). Jeśli ten pozostały okres czasu jest ujemny, to zostanie wygenerowane przepełnienie i jednostka centralna nie będzie sterowana przez 1 takt w celu wymuszenia wykonywania zadań systemowych.

Jeśli taktowanie cyklu nie jest określone lub jeśli pozostały czas jest mniejszy lub równy 1 taktowi lub jest równy zeru, to jednostka centralna nie będzie sterowana przez 1 takt, aby wymusić wykonywanie zadań systemowych.

Dokładność taktowania oprogramowania wbudowanego odpowiada jednemu taktowi systemu Windows NT.

Taktowanie z określoną częstotliwością jest powszechnie stosowane do wyzwalania cykli lub do wykorzystania jednostki centralnej do innych zadań pracujących pod systemem.

- Przyciski wyjścia

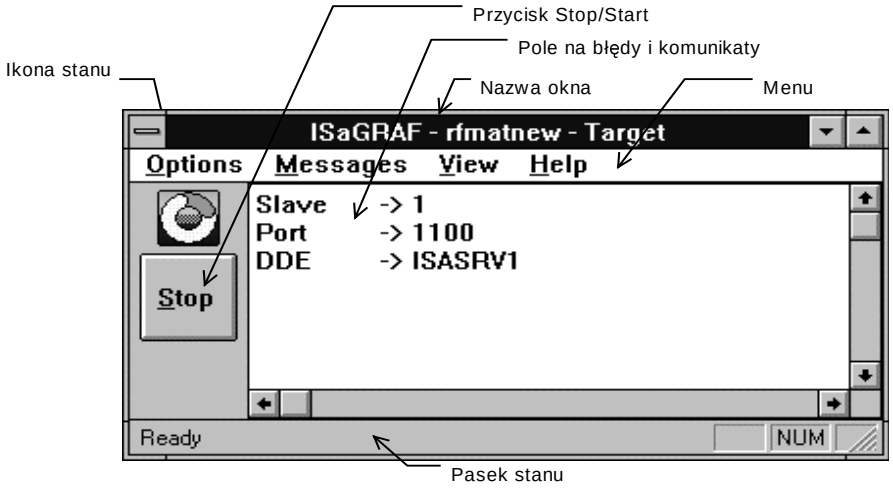
Testując aplikację w warunkach nieprzemysłowych na komputerze osobistym użytkownik może chcieć zatrzymać ISaGRAF: odbywa się to przez naciśnięcie kombinacji przycisków, która ma zapobiec niespodziewanemu zatrzymaniu. Tą sekwencją przycisków jest:

alt + F4

Pewnym niebezpiecznym efektem ubocznym takich szybkich przerwań jest brak zamknięcia interfejsu karty We/Wy. Dlatego właściwym sposobem zatrzymania oprogramowania wbudowanego ISaGRAF jest:
- zatrzymanie aplikacji z debagera lub przez naciśnięcie przycisku Start/Stop (spowoduje to zamknięcie kart We/Wy)
zatrzymanie oprogramowania wbudowanego ISaGRAF z menu systemowego.

C.6.4 Interfejs użytkownika

Poniżej przedstawiono interfejs użytkownika oprogramowania wbudowanego ISaGRAF NT:



Składa się z następujących głównych elementów:
- Nazwy okna
- Paska menu
- Ikony stanu realizacji
- Przycisku Start/Stop
- Pola na wyświetlanie błędów i komunikatów, i
- Paska stanu.

Nazwa okna zawiera « ISaGRAF - name_of_appli - target », gdzie name_of_appli jest nazwą bieżącej aplikacji. Gdy nie jest wykonywana żadna aplikacja, to nazwa ta składa się tylko z « ISaGRAF - - Target ».

Pasek menu oprogramowania wbudowanego ISaGRAF NT:

Pasek menu składa się z czterech menu:
- Options
- Messages
- View (Widok)

Help (Pomoc)

- **Menu "Options"**

(patrz też rozdział pierwszy: Ogólne informacje o opcjach)
Menu "**Options**" zapewnia dostęp do bieżących opcji. Są proponowane następujące opcje:

Slave number umożliwia modyfikowanie numeru sterownika. Modyfikacja opcji zostanie zrealizowana dopiero po następnym uruchomieniu oprogramowania wbudowanego. Jeśli oprogramowanie zostało uruchomione z przynajmniej jedną opcją w linii komendy, to z tej funkcji nie można korzystać.

Communication umożliwia konfigurowanie parametrów transmisji. Modyfikacja opcji zostanie zrealizowana dopiero po następnym uruchomieniu oprogramowania wbudowanego. Jeśli oprogramowanie zostało uruchomione z przynajmniej jedną opcją różną od –s, to z tej funkcji nie można korzystać.

DDE umożliwia modyfikowanie tempa wymiany informacji poprzez DDE . Modyfikacja opcji zostanie zrealizowana dopiero po następnym uruchomieniu oprogramowania wbudowanego. Jeśli oprogramowanie zostało uruchomione z przynajmniej jedną opcją różną od –s, to z tej funkcji nie można korzystać.

I/O simulation (Symulacja We/Wy) jest zaznaczona lub odznaczona odzwierciedlając stan opcji. Modyfikacja opcji zostanie zrealizowana dopiero po zatrzymaniu i uruchomieniu aplikacji.

Priority (Priorytet) umożliwia modyfikowanie priorytetów. Modyfikacja opcji jest zrealizowana natychmiast.

Default settings (Opcje domyślne) wyszukuje domyślne ustawienia następujących bieżących opcji:
- komunikacja, - DDE, - współrzędne okna na ekranie.
Zmodyfikowane opcje zostaną uaktywnione dopiero pod ponownym uruchomieniu oprogramowania wbudowanego. Jeśli oprogramowanie zostało uruchomione z przynajmniej jedną opcją różną od –s, to z tej funkcji nie można korzystać.

- **Menu "Messages" (Komunikaty)**

Menu "**Komunikaty**" służy do obsługi wyprowadzanych wiadomości. Zawiera następujące dwie komendy:

Acnowledge (Potwierdź), która zatrzymuje migającą czerwoną lampkę w przypadku wystąpienia błędu lub pojawienia się komunikatu.

Clear (Wyczyść), która całkowicie usuwa wyprowadzoną wiadomość

Ikona stanu oprogramowania wbudowanego ISaGRAF NT:
Ikona odzwierciedla stan oprogramowania wbudowanego:
– jeśli aplikacja jest wykonywana, to ikona obraca się
– brak aplikacji (lub aplikacja zatrzymana): ikona jest nieruchoma
– w oknie wyjściowym są informacje o błędach lub komunikaty. Środek ikony miga na czerwono. Aby zatrzymać miganie można wybrać « Acnowledge » (« Potwierdź») lub « Clear » (« Wyczyść ») w menu « Messages » (« Komunikaty ») (należy jednak pamiętać, że

wybranie « Wyczyść » całkowicie wyzeruje zawartość okna). Więcej informacji o błędach podano w rozdziale Obsługa błędów i komunikaty.

Stany oprogramowania wbudowanego są zebrane w następującej tabeli:

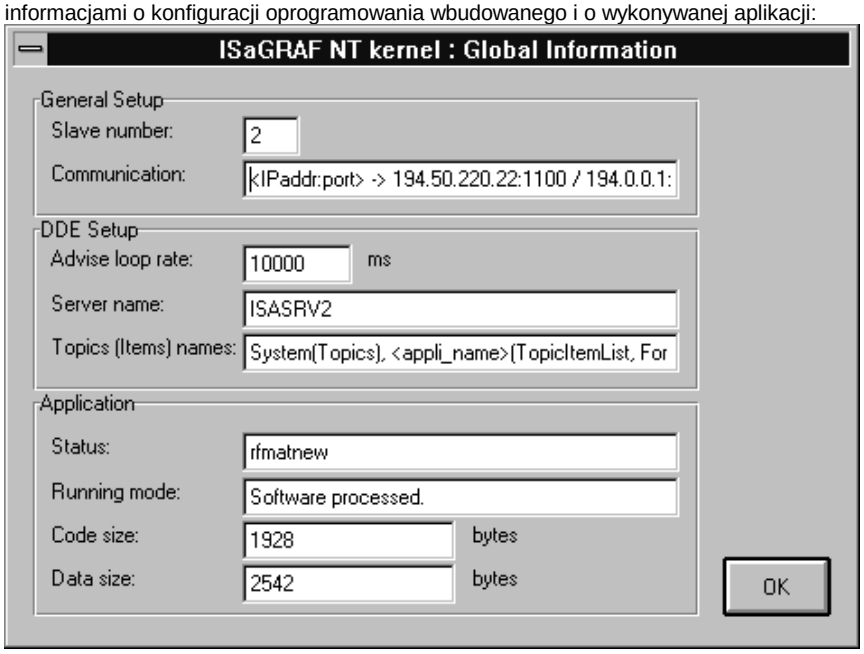
	Brak błędu	**Błędy lub komunikaty (środek ma kolor czerwony)**
Aplikacja jest wykonywana		
Brak aplikacji		

- Przycisk Start/Stop oprogramowania wbudowanego ISaGRAF NT:

Przycisk Start/Stop jest taki sam, jak funkcja start/stop debagera. Tekst pod przyciskiem odzwierciedla bieżący stan aplikacji. Jeśli aplikacja jest aktualnie wykonywana, to tekst będzie brzmiał « Stop », a jeśli aplikacja jest zatrzymana (lub w przypadku braku aplikacji), tekst będzie brzmiał « Run » (« Uruchom ») (należy zwrócić uwagę, że jeśli w przypadku braku aplikacji wystąpi żądanie Start, to przycisk Start/Stop przełączy się do ustawienia Stop a potem ponownie powróci do ustawienia « Run » (« Uruchom »)).

- Oprogramowanie wbudowane ISaGRAF NT – informacje ogólne

Po wybraniu komendy "View / Information" pojawi się następujące okno dialogowe z ogólnymi informacjami o konfiguracji oprogramowania wbudowanego i o wykonywanej aplikacji:



Składa się on z trzech części:

a) General setup (Ustawiania ogólne):

- Numer sterownika

- Komunikacja – ustawienia (Jeśli łączem komunikacyjnym jest łącze Ethernet, to oprócz numeru portu jest też wyświetlana lista dostępnych adresów IP w aktualnym systemie NT)

b) DDE setup (Ustawienia DDE) :

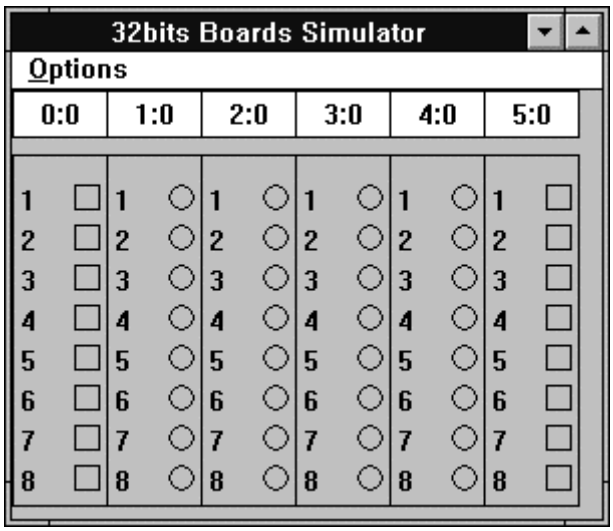
- Tempo wymiany informacji w trybie ADVISE LOOP
- Nazwa serwera DDE
- Nazwy tematów i pozycji DDE. Są to informacje ogólne, które nie odzwierciedlają wartości rzeczywistych. W rzeczywistości pola pomiędzy < > powinny zostać zastąpione wartościami rzeczywistymi.

c) Application (Aplikacja) :

- Status aplikacji, który jest wskazywany nazwą aplikacji, gdy jakaś aplikacja jest wykonywana lub zwrotem 'No application' ('Brak aplikacji') gdy nie jest wykonywania żadna aplikacja
- Tryb pracy aplikacji, który wskazuje, czy aplikacja jest wykonywana przez procesor. W tym przypadku na ekranie podany jest ciąg: « Software processed ». Jeśli aplikacja została skompilowana przy pomocy kompilatora języka C, to na tym polu pojawi się ciąg: « C Compiled». Jeśli natomiast nie jest wykonywana żadna aplikacja, będzie widoczny ciąg: «No application ».
- Wielkość kodu w bajtach. Jeśli trybem wykonywania aplikacji jest « C Compiled », to na tym polu widnieje zero.
- Rozmiar danych w bajtach. Jest to suma wewnętrznych danych używanych podczas przebiegu i danych w bazie zmiennych.

Symulacja kart wirtualnych przez oprogramowanie wbudowane ISaGRAF NT:

Wybranie opcji « I/O Simulation » spowoduje pojawienie się następującego okna po następnym uruchomieniu aplikacji:



W zależności od konfiguracji połączenia We/Wy będzie mniej lub więcej różnych kart i mniej lub więcej różnych zmiennych. Liczby « s :b »nad każdą kolumną karty stanowią identyfikator

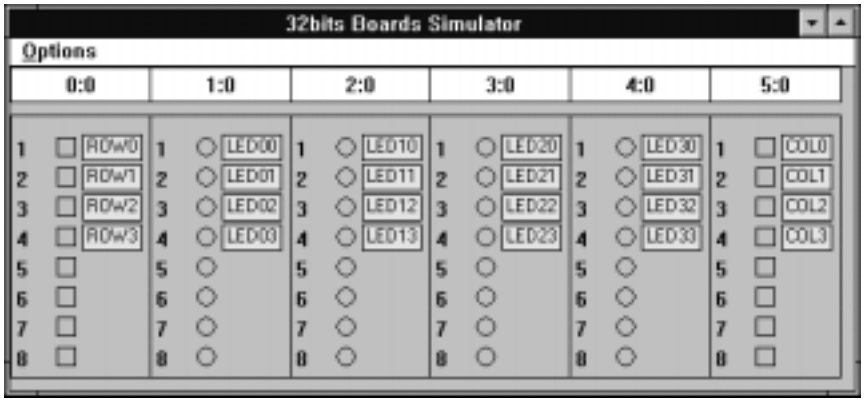
gniazda (s) i identyfikator karty (b). Zliczanie rozpoczyna się od zera i nie ma możliwości zmiany.
Okno ‘Symulator karty 32 bitowej’ współpracuje z aplikacją w trybie Start/Stop. A więc, okno pojawia się wtedy, gdy jest wykonywana aplikacja, w której występują karty wirtualne (lub która wykorzystuje karty symulowane) i jest ustawiona flaga « I/O Simulation ». Okno to zostanie zamknięte natychmiast po naciśnięciu przycisku « Stop ». Okno to działa razem z wywołaniami We/Wy.

Menu "**Options**" zawiera dwa elementy:

Element **Nazwy zmiennych** pokaże nazwy zmiennych tylko wtedy, gdy przed kodem aplikacji została załadowana tablica symboli.

Element **Wartości szesnastkowe** pokaże każdą liczbę typu integer w formacie szesnastkowym zamiast w domyślnym formacie dziesiętnym

Nazwy zmiennych będą wyglądać następująco:



C.7 Programowanie w języku "C"

C.7.1 Wprowadzenie

Podręcznik jest przeznaczony dla użytkowników już posiadających doświadczenia w korzystaniu z ISaGRAF i narzędzi tego pakietu. Po opracowaniu aplikacji dotyczących czystej automatyki przy użyciu **funkcji konwersji**, **funkcji i bloków funkcyjnych "C"** ze standardowych bibliotek CJ International możliwe jest opracowanie "zdefiniowanych przez użytkownika" funkcji konwersji, funkcji "C" i bloków funkcyjnych. Dzięki temu użytkownik może wzbogacić PLC z oprogramowaniem wbudowanym ISaGRAF przez tworzenie nowych bibliotek i czerpać maksymalne korzyści z rozbudowanych możliwości stacji roboczej i platformy sprzętowej.

W powiązaniu z systemem opracowywania aplikacji w języku "C" i dotychczasowymi doświadczeniami w programowaniu w języku "C", ten podręcznik umożliwi użytkownikowi przystosowanie PLC z oprogramowaniem wbudowanym ISaGRAF do sprawowania najbardziej efektywnej kontroli. Nie tylko zwiększy się wydajność PLC, ale także wzrośnie komfort programowania automatyki i jakość produktów powstałych przy użyciu pakietu ISaGRAF.

Informacje zawarte w tym dokumencie nie dotyczą tylko jednego specjalnego systemu oprogramowania wbudowanego. Jednak niektóre własności (takie, jak wielozadaniowość) nie mogą być stosowane w pewnych systemach jednozadaniowych.

Standardowe własności pakietu ISaGRAF

Pakiet ISaGRAF oferuje wiele funkcji pozwalających na zarządzanie bibliotekami komponentów "C" do opracowywania automatyki. Dla potrzeb programowania automatyki funkcja konwersji, funkcja "C" lub blok funkcyjny stanowi "**czarną skrzynkę**", która jest całkowicie zdefiniowana przez swój interfejs.

Program Biblioteki ISaGRAF służy do rozbudowywania istniejących bibliotek o nowe komponenty i definiowania interfejsu pomiędzy implementacją w języku "C" a stosowaniem tych komponentów do programowania w języku **ST**/**FBD**. Menedżer Bibliotek ISaGRAF zapewnia też automatyczne generowanie ramki kodu źródłowego w języku "C" dla konwersji, funkcji i bloków funkcyjnych oraz obejmuje narzędzia do edytowania takich plików źródłowych w języku "C". Dalsze informacje o funkcjach programu Biblioteki podano w **Podręczniku użytkownika ISaGRAF**.

Opracowywanie aplikacji w języku "C"

Pakiet ISaGRAF nie obejmuje żadnego kompilatora języka "C" ani narzędzia do kompilacji dedykowanego na inne środowisko. Użytkownik musi posiadać własny kompilator języka "C" dla systemu oprogramowania wbudowanego ISaGRAF w celu zintegrowania swoich komponentów w języku "C" z jądrem ISaGRAF.

Dla potrzeb kompilatora dedykowanego na inne środowisko pakiet ISaGRAF oferuje możliwość wejść zewnętrznych dla użytkownika, pozwalających na wykonywanie w oknie

DOS zdefiniowanego przez użytkownika pliku komend MS-DOS (.bat). Stosowany kompilator musi pracować w oknie emulacji DOS. W przeciwnym wypadku trzeba zamknąć Windows, aby kompilatory i konsolidatory zostały uruchomione w środowisku czysto MS-DOS.

Opisy techniczne

Program Biblioteki ISaGRAF pozwala użytkownikowi na sporządzenie opisu tekstowego dla każdego komponentu biblioteki. Taki **opis techniczny** jest przewodnikiem użytkownika po opracowanym komponencie w języku "C" i służy programiście do opisywania w aplikacjach ISaGRAF zawartych w tym komponencie funkcji: konwersji, funkcji lub bloków funkcyjnych.

Konwersja, funkcja lub blok funkcyjny "C" muszą być precyzyjnie zdefiniowane w opisie technicznym, aby programista automatyki mógł faktycznie wykorzystać je w formie pakietowej funkcji ISaGRAF. W odniesieniu do funkcji "C" opis techniczny musi zawierać:

- ❑ szczegółowy opis działania danej funkcji
- ❑ kompletny opis parametrów wywołania
- ❑ znaczenie wartości zwracanej tej funkcji
- ❑ dokładną pisownię parametrów wywołania i wartości zwracanej
- ❑ środowisko wykonywania aplikacji

W odniesieniu do bloku funkcyjnego "C" opis techniczny musi zawierać:

- ❑ szczegółowy opis działania danego bloku funkcyjnego
- ❑ kompletny opis parametrów wywołania
- ❑ znaczenie parametrów zwracanych
- ❑ dokładną pisownię parametrów wywołania i zwracanych
- ❑ środowisko wykonywani aplikacji

W odniesieniu do funkcji konwersji opis techniczny musi zawierać:

- ❑ dokładne znaczenie konwersji w odniesieniu do zmiennej wejściowej
- ❑ dokładne znaczenie konwersji w odniesieniu do zmiennej wyjściowej
- ❑ limity wartości, jakie mogą być przedmiotem konwersji

Opisy techniczne mogą też zawierać informacje określające:

- ❑ kompletny identyfikator konwersji, funkcji lub bloku funkcyjnego
- ❑ informacje o ich prawidłowym prowadzeniu i uaktualnianiu
- ❑ obsługiwany system oprogramowania wbudowanego
- ❑ specjalne funkcje wielozadaniowe
- ❑ wymagane usługi systemowe, pamięć, sterowniki ...

C.7.2 Funkcje konwersji "C"

Pakiet ISaGRAF obejmuje program do **konwersji liniowej** przeznaczony do wykonywania prostej konwersji analogowych wielkości We/Wy w czasie wykonywania aplikacji na PLC z oprogramowaniem wbudowanym ISaGRAF. Ten program nie wymaga opracowywania niczego w języku "C" ponieważ jest ograniczony wyłącznie do zwiększania lub zmniejszania wartości ciągłych funkcji. Kompletny opis tych narzędzi podano w Podręczniku użytkownika ISaGRAF.

Funkcje konwersji umożliwiają użytkownikowi zastosowanie dowolnej złożonej konwersji z użyciem specjalnych operacji opisanych w języku "C". Zasadniczo funkcja konwersji jest definiowana **zarówno dla wejść jak i dla wyjść**. Nawet, jeśli konwersja w jednym z kierunków nie jest stosowana, to przed zintegrowaniem funkcji konwersji z jądrem ISaGRAF trzeba przeprowadzić implementację i próby, aby zapobiec awaryjnemu zatrzymaniu systemu z powodu złego wywołania.

Funkcje konwersji są pisane w języku "C", a potem kompilowane i włączane do jądra ISaGRAF. Przed użyciem nowych funkcji konwersji w projektach ISaGRAF w PLC z oprogramowaniem wbudowanym ISaGRAF musi zostać zainstalowane większe jądro. Nowe funkcje konwersji nie mogą być zintegrowane w symulatorze ISaGRAF. Aplikacje ISaGRAF wymagają przeprowadzenia symulacji **przed** wstawieniem niestandardowych funkcji konwersji.

Kody źródłowe w języku "C" standardowych funkcji konwersji napisane przez CJ International są instalowane w pakiecie ISaGRAF. Mogą zostać wykorzystane jako przykłady do tworzenia nowych funkcji. Zaleca się **nie dokonywanie modyfikacji** w funkcjach standardowych, aby mogły one być używane w aplikacji ISaGRAF. Standardowe funkcje konwersji dostarczone wraz z pakietem ISaGRAF są obsługiwane przez symulator ISaGRAF.

Ostrzeżenie: Funkcje konwersji są operacjami **synchronicznymi**, uruchamianymi przez Menedżera WE/Wy ISaGRAF podczas wejściowej i wyjściowej fazy cyklu wykonywania aplikacji. Czas wykonywania funkcji konwersji jest wliczony do **parametrów czasowych cyklu** aplikacji ISaGRAF. Użytkownik musi zadbać, aby w funkcji konwersji nie została zaprogramowana żadna "operacja czekania", aby wykonywanie cyklu ISaGRAF nie było niepotrzebnie wydłużone.

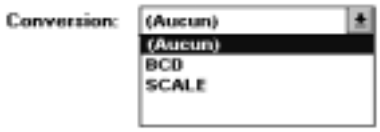
Dodawanie funkcji do biblioteki ISaGRAF

Do dodania nowej funkcji konwersji do biblioteki w pakiecie ISaGRAF należy użyć programu Biblioteki ISaGRAF. Po wybraniu bibliotek funkcji konwersji należy użyć komendę "**Nowy**" w menu "**Pliki**". W pakiecie ISaGRAF nie trzeba definiować żadnego parametru, ponieważ funkcje konwersji korzystają z wstępnie określonego standardowego interfejsu.

Po utworzeniu nowej funkcji konwersji należy napisać dla niej **opis techniczny**. Program Biblioteki ISaGRAF automatycznie wygeneruje dla nowej funkcji konwersji ramkę kodu źródłowego w języku "C".

Stosowanie konwersji w projekcie ISaGRAF

Zdefiniowane funkcje konwersji mogą zostać użyte do filtrowania wartości dowolnej zmiennej analogowej wejściowej lub wyjściowej wybranego projektu. Aby powiązać funkcję konwersji ze zmienną, uruchamia się edytor deklarowania zmiennych, wybiera wejściową lub wyjściową zmienną analogową a potem edytuje jej parametry. Do ustawienia funkcji konwersji powiązanej ze zmienną analogową We/Wy jest stosowane pole "**konwersja**" w oknie dialogowym deklarowania zmiennych analogowych:



Na liście pojawią się zarówno funkcje jak i tablice konwersji. Oznacza to, że funkcji i tablicy nie można nadać tej samej nazwy. Nie można powiązać zmiennej z funkcją konwersji, która jeszcze nie została zdefiniowana lub zintegrowana z jądrem ISaGRAF.

Standardowy interfejs języka "C"

Iinterfejs funkcji konwersji ma zawsze ten sam format. Parametry wywołania i parametr zwracany są przekazywane przez pewną strukturę. Struktura ta jest zdefiniowana w pliku "**TACN0DEF.h**":

```
/*
  Nazwa: tacn0def.h
  Plik definicyjny konwersji w oprogramowaniu wbudowanym
*/

#define DIR_INPUT 0          /* direction („kierunek”) = konwersja zm. wejściowej */
#define DIR_OUTPUT 1         /* direction („kierunek”) = konwersja zm. wyjściowej */

typedef int32   T_ANA;       /* wielkość analogowa typu integer */
typedef float   T_REAL;      /* wielkość analogowa typu real */

typedef struct {       /* struktura konwersji */
     uint16 number;  /* numer konwersji (zastrzeżony)  */
     uint16 direction;/* direction („kierunek”) konwersji          */
     T_REAL *before;           /* wartość przed konwersją       */
     T_REAL *after; /* wartość po konwersji       */
  } str_cnv;

#define ARG_BEFORE (*(arg->before))
#define ARG_AFTER  (*(arg->after))
#define DIRECTION  (arg->direction)

/* koniec pliku */
```

Struktura "**str_cnv**" jest kompletnym opisem interfejsu. Jedynym parametrem funkcji konwersji "C" jest wskaźnik do struktury. Pole "**Numbrer**" jest numerem logicznym funkcji konwersji (lokalizacji w bibliotece ISaGRAF) i nie musi być używane w programowaniu.

Pole "**direction („kierunek”)**" wskazuje, czy konwersja dotyczy zmiennej wejściowej czy wyjściowej. Zawiera wartość **DIR_INPUT**, gdy konwersja dotyczy zmiennej wejściowej lub wartość **DIR_OUTPUT**, gdy dotyczy zmiennej wyjściowej.

Pole "**przed**" wskazuje wartość przed konwersją. Pole to ma inne znaczenie dla konwersji zmiennej wejściowej a inne dla zmiennej wyjściowej. W przypadku konwersji zmiennej wejściowej, gdy pole **direction („kierunek”)** przyjmuje wartość **DIR_INPUT**, oznacza ono

wartość elektryczną (odczytywaną na urządzeniu wejściowym). W przypadku konwersji zmiennej wyjściowej, gdy pole **direction („kierunek”)** przyjmuje wartość **DIR_OUTPUT**, oznacza ono wartość fizyczną (stosowaną w programowanych równaniach).

Pole "**po**" wskazuje wartość po konwersji. Pole to ma inne znaczenie w konwersji zmiennej wejściowej i wyjściowej. W przypadku konwersji zmiennej wejściowej, gdy pole **direction („kierunek”)** przyjmuje wartość **DIR_INPUT**, oznacza ono wartość fizyczną (stosowaną w programowanych równaniach). W przypadku konwersji zmiennej wyjściowej, gdy pole **direction („kierunek”)** przyjmuje wartość **DIR_OUTPUT**, oznacza ono wartość elektryczną (wysyłaną do urządzenia wyjściowego.

Programista może wykorzystać definicje "**ARG_BEFORE**" i "**ARG_AFTER**" do uzyskania bezpośredniego dostępu do pól **przed** i **po** struktury przekazywanej do funkcji konwersji "C". Przetwarzane wartości są **liczbami zmiennoprzecinkowymi pojedynczej precyzji**. W rezultacie zostają przekształcone na liczbę typu long integer, gdy konwersja dotyczy zmiennej analogowej typu integer. Oznacza to, że tę samą funkcję konwersji można zastosować do zmiennych analogowych We/Wy typu real lub integer.

Kod źródłowy

Ponieważ funkcja konwersji może być stosowana zarówno do zmiennych analogowych wejściowych jak i wyjściowych, więc kod źródłowy "C" funkcji jest podzielony na dwie główne części: konwersję wejścia i konwersję wyjścia. Do wyboru rodzaju stosowanej konwersji służy pole **direction („kierunek”)** struktury interfejsowej. Po utworzeniu funkcji konwersji Menedżer Bibliotek ISaGRAF automatycznie generuje kompletną ramkę funkcji. Zawiera ona główną strukturę wyboru "**IF**". Poniżej przedstawiono standardową ramkę funkcji konwersji:

```
/*
  nazwa funkcji
  konwersji: przykład
*/

#include <tasy0def.h>
#include <tacn0def.h>

void CNV_sample (str_cnv *arg)
{
        if (DIRECTION == DIR_INPUT) { /*INPUT CONV*/

        }
        else { /*OUTPUT CONV*/

        }
}

/* Następująca funkcja pokazuje powiązanie z Menedżerem We/Wy ISaGRAF stosującym tę
samą nazwę funkcji konwersji. Funkcja ta jest całkowicie generowana przez Menedżera
Bibliotek ISaGRAF. */

UFP cnvdef_sample (char *name)
{
        sys_strcpy (name, "SAMPLE");          /* podaje nazwę konwersji */
```

```
        return (CNV_sample);        /* zwraca funkcję implementacji */
}
```

Najlepszym sposobem realizacji określonej części funkcji jest napisanie dwóch odrębnych funkcji, jednej dla konwersji zmiennej wejściowej a drugiej dla konwersji zmiennej wyjściowej. Funkcje te będą wywoływane przez zasadniczy algorytm w głównej strukturze **IF**, jak to pokazano w komentarzu do poprzedniego przykładu.

"**TASY0DEF.H**" obejmuje plik z jądra ISaGRAF, który jest potrzebny do definicji systemowych. Zawiera też definicję typu **UFP**, która jest wskaźnikiem do pustej funkcji i jest używana w funkcji deklaracji.

- Połączenia pomiędzy projektami a implementacją "C"

Logicznym łącznikiem pomiędzy implementacją funkcji konwersji a zastosowaniem konwersji w projekcie ISaGRAF jest nazwa tej konwersji. Do kodu źródłowego funkcji konwersji w języku "C" jest dodawana funkcja "deklaracji". Funkcja ta jest wywoływana tylko jeden raz w chwili uruchamiania aplikacji i podaje menedżerowi We/Wy ISaGRAF nazwę konwersji, która odpowiada funkcji, jaka ma zostać zaimplementowana. Poniżej przedstawiono standardowy format takiej funkcji deklaracji:

```
UFP cnvdef_xxx (char *name)
{
        strcpy (name, "XXX");       /* podaje nazwę konwersji */
        return (CNV_xxx);   /* zwraca funkcję implementacji */
}
/* (xxx jest nazwą konwersji) */
```

Nazwa funkcji używana w instrukcji **strcpy** musi być napisana **dużymi literami**. Należy ją pisać małymi literami w nazwie funkcji implementacji konwersji i w nazwie funkcji deklaracji.

Stosowanie przedrostków "**CNV**_" i "**cnvdef**_" w nazwach funkcji implementacji i funkcji definicji umożliwia użytkownikowi nazwanie konwersji zastrzeżonym słowem kluczowym języka "C" lub nazwą funkcji istniejącej w bibliotekach ISaGRAF z aplikacjami w języku "C".

Do funkcji deklaracji można dodać inne instrukcje, aby zrealizować określoną operację inicjalizacji dotyczącą tej konwersji. W systemie ISaGRAF funkcja ta jest wywoływana **tylko jeden raz** w chwili uruchamiania aplikacji.

Funkcja deklaracji jest wywoływana dla każdej zintegrowanej z jądrem funkcji konwersji nawet, jeśli nie jest używana w aplikacji ISaGRAF. Jeśli używana w aplikacji funkcja konwersji nie jest zintegrowana z jądrem, to w jądrze ISaGRAF wystąpi błąd krytyczny.

Przed połączeniem nowych funkcji z jądrem użytkownik musi napisać inny program źródłowy w języku "C" z nazwie "**GRCN0LIB.C**" i umieścić go (wraz z funkcjami konwersji) na liście plików do włączenia. Program "**GRCN0LIB.C**" zawiera tylko tablicę funkcji deklaracji. Tablica ta jest odczytywana podczas inicjowania aplikacji w celu stworzenia dynamicznego połączenia z funkcjami konwersji napisanymi w języku "C". Poniżej podano przykład takiego pliku:

```
/* File "GRCN0LIB.c" - Przykład konwersji ze standardowej biblioteki */

#include <tasy0def.h>          /* potrzebne do definicji typu */
```

```
extern UFP cnvdef_scale (char *name); /* funkcja deklar. dla konw. SCALE */
extern UFP cnvdef_bcd (char *name);   /* funkcja deklar. dla konw BCD */

UFP_LIST CNVDEF[ ] = {      /* tablica funkcji deklaracji dla */
        /* zintegrowanych funkcji konwersji */
        cnvdef_scale,
        cnvdef_bcd,

NULL };

/* koniec pliku */
```

Tablica **CNVDEF** musi być zakończona wskaźnikiem NULL. Niespełnienie tego warunku spowoduje pewne problemy. Niezdefiniowanie tablicy **CNVDEF** spowoduje wystąpienie niezdefiniowanych wskaźników przy dołączaniu nowego jądra ISaGRAF.

Napisanie tego pliku umożliwi zbudowanie nowego jądra łącznie ze wszystkimi istniejącymi funkcjami konwersji. Można też zbudować jądro przystosowane do realizacji jednego projektu. W tym celu w tablicy **CNVDEF** należy umieścić tylko te funkcje konwersji, które są używane w projekcie. Plik "**GRCN0LIB.C**" jest automatycznie generowany przez Generator Kodu ISaGRAF podczas tworzenia kodu aplikacji. Plik jest umieszczany w katalogu projektów ISaGRAF i grupuje tylko te funkcje konwersji, które są stosowane w projekcie.

- Ograniczenia

Biblioteka ISaGRAF może zawierać do **128** funkcji konwersji. Funkcja konwersji może być użyta do przetworzenia dowolnego typu operacji. Należy zwrócić uwagę, że funkcje są wywoływane w cyklu ISaGRAF w sposób **synchroniczny**, a więc ich wykonanie ma bezpośredni wpływ na parametry czasowe cyklu.

C.7.3 Funkcje "C"

Funkcje "C" służą do zwiększania standardowych możliwości języka **ST** i **FBD**. Mogą być użyte do wykonania określonych obliczeń, wywołań systemu, komunikacji lub do zainstalowania zbioru usług do porozumiewania się aplikacji ISaGRAF z innymi zadaniami. Funkcje są pisane w języku "C", kompilowane i łączone z jądrem ISaGRAF. Przed zastosowaniem nowych funkcji w projektach ISaGRAF należy w PLC z oprogramowaniem wbudowanym ISaGRAF zainstalować powiększone jądro.

Nowe funkcje nie mogą być zintegrowane z Symulatorem ISaGRAF. Symulacja aplikacji ISaGRAF musi się odbyć **przed** zastosowaniem funkcji niestandardowych.

Ostrzeżenie: Funkcje są operacjami **synchronicznymi**, uruchamianymi przez jądro ISaGRAF podczas cyklu wykonywania aplikacji. Czas wykonywania funkcji jest wliczony do **parametrów czasowych cyklu** aplikacji ISaGRAF. Użytkownik musi zadbać, aby w funkcji konwersji nie została zaprogramowana żadna "operacja czekania", aby wykonywanie cyklu ISaGRAF nie było niepotrzebnie wydłużone.

- ***Dodawanie funkcji do biblioteki ISaGRAF***

Do dodania nowej funkcji do biblioteki w pakiecie ISaGRAF należy użyć program Biblioteki ISaGRAF. Po wybraniu biblioteki funkcji należy użyć komendę "**Nowy**" w menu "**Plik**". Po utworzeniu nowej funkcji należy napisać dla niej **opis techniczny**. Program Biblioteki ISaGRAF automatycznie wygeneruje dla nowej funkcji ramkę kodu źródłowego w języku "C".

Do definiowania parametrów wywołania i parametru zwracanego służy komenda "**Parametry**" z menu "**Edycja**".

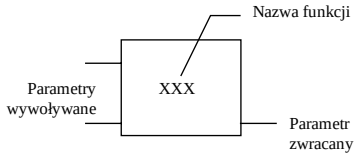
Stosowanie funkcji "C" w projekcie ISaGRAF

Każda zintegrowana funkcja "C" może być stosowana jako funkcja standardowa w programach projektu ISaGRAF. Funkcje "C" mogą być wywołane instrukcjami języków **ST** i **FBD** i specjalnymi instrukcjami języka **SFC**.

Wywołanie funkcji "C" w języku **ST** odbywa się według zasad wywoływania funkcji obowiązujących w tym języku. Parametry wywołania funkcji są podawane po nazwie funkcji, są ujęte w nawiasy okrągłe i oddzielone przecinkami. Wyrażenie oznacza wartość zwracaną przez funkcję. Wywołanie funkcji "C" może być wstawione do każdej instrukcji przyporządkowania lub do wyrażenia złożonego. Poniżej podano przykład wywołania funkcji "C" w instrukcji przyporządkowania:

result := ProcName (par1, par2, ... parN);

Program w języku **FBD** może wywołać dowolną funkcje "C". Funkcja jest stosowana jako standardowa skrzynka funkcji oznaczana prostokątem. Parametry wywołania są doprowadzane z lewej strony prostokąta. Parametr zwracany jest widoczny po prawej stronie. Poniżej przedstawiono schemat standardowej funkcji:



Funkcja "C" może być wywołana z dowolnego bloku operacji **SFC** lub w dowolnym warunku dwustanowym powiązanym z operacją przejścia.

Definicja interfejsu funkcji "C"

Do definiowania parametrów wywołania i parametru zwracanego nowej funkcji służy komenda "**Parametry**" w menu "**Edycja**". Funkcja może zawierać do **31** parametrów wywołania i zawsze **jeden** parametr zwracany.

Lista w górnej części okna przedstawia parametry funkcji "C" w kolejności ich wywoływania: najpierw parametry wywołania, a na końcu parametr zwracany. Dolna część okna przedstawia szczegółowy opis parametru aktualnie wybranego z listy:
- nazwa parametru
- rodzaj parametru (wywołania / zwracany)
- typ parametru

Parametr może zawierać dowolny typ danych ISaGRAF: dane binarne, analogowe typu integer, analogowe typu real, Timer lub komunikat. Trzeba wyróżnić dane analogowy typu integer i real.
Poniżej podano typy danych ISaGRAF i typy w języku "C":

BINARNA	Long integer bez znaku	Słowo 32 bitowe bez znaku: 1=prawda / 0=fałsz
ANALOGOWA	Long integer	Słowo 32 bitowe integer ze znakiem
REAL	Liczba zmienno-przecinkowa	Wartość zmiennoprzecinkowa pojedynczej precyzji
TIMER	Long integer bez znaku	Słowo 32 bitowe integer bez znaku (jednostką jest 1 milisekunda)
KOMUNIKAT	znak *	Ciąg znaków.

Jeśli parametrem funkcji "C" jest komunikat, to nie może on zawierać znaków pustych. Ciąg znaków w funkcji "C" jest zamykany znakiem pustym. Należy pamiętać, że parametr zwracany musi być ostatnim parametrem na liście. Parametry muszą być nazywane zgodnie z poniższymi zasadami:
- długość nazwy nie może przekroczyć 16 znaków
- pierwszym znakiem musi być litera
- kolejne znaki muszą być literami, cyframi lub znakiem podkreślenia
- ważne jest, czy są użyte duże czy małe litery alfabetu.

Ta sama nazwa nie może być użyta do identyfikacji więcej niż jednego parametru funkcji. Parametr wywołania nie może mieć takiej samej nazwy jak parametr zwracany. Ta sama nazwa **może** być użyta dla oznaczenia parametrów różnych funkcji. Domyślną nazwą parametru zwracanego jest "**Q**". Ta nazwa może być dowolnie modyfikowana. Nazwa parametru służy do identyfikacji parametru w kodzie źródłowym w języku "C".

Komenda "**Wstaw**" służy do wstawienia nowego parametru przed wybrany parametr. Komenda "**Usuń**" jest używana do usunięcia wybranego parametru. Komenda "**Ułóż**" automatycznie porządkuje (sortuje) parametry, więc parametr zwracany jest umieszczany na końcu listy. Naciśnięcie przycisku "**OK**" zapamiętuje definicję interfejsu funkcji i zamyka okienko dialogowe. Naciśnięcie przycisku "**Anuluj**" zamyka okienko dialogowe bez zmiany definicji interfejsu funkcji.

Interfejs funkcji "C"

Interfejs funkcji zależy od definicji jej parametrów. Parametry wywołania i parametr zwracany przechodzą przez pewną strukturę. Struktura ta jest zdefiniowana w pliku "**GRUS0nnn.H**", gdzie "**nnn**" jest numerem logicznym funkcji w bibliotece ISaGRAF. Poniżej podano przykład interfejsu "C" dla funkcji "**SIN**" (obliczanie sinusa):

```
/* File: GRUS0255.h – funkcja "Sample" */

typedef  long          T_BOO;
typedef  long          T_ANA;
typedef  float         T_REAL;
typedef  long          T_TMR;
typedef  char          *T_MSG;
```

```
typedef struct {
        /* CALL   */                T_REAL   _param1;
        /* RETURN */                T_REAL   _param2;
} str_arg;

#define PARAM1                      (arg->_param1)
#define PARAM2                      (arg->_param2)

/* koniec pliku */
```

Zależności pomiędzy typami danych ISaGRAF a typami "C" pokazano w poniższej tabelce. W pliku definicji funkcji typy ISaGRAF są zdefiniowane jako typy "C".

Binarna	T_BOO	Liczba długa (32 bity)
Analogowa integer	T_ANA	Liczba długa
Analogowa real	T_REAL	Liczba zmiennoprzecinkowa (32 bity – pojedynczej precyzji)
Timer	T_TMR	Liczba długa
Komunikat	T_MSG	Znak * (32 bity – wskaźnik do tekstu)

Każde pole struktury "**str_arg**" odpowiada jednemu parametrowi funkcji. Parametr zwracany jest ostatnim w strukturze. Parametry wywołania pojawiają się w strukturze w takiej samej kolejności, jaką ustalono w definicji funkcji. Identyfikator określony dużymi literami alfabetu jest tak zdefiniowany, że zapewnia dostęp do jednego parametru struktury przekazanej do implementacji "C" funkcji. Nazwy identyfikatorów są wprowadzane podczas definiowania funkcji przy pomocy Menedżera Bibliotek ISaGRAF.

Plik definicyjny "C" jest uaktualniany po każdej zmianie interfejsu funkcji przy pomocy Menedżera Bibliotek ISaGRAF. Zapewnia to całkowitą zgodność pomiędzy implementacją funkcji a jej zastosowaniem w programach aplikacji ISaGRAF.

Kod źródłowy

Poniżej przedstawiono standardową ramkę implementacji funkcji "C":

```
/* Przykład funkcji użytkownika - Numer "255" – Nazwa "SAMPLE" */

#include "tasy0def.h"               /* typowe definicje jądra ISaGRAF */
#include "grus0255.h"               /* definicja interfejsu dla funkcji 255 */

void USP_sample (str_arg *arg)
{
        /* struktura funkcji */
}
```

```
/* Następująca funkcja służy do inicjalizacji funkcji i deklaracji dla potrzeb jej
implementacji. Poprzez nazwę funkcji zapewnia ona połączenie z jądrem
ISaGRAF. Funkcja ta jest całkowicie generowana przez Menedżera Bibliotek
ISaGRAF. */

UFP uspdef_sample (char *name)
{
        strcpy (name, "SAMPLE"); /* podaje nazwę funkcji */
        return (USP_sample);     /* zwraca funkcję implementacji */
}

/* koniec pliku */
```

"**TASY0DEF.H**" obejmuje plik z jądra ISaGRAF, który jest potrzebny do definicji systemowych. Zawiera też definicję typu **UFP** stanowiącą wskaźnik do pustej funkcji i jest używany w funkcji deklaracji.

Połączenia pomiędzy projektami a implementacją "C"

Logicznym łącznikiem pomiędzy implementacją funkcji "C" a jej zastosowaniem w programach projektu ISaGRAF jest nazwa tej funkcji. Do kodu źródłowego funkcji w języku "C" jest dodawana funkcja "deklaracji". Funkcja ta jest wywoływana tylko jeden raz w chwili uruchamiania aplikacji i wskazuje jądru ISaGRAF nazwę funkcji "C", która odpowiada funkcji implementowanej. Poniżej przedstawiono standardowy format takiej funkcji deklaracji:

```
UFP uspdef_xxx (char *name)
{
        strcpy (name, "XXX");     /* podaje nazwę funkcji */
        return (USP_xxx);         /* zwraca funkcję implementacji */
}
/* (xxx jest nazwą funkcji) */
```

Nazwa funkcji "C" używana w instrukcji **strcpy** musi być napisana **dużymi literami**. Należy ją pisać małymi literami w nazwie funkcji implementacji i w nazwie funkcji deklaracji. Stosowanie przedrostków "**USP**_" i "**uspdef**_" w nazwach funkcji implementacji i funkcji definicji umożliwia użytkownikowi nazwanie funkcji zastrzeżonym słowem kluczowym języka "C" lub nazwą funkcji istniejącej w bibliotekach ISaGRAF z aplikacjami w języku "C".

Do funkcji deklaracji można dodać inne instrukcje, aby zrealizować określoną operację inicjalizacji dotyczącą tej funkcji. W systemie ISaGRAF funkcja ta jest wywoływana **tylko jeden raz** w chwili uruchamiania aplikacji. Funkcja deklaracji jest wywoływana dla każdej zintegrowanej z jądrem funkcji "C" nawet, jeśli nie jest używana w programach aplikacji ISaGRAF. Jeśli używana w aplikacji funkcja "C" nie jest zintegrowana z jądrem, to w jądrze ISaGRAF wystąpi błąd krytyczny.

Przed połączeniem nowych funkcji z jądrem użytkownik musi napisać inny program źródłowy w języku "C" z nazwie "**GRUS0LIB.C**" i umieścić go (wraz z funkcjami) na liście plików do

włączenia. Program "**GRUS0LIB.C**" zawiera tylko tablicę funkcji deklaracji. Tablica ta jest odczytywana podczas inicjowania aplikacji w celu stworzenia dynamicznego połączenia z funkcjami napisanymi w języku "C". Poniżej podano przykład takiego pliku:

```
/* Plik "GRUS0LIB.c" - Przykład wykorzystujący funkcje trygonometryczne */

#include <tasy0def.h>                /* potrzebne do definicji typów */

extern UFP uspdef_fc1 (char *name);        /* funkcje deklaracji */
extern UFP uspdef_fc2 (char *name);
extern UFP uspdef_fc3 (char *name);
extern UFP uspdef_fc4 (char *name);

UFP_LIST USPDEF[ ] = {              /* tablica funkcji deklaracji  */
        /* dla funkcji zintegrowanych */
        uspdef_fc1,
        uspdef_fc2,
        uspdef_fc3,
        uspdef_fc4,

NULL };

/* koniec pliku */
```

Tablica **USPDEF** musi być zakończona wskaźnikiem NULL. Niespełnienie tego warunku spowoduje pewne problemy. Niezdefiniowanie tablicy **USPDEF** spowoduje wystąpienie niezdefiniowanych odsyłaczy przy dołączaniu nowego jądra ISaGRAF. Napisanie tego pliku umożliwi zbudowanie nowego jądra łącznie ze wszystkimi istniejącymi funkcjami. Można też zbudować jądro przystosowane do realizacji jednego projektu. W tym celu w tablicy **USPDEF** należy umieścić tylko te funkcje, które są używane w projekcie. Plik "**GRUS0LIB.C**" jest automatycznie generowany przez Generator Kodu ISaGRAF podczas tworzenia kodu aplikacji. Plik jest umieszczany w katalogu projektów ISaGRAF i grupuje tylko te funkcje, które są używane w projekcie.

- Ograniczenia

Biblioteka ISaGRAF może zawierać do **255** funkcji "C". Funkcja może być użyta do wykonania dowolnego typu operacji. Należy zwrócić uwagę, że funkcje są wywoływane w cyklu ISaGRAF w sposób **synchroniczny**, a więc ich wykonanie ma bezpośredni wpływ na parametry czasowe cyklu.

- Kompletny przykład

Poniżej podano kompletnie zaprogramowaną funkcję "**Sample**" wykonującą operację dodawania. Opis techniczny funkcji jest następujący:

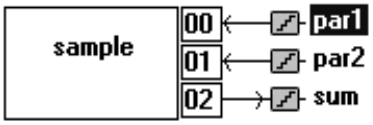
nazwa: SAMPLE
opis: wykonuje dodawanie wielkości analogowych integer

data utworzenia: 1 lipca 1992
autor: CJ International

par. wywołania: par1, par2: operandy typu integer
par. zwracany: suma typu integer

prototype: sum := sample (par1, par2);

Poniżej podano interfejs funkcji:



Nagłówek funkcji w kodzie źródłowym w języku "C" jest następujący:

```
/* Plik: GRUS0255.h – definicja funkcji C - Nazwa: Sample */

/* definicja standardowych typów danych  ISaGRAF */

typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definicja struktury parametrów wywołania i parametru zwracanego */

typedef struct {
        T_ANA _par1;      /* parametr wywołania #1 */
        T_ANA _par2;      /* parametr wywołania #2 */
        T_ANA _sum;       /* parametr zwracany */
} str_arg;

/* identyfikatory używane do korzystania z parametrów wywołania i parametru zwracanego*/

#define PAR1      (arg->_par1)
#define PAR2      (arg->_par2)
#define SUM       (arg->_sum)

/* koniec pliku  */
```

Poniżej przedstawiono kod źródłowy funkcji w języku "C". Ręcznie zostały przez programistę wprowadzone tylko wiersze wydrukowane drukiem wytłuszczonym.

```
/* Plik: GRUS0255.c – funkcja C - Nazwa: SAMPLE */

#include "tasy0def.h"       /* potrzebne do definicji typów */
#include "grus0255.h"       /* nagłówek źródłowy funkcji C */
```

```
/* główne zadanie funkcji C: wykonuje dodawanie */

void USP_sample (str_arg *arg)
{
        SUM = PAR1 + PAR2;
}

/* deklaracja potrzebna do dynamicznego połączenia z jądrem ISaGRAF */

UFP uspdef_sample (char *name)
{
        strcpy (name, "SAMPLE");
        return (USP_sample);
}
/* koniec pliku */
```

C.7.4 BLOKI FUNKCYJNE "C"

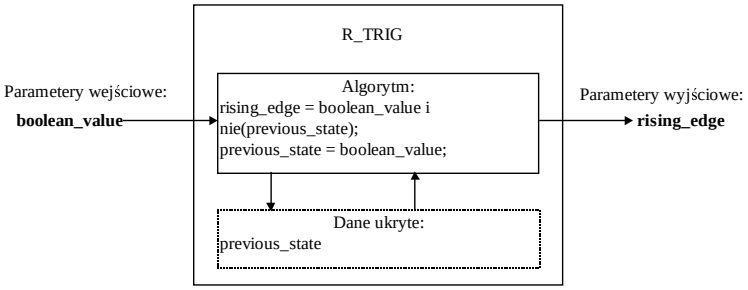
Bloki funkcyjne "C" łączą z sobą operacje i dane statyczne. Uzupełniają zbiór funkcji "C" przez stworzenie możliwości przetwarzania obiektów statycznych. Są powszechnie stosowane do zwiększenia standardowych możliwości języków **ST** i **FBD**. W przeciwieństwie do funkcji, które przetwarzają wartości, bloki funkcyjne mogą przetwarzać dane statyczne. Oznacza to, że algorytm bloku funkcyjnego może uwzględniać zmiany danych w czasie.

Bloki funkcyjne są napisane w języku "C", kompilowane i łączone z jądrem ISaGRAF. Przed zastosowaniem nowych bloków funkcyjnych w projektach ISaGRAF należy w PLC z oprogramowaniem wbudowanym zainstalować powiększone jądro. Nie można integrować nowych bloków funkcyjnych w Symulatorze ISaGRAF. Symulację aplikacji ISaGRAF należy przeprowadzić **przed** zastosowaniem funkcji niestandardowych.

Ostrzeżenie: Bloki funkcyjne są operacjami **synchronicznymi**, uruchamianymi przez jądro ISaGRAF podczas cyklu wykonywania aplikacji. Czas przeznaczony na wykonanie aktywacji lub odczytania bloku funkcji jest wliczony do **parametrów czasowych cyklu** aplikacji ISaGRAF. Użytkownik musi zadbać, aby w funkcji konwersji nie została zaprogramowana żadna "operacja czekania", ażeby wykonywanie cyklu ISaGRAF nie przekroczyło maksymalnie dozwolonego limitu.

Deklarowanie instancji bloków funkcyjnych

Blok funkcyjny jest obiektem łączącym operacje z danymi statycznymi. Poniżej podano przykład bloku funkcyjnego "**R_TRIG**", który wykrywa zbocze narastające wyrażenia dwustanowego. Oto opis funkcjonalny bloku:



Do wyliczenie zbocza jest potrzebna ukryta zmienna statyczna "**previous_state**". Zmienna ta musi być różna dla każdego zastosowania w aplikacji bloku funkcyjnego "**TRIG**". Instancje bloków funkcyjnych stosowanych w języku ST muszą być zadeklarowane w słowniku. Ponieważ blok funkcyjny ma ukryte wewnętrzne dane, więc każda kopia (instancja) bloku funkcyjnego musi być identyfikowana charakterystyczną nazwą. Nazwy typów bloków są nadawane przy pomocy programu Biblioteki. Nazwy instancji są nadawane przy pomocy edytora słownika.

Bloki funkcyjne stosowane w języku FBD nie muszą być deklarowane, ponieważ edytor FBD ISaGRAF automatycznie deklaruje instancje stosowanych bloków. Instancje bloków funkcyjnych automatycznie deklarowane przez edytor FBD są zawsze **LOKALNE** dla edytowanego programu.

Dodawanie bloku funkcyjnego do biblioteki ISaGRAF

Do dodania nowego bloku funkcyjnego "C" do biblioteki w pakiecie ISaGRAF należy użyć programu Biblioteki ISaGRAF. Po wybraniu biblioteki bloków funkcyjnych należy użyć komendę "**Nowy**" w menu "**Plik**". Po utworzeniu nowego bloku funkcyjnego należy napisać dla niego **opis techniczny**. Program Biblioteki ISaGRAF automatycznie wygeneruje dla nowego bloku funkcyjnego ramkę kodu źródłowego w języku "C". Do definiowania parametrów wywołania i parametru zwracanego nowego bloku funkcyjnego służy komenda "**Parametry**" z menu "**Edycja**".

Stosowanie bloku funkcyjnego "C" w projekcie ISaGRAF

Każdy zintegrowany blok funkcyjny "C" może być stosowany w programach projektu ISaGRAF. Bloki funkcyjne "C" mogą być wywołane z języków **ST** i **FBD**.

Wywołanie bloku funkcyjnego "C" z języka **ST** odbywa się według zasad wywoływania funkcji obowiązujących w tym języku. Parametry wywołania bloku są podawane po nazwie funkcji, są ujęte w nawiasy okrągłe i oddzielone przecinkami. Z parametrów zwracanych korzysta się pojedynczo. Każdy parametr zwracany jest reprezentowany przez nazwę łączącą nazwę instancji bloku i nazwę parametru. Komponenty nazwy są oddzielane kropką. Na przykład nazwa:

FBINSTNAME.parname

przedstawia parametr zwracany o nazwie "**parname**" dla instancji bloku funkcyjnego o nazwie "**FBINSTNAME**".

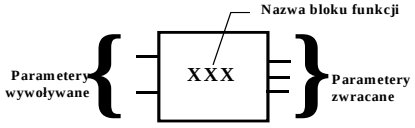
Instancje bloków funkcyjnych stosowane w języku ST muszą być deklarowane w słowniku. Każda kopia (instancja) bloku funkcyjnego musi być identyfikowana charakterystyczną nazwą. Poniżej podano przykład deklaracji instancji w słowniku ISaGRAF:

instancja:	TRIG1	typ:	R_TRIG
	TRIG2		R_TRIG

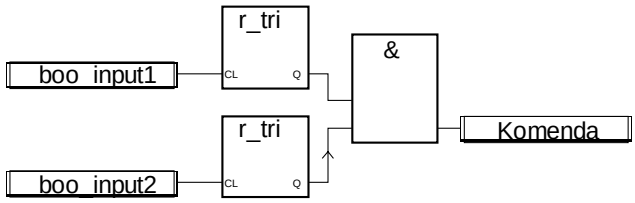
Kolejny przykład przedstawia zastosowanie tych zadeklarowanych instancji w programie w języku ST:

```
TRIG1 (boo_input1);
TRIG2 (boo_input2);
Komenda := (TRIG1.Q & TRIG2.Q);
```

Program w języku **FBD** może wywołać dowolny blok funkcyjny "C". Blok funkcyjny jest stosowany jako standardowa skrzynka funkcji oznaczana prostokątem. Parametry wywołania są doprowadzane z lewej strony prostokąta. Parametry zwracane są widoczne po prawej stronie. Poniżej przedstawiono schemat standardowej funkcji:



Bloki funkcyjne stosowane w języku FBD nie muszą być deklarowane, ponieważ edytor FBD ISaGRAF automatycznie deklaruje instancje stosowanych bloków. Instancje bloków funkcyjnych automatycznie deklarowane przez edytor FBD są zawsze **LOKALNE** dla edytowanego programu. Poniżej przedstawiono przykład zaprogramowany w języku FBD:



- Definicja interfejsu bloku funkcyjnego "C"

Do definiowania parametrów wywołania i zwracanych nowego bloku funkcyjnego służy komenda "**Parametry**" w menu "**Edit**". Blok funkcyjny może zawierać do **32** parametrów dowolnie dobieranych jako wywołania i zwracane. W przeciwieństwie do funkcji "C" blok funkcyjny może zawierać kilka parametrów zwracanych.

Lista w górnej części okna przedstawia parametry bloku funkcyjnego "C" w kolejności ich wywoływania: najpierw parametry wywołania, a potem parametry zwracane. Dolna część okna przedstawia szczegółowy opis parametru aktualnie wybranego z listy:
- nazwa parametru
- rodzaj parametru (wywołania / zwracany)
- typ parametru

Parametr może zawierać dowolny typ danych ISaGRAF: wielkości dwustanowe, analogowe typu integer, analogowe typu real, Timer lub komunikat. Wielkości analogowe typu integer i real muszą być wyróżnione.
Poniżej podano zależności pomiędzy typami danych ISaGRAF i w języku "C":

BINARNA	Long integer bez znaku	Słowo 32 bitowe bez znaku: 1=prawda / 0=fałsz
ANALOGOWA	Long integer	Słowo 32 bitowe integer ze znakiem
REAL	Liczba zmienno-przecinkowa	Wartość zmiennoprzecinkowa pojedynczej precyzji
TIMER	Long integer bez znaku	Słowo 32 bitowe integer bez znaku (jednostką jest 1 milisekunda)
KOMUNIKAT	znak *	Ciąg znaków.

Jeśli parametrem bloku funkcyjnego "C" jest komunikat, to nie może on zawierać znaków pustych. Ciąg przekazywany do kodu w języku "C" jest zamykany znakiem pustym. Należy pamiętać, że parametry zwracane muszą się znajdować na końcu listy. Parametry muszą być nazywane zgodnie z poniższymi zasadami:
- długość nazwy nie może przekroczyć 16 znaków
- pierwszym znakiem musi być litera
- kolejne znaki muszą być literami, cyframi lub znakiem '_'
- ważne jest, czy są użyte duże czy małe litery alfabetu.

Ta sama nazwa nie może być użyta do identyfikacji więcej niż jednego parametru bloku funkcyjnego. Parametr wywołania nie może mieć takiej samej nazwy jak parametr zwracany. Ta sama nazwa **może** być użyta dla oznaczenia parametrów różnych bloków funkcyjnych. Nazwa parametru służy do identyfikacji parametru w kodzie źródłowym w języku "C".

Komenda "**Wstaw**" służy do wstawienia nowego parametru przed wybrany parametr. Komenda "**Usuń**" jest używana do usunięcia wybranego parametru. Komenda "**Ułóż**" automatycznie porządkuje (sortuje) parametry, aby parametry zwracane były zawsze umieszczane na końcu listy. Naciśnięcie przycisku "**OK**" zapamiętuje definicję interfejsu bloku funkcyjnego i zamyka okienko dialogowe. Naciśnięcie przycisku "**Anuluj**" zamyka okienko dialogowe bez zmiany definicji interfejsu bloku funkcyjnego.

- Interfejs bloku funkcyjnego "C"

Interfejs bloku funkcyjnego zależy od definicji jego parametrów. Parametry wywołania przechodzą przez pewną strukturę. Struktura ta jest zdefiniowana w pliku "**GRFB0nnn.H**", gdzie "**nnn**" jest numerem logicznym bloku funkcyjnego w bibliotece ISaGRAF. Parametry zwracane są reprezentowane przez numery logiczne, które są również definiowane w pliku "**GRFB0nnn.h**". Poniżej podano przykład interfejsu "C" dla bloku funkcyjnego "**LIM_ALRM**" (alarm po przekroczeniu ograniczeń):

```
/* interfejs bloku funkcyjnego - nazwa: Sample */

/* standardowe typy danych ISaGRAF */

typedef   long     T_BOO;
typedef   long     T_ANA;
typedef   float    T_REAL;
```

```
typedef   long      T_TMR;
typedef   char      *T_MSG;

/* struktura parametrów wywołania */

typedef struct {
          /* CALL  */          T_BOO   _par1;
          /* CALL  */          T_BOO   _par2;
} str_arg;

/* dostęp do pól struktury str_arg  */

#define   PAR1    (arg->_par1)
#define   PAR2    (arg->_par2)

/* numery logiczne parametrów zwracanych */

#define   FBLPNO_Q1       0
#define   FBLPNO_Q2       1

/* koniec pliku */
```

Zależności pomiędzy typami danych ISaGRAF a typami "C" pokazano w poniższej tabelce. W pliku definicji funkcji typy ISaGRAF są zdefiniowane jako typy "C".

Binarna	T_BOO	Liczba długa (32 bity)
Analogowa integer	T_ANA	Liczba długa
Analogowa real	T_REAL	Liczba zmiennoprzecinkowa (32 bity – pojedynczej precyzji)
Timer	T_TMR	Liczba długa
Komunikat	T_MSG	Znak * (32 bity – wskaźnik do tekstu)

Każde pole struktury "**str_arg**" odpowiada jednemu parametrowi wywołania bloku funkcyjnego. Parametry pojawiają się w strukturze w takiej samej kolejności, jaką ustalono w definicji bloku funkcyjnego. Identyfikator określony dużymi literami alfabetu jest tak zdefiniowany, że zapewnia dostęp do jednego parametru struktury przekazanej do implementacji "C" aktywacji bloku funkcyjnego. Nazwy identyfikatorów są wprowadzane podczas definiowania funkcji przy pomocy programu Biblioteki ISaGRAF.

Kolejność numerowania parametrów zwracanych jest ustalona w definicji bloku funkcyjnego. Numerem logicznym pierwszego parametru zwracanego jest zawsze **0**.

W programach źródłowych w języku "C" parametry zwracane powinny być oznaczane zdefiniowanymi identyfikatorami a nie wartościami liczbowymi. Dzięki temu plik źródłowy będzie można łatwo ponownie skompilować po zmodyfikowaniu definicji interfejsu.

Plik definicyjny "C" jest uaktualniany po każdej zmianie interfejsu bloku funkcyjnego przy pomocy programu Biblioteki ISaGRAF. Zapewnia to całkowitą spójność pomiędzy implementacją bloku funkcyjnego a jego zastosowaniem w programach aplikacji ISaGRAF.

- Kod źródłowy

Implementacja "C" bloku funkcyjnego jest podzielona na trzy etapy:

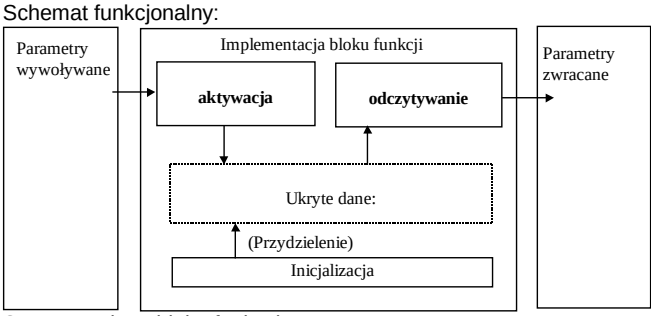
- ❑ inicjalizacja
- ❑ aktywacja – przetwarzanie wywołanych parametrów
- ❑ odczytanie parametrów zwróconych

Do każdej instancji tego samego bloku funkcyjnego jest stosowany ten sam kod i nie jest on powielany. Z każdą instancją jest związana struktura danych statycznych. Z danych tych programy ISaGRAF nie mogą korzystać bezpośrednio i dane zawierają "ukryte zmienne " instancji bloku funkcyjnego.

"Aktywacja" jest wywoływana jeden raz dla każdej instancji każdego bloku w każdym cyklu wykonywania oprogramowania wbudowanego. Jest odpowiedzialna za przetwarzanie parametrów wywołania i uaktualnianie skojarzonych danych. Stanowi ona "główny algorytm" bloku funkcyjnego.

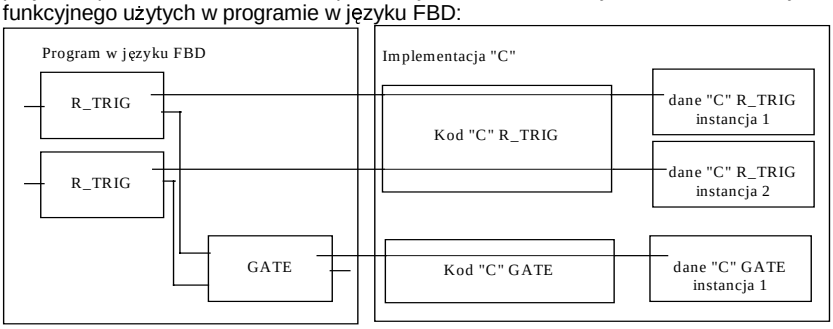
"Odczytywanie" jest wywoływane przez jądro ISaGRAF w celu odczytania aktualnej wartości jednego parametru zwracanego dla jednej instancji. Nie muszą tu być wykonywane żadne specjalne obliczenia. Odczytywanie tylko przekazuje ukryte dane do aplikacji ISaGRAF.

Schemat funkcjonalny:



Statyczne dane bloku funkcyjnego

Blok funkcyjny łączy operacje z danymi statycznymi. Do każdej instancji tego samego bloku funkcyjnego jest przydzielana struktura danych. Każde użycie bloku funkcyjnego w programie w języku ST lub FBD odpowiada jednej instancji i jednej strukturze danych. Następujący przykład przedstawia zależność pomiędzy strukturami danych "C" a instancjami bloku funkcyjnego użytych w programie w języku FBD:



Pamięć, jaką potrzebuje struktura danych każdej instancji jest przydzielana przez system ISaGRAF po uruchomieniu aplikacji. Wskaźnik do związanej z instancją struktury danych jest przekazywany do "aktywacji" i "odczytywania".

Program Biblioteki ISaGRAF automatycznie generuje ramkę kodu źródłowego w języku "C" dla definicji typu struktury danych. Typ struktury danych ma zawsze nazwę "**str_data**". Dla zapewnienia zgodności z brzemieniem nagłówków operacji programista nie powinien zmieniać tej nazwy. Ukryte dane generalnie łączą wewnętrzne zmienne z parametrami zwracanymi. "Odczytywanie" jako czynność wykonywana przez blok funkcyjny służy tylko zapewnienia dostępu do parametru zwracanego, i nie powinna być używana do wykonywania innych operacji.

Inicjalizacja

Funkcja bloku funkcyjnego polegająca na "inicjalizacji" jest wywoływana przez jądro ISaGRAF przy uruchamianiu aplikacji. Pozwala ona programiście języka "C" poprosić system o przydzielenie pamięci dla instancji. Poniżej przedstawiono standardowy program inicjalizacji:

```
uint16 FBINIT_xxx (uint16 hinstance)
/* "xxx" jest nazwą bloku funkcyjnego */
{
  return (sizeof (str_data));
}
```

Argument "**hinstance**" jest numerem logicznym instancji. Jest zastrzeżony dla wewnętrznych operacji ISaGRAF i nie powinien być użyty do programowania tej operacji. Po inicjalizacji zwracana jest liczba bajtów pamięci potrzebnych do zapamiętania danych **jednej** instancji. Wielkość potrzebnej pamięci (wartość zwracana) nie może przekraczać **64** Kbajtów. W ramach tej obsługi nie powinna być wykonywania żadna inna operacja. Kod źródłowy w języku "C" tej operacji jest automatycznie generowany przez Menedżera Bibliotek ISaGRAF w czasie tworzenia *bloku funkcyjnego*.

Aktywacja

"Aktywacja" jest wywoływana przy każdym cyklu realizacji oprogramowania wbudowanego dla każdej instancji *bloku funkcyjnego* stosowanego w aplikacji. Polega na przetwarzaniu parametrów wywołania i wykonaniu głównego algorytm *bloku funkcyjnego* w celu uaktualnienia ukrytych danych statycznych i wartości zwracanych parametrów. Poniżej podano standardową ramkę aktywacji:

```
void FBACT_xxx (
uint16 hinstance,   /* "xxx" jest nazwą bloku funkcyjnego */
                    /* numer logiczny instancji */
str_data *data,     /*data: wskaźnik do struktury danych instancji */
str_arg *arg        /* wskaźnik do struktury par. wywołania */
)
{
}
```

Argument "**hinstance**" jest numerem logicznym instancji. Jest zastrzeżony dla wewnętrznych operacji ISaGRAF i nie powinien być użyty do programowania tej operacji. Argument "**data**" jest wskaźnikiem do struktury danych związanej z instancją. Argument "**arg**" jest wskaźnikiem do struktury zawierającej wartość parametrów wywołania. Aby mieć dostęp do pól struktury "**arg**" programista powinien stosować identyfikatory zdefiniowane w nagłówku *bloku funkcyjnego* "C".

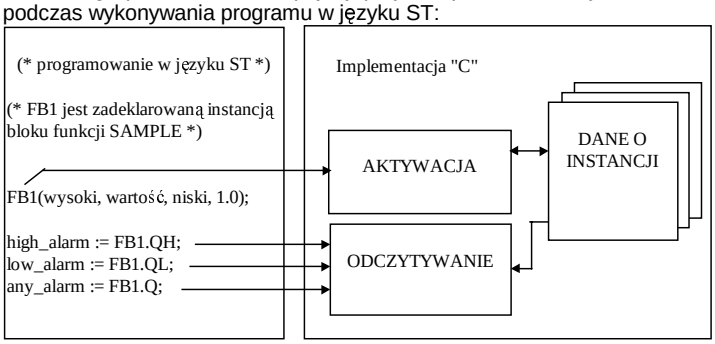
Algorytm "aktywacji" przetwarza parametry wywołania (zapamiętane w strukturze "**arg**") i uaktualnia pola struktury "**data**". Następujący przykład przedstawia "aktywizację" *bloku funkcyjnego* **TRIG** (wykrywania zbocza narastającego):

```
/* definicje zapamiętane w nagłówku bloku funkcyjnego "C" */

typedef struct {       /* parametry wywołania */
        T_BOO  _clk;         /* wyzwalane wejście */
} str_arg;

#define CLK      (arg->_clk)

/* struktura danych instancji bloku funkcyjnego */

typedef struct {
        T_BOO prev_state; /* poprzedni stan wyzwolonego wejścia */
        T_BOO edge_detect;          /* wartość zbocza: obraz zwracanego par. */
} str_data;

/* aktywacja */

void FBACT_trig (uint16 hinstance, str_data *data, str_arg *arg)
{
        data->edge_detect = (T_BOO)(CLK && !data->prev_state);
        data->prev_state = CLK;  /* parametr wywołania */
}
```

Ramka kodu źródłowego "C" tej obsługi jest automatycznie generowana przez program Biblioteki ISaGRAF podczas tworzenia bloku funkcyjnego.

Odczytywanie zwracanych parametrów

"Odczytywanie" jest wywoływane po każdym odwołaniu do wracanego parametru instancji bloku funkcyjnego w programie w języku ST lub FBD. Służy do pobrania wartości **jednego** zwracanego parametru. Następujący przykład przedstawia wywołania "odczytu" występujące podczas wykonywania programu w języku ST:



Ponieważ "odczytywanie" może być wywołane więcej niż jeden raz w danym cyklu dla tego samego parametru zwracanego lub tej samej instancji bloku funkcyjnego, w operacji tej nie

mogą być wykonywane żadne specjalne obliczenia. Służy ona jedynie do transferu ukrytych danych do aplikacji ISaGRAF. Poniżej podano standardową ramkę odczytywania:

```
/* operacja cast używana do kopiowania wartości zwracanego parametru */

#define BOO_VALUE       ((T_BOO *)value)
#define ANA_VALUE       ((T_ANA *)value)
#define REAL_VALUE      ((T_REAL *)value)
#define TMR_VALUE       ((T_TMR *)value)
#define MSG_VALUE       ((T_MSG *)value)

/* odczytywanie zwracanych parametrów: wywoływane dla każdego parametru */

void FBREAD_xxx (/* "xxx" jest nazwą bloku funkcyjnego */
uint16 hinstance,    /* numer logiczny instancji */
str_data *data,      /* wskaźnik do struktury danych instancji */
uint16 parno,        /* numer logiczny odczytanego parametru */
void *value)         /* bufor, gdzie jest kopiowana wartość par. */
{
        switch (parno) {
            case FBLPNO_XX: /* ... */ break;
            case FBLPNO_YY: /* ... */ break;
             /* .... */
        }
}
```

Argument "**hinstance**" jest numerem logicznym instancji. Jest zastrzeżony dla wewnętrznych operacji ISaGRAF i nie powinien być użyty do programowania tej obsługi. Argument "**data**" jest wskaźnikiem do struktury danych związanych z instancją.

Argument "**parno**" jest numerem logicznym tego zwracanego parametru, którego wartość jest potrzebna. Do identyfikowania zwracanych parametrów należy użyć identyfikatorów zdefiniowanych w nagłówku *bloku funkcyjnego* "C". Identyfikatory takie zaczynają się od przedrostka "**FBLPNO**_". Argument "**value**" jest wskaźnikiem do bufora, do którego jest kopiowana aktualna wartość odczytanego parametru. Typ danych wskazywany przez ten argument zależy od typu zwracanego parametru ISaGRAF. Następująca tabela podaje zależności pomiędzy typami danych ISaGRAF a typami danych "C" w buforze:

Binarna	Long integer	Słowo 32 bitowe bez znaku: 1=prawda/0=fałsz
Analogowa	Long integer	Słowo 32 bitowe ze znakiem
Real	Liczba zmienno-przecinkowa	Słowo 32 bitowe zmiennoprzecinkowe pojedynczej precyzji
Timer	Long integer	Słowo 32 bitowe bez znaku (jednostką jest 1ms)
Komunikat	znak *	tablica znaków

Dostęp do bufora jest uzyskiwany poprzez następujące makra zgodnie z typem odczytywanego parametru zwracanego:

```
#define   BOO_VALUE      ((T_BOO *)value)
#define   ANA_VALUE      ((T_ANA *)value)
#define   REAL_VALUE     ((T_REAL *)value)
```

```
#define   TMR_VALUE       ((T_TMR *)value)
#define   MSG_VALUE       ((T_MSG *)value)
```

Do kopiowanie wartości lub parametru do bufora ISaGRAF służą powszechnie stosowane zaprogramowane operacje:

```
/* dla parametru dwustanowego: */
  *BOO_VALUE = parameter_value;
/* dla parametru analogowego typu integer: */
  *ANA_VALUE = parameter_value;
/* dla parametru real: */
  *REAL_VALUE = parameter_value;
/* dla parametru Timer: */
  *TMR_VALUE = parameter_value;
/* dla parametru w postaci ciągu znaków: */
  strcpy (*MSG_VALUE, parameter_value);
```

Ramka kodu źródłowego "C" tej usługi jest automatycznie generowana przez program Biblioteki ISaGRAF podczas tworzenia *bloku funkcyjnego*.
Przykład pliku źródłowego w języku "C"
Poniżej przedstawiono standardową ramkę implementacji bloku funkcyjnego "C":

```
/* blok funkcyjny (xxx jest nazwą bloku funkcyjnego) */

#include <tasy0def.h>
#include <grfb0nnn.h>   /* nnn jest numerem bloku funkcyjnego w bibliotece */

/* struktura ukrytych danych dla każdej instancji bloku */
typedef struct {
        /* definicja pól */
} str_data;

/* inicjalizacja: zwracana jest wielkość potrzebnych ukrytych danych */
word FBINIT_xxx (uint16 hinstance)
{
        return (sizeof (str_data));
}

/* aktywacja: przetwarzane są parametry wywołania */
void FBACT_xxx (uint16 hinstance, str_data *data, str_arg *arg)
{
        /* ... */
}

/* operacja cast służąca do kopiowania wartości zwracanego parametru */
#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* odczytywanie zwracanych par.: wywoływane dla każdego par. zwracanego */
```

```
void FBREAD_xxx (uint16 hinstance, str_data *data, uint16 parno, void *value)
{
        switch(parno)
        {
          case FBLPNO_XX: *???_VALUE = ...; break;
          case FBLPNO_YY: *???_VALUE = ...; break;
        ....

}
```

/* Następująca funkcja jest używana do inicjalizacji bloku funkcyjnego i deklaracji jego wykonania. Nazwa bloku funkcyjnego stanowi połączenie z jądrem ISaGRAF. Operacja ta jest całkowicie generowana przez program Biblioteki ISaGRAF. */

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");
        *initproc = (IBP)FBINIT_xxx;
        *readproc = (RBP)FBREAD_xxx;
        return ((ABP)FBACT_xxx);
}

/* koniec pliku */
```

"**TASY0DEF.H**" zawiera plik jądra ISaGRAF , który jest potrzebny do definicji systemowych. Obejmuje też definicję typów danych stanowiących wskaźnik do implementowanych operacji.

- Połączenia pomiędzy projektami a implementacją "C"

Logicznym łącznikiem pomiędzy implementacją bloku funkcyjnego "C" a jego zastosowaniem w programach projektu ISaGRAF jest nazwa funkcji. Do kodu źródłowego bloku funkcyjnego w języku "C" jest dodawana "deklaracja". Operacja ta jest wywoływana tylko jeden raz w chwili uruchamiania aplikacji i wskazuje jądru ISaGRAF nazwę bloku funkcyjnego "C" odpowiadającej implementowanym operacjom. Poniżej przedstawiono standardowy format takiej deklaracji:

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");                   /* nazwa bloku funkcyjnego */
        *initproc = (IBP)FBINIT_xxx;            /* usługa inicjalizacji */
        *readproc = (RBP)FBREAD_xxx;                    /* usługa odczytywania */
        return ((ABP)FBACT_xxx);                /* usługa aktywacji */
}
/* xxx jest nazwą bloku funkcyjnego */
```

Nazwa bloku funkcyjnego używana w instrukcji **strcpy** musi być napisana **dużymi literami**. Należy ją pisać małymi literami w nazwie implementowanych operacji i w nazwie deklaracji.

Stosowanie przedrostków "**FBACT**_", "**FBINIT**_", "**FBREAD**_" i "**fbldef**_" w nazwach zaimplementowanych operacji i w definicji umożliwia użytkownikowi nazwanie bloku funkcyjnego zastrzeżonym słowem kluczowym języka "C" lub nazwą funkcji istniejącej w bibliotekach ISaGRAF. Do deklaracji nie można dodać żadnej innej instrukcji.

Operacja deklaracji jest wywoływana dla każdego zintegrowanego bloku funkcyjnego "C" nawet, jeśli nie jest on używany w programach aplikacji ISaGRAF. Jeśli używany w aplikacji blok funkcyjny "C" nie jest zintegrowany z jądrem, to jądro ISaGRAF wykryje błąd krytyczny.

Przed połączeniem nowych bloków funkcyjnych z jądrem użytkownik musi napisać inny program źródłowy w języku "C" z nazwie " **GRFB0LIB.C**" i umieścić go (wraz z blokami funkcyjnymi) na liście plików do włączenia. Program " **GRFB0LIB.C**" zawiera tylko tablicę operacji deklaracji. Tablica ta jest odczytywana podczas inicjowania aplikacji w celu stworzenia dynamicznego połączenia z blokami funkcyjnymi napisanymi w języku "C". Poniżej podano przykład takiego pliku:

```
/* Plik: grfb0lib.c – bloki zaimplementowanych funkcji */

#include <tasy0def.h>

extern ABP fbldef_fb1(char *name, IBP *init, RBP *read);
extern ABP fbldef_fb2(char *name, IBP *init, RBP *read);

FBL_LIST FBLDEF[ ] = {
        fbldef_fb1,
        fbldef_fb2,

NULL };

/* koniec pliku */
```

Tablica **FBLDEF** musi być zakończona wskaźnikiem NULL. Niespełnienie tego warunku spowoduje pewne problemy. Niezdefiniowanie tablicy **FBLDEF** spowoduje wystąpienie niezdefiniowanych odsyłaczy przy dołączaniu nowego jądra ISaGRAF.

Napisanie tego pliku umożliwi zbudowanie nowego jądra łącznie ze wszystkimi istniejącymi blokami funkcyjnymi. Można też zbudować jądro przystosowane do realizacji jednego projektu. W tym celu w tablicy **FBLDEF** należy umieścić tylko te bloki funkcyjne, które są używane w projekcie. Plik " **GRFB0LIB.C**" jest automatycznie generowany przez Generator Kodu ISaGRAF podczas tworzenia kodu aplikacji. Plik jest umieszczany w katalogu projektów ISaGRAF i grupuje tylko te funkcje, które są używane w projekcie.

- Ograniczenia

Biblioteka ISaGRAF może zawierać do **255** bloków funkcyjnych "C". Funkcja może być użyta do wykonania dowolnego typu operacji. Każdy typ bloku funkcyjnego może być kopiowany (użyty) w danym projekcie aż **255** razy.

Należy zwrócić uwagę, że usługi związane z blokami funkcyjnymi są wywoływane w cyklu ISaGRAF w sposób **synchroniczny**, a więc wykonanie bloku ma bezpośredni wpływ na parametry czasowe cyklu.

- Kompletny przykład

Poniżej podano kompletny program bloku funkcyjnego "**Sample**" wykonującego zliczanie w górę.

Opis techniczny bloku funkcyjnego jest następujący:

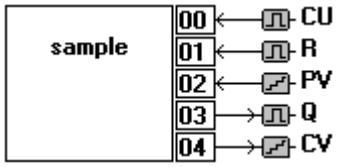
nazwa: SAMPLE
opis: Licznik zliczający w górę

data utworzenia: 01 lutego 1994
autor: CJ international

par. wywołania: CU : wejście zliczające
R : komenda reset
PV : wartość maksymalna zaprogramowana
par. zwracane: Q : wartość maksymalna wykryta
CV : wynik zliczania

prototype: SAMPLE (count, reset_command, maximum_value);
max_detect := SAMPLE.Q;
count_result := SAMPLE.CV;

Poniżej podano interfejs bloku funkcyjnego:



Nagłówek źródłowy w języku "C" bloku funkcyjnego wygląda następująco:

```
/* interfejs bloku funkcyjnego - nazwa: SAMPLE */

/* definicja standardowych typów danych  ISaGRAF */

typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definicja struktury parametrów wywołania */

typedef struct {
  T_BOO  _cu;
  T_BOO  _r;
  T_ANA  _pv;
} str_arg;

/* identyfikatory służące do korzystania z  parametrów wywołania */

#define CU          (arg->_cu)
#define R           (arg->_r)
#define PV          (arg->_pv)
```

```
/* numeracja logiczna zwracanych parametrów */

#define FBLPNO_Q        0
#define FBLPNO_CV       1

/* koniec pliku */
```

Poniżej podano kod źródłowy bloku funkcyjnego w języku "C". Ręcznie są wprowadzane przez programistę tylko wiersze wydrukowane tłustym drukiem.

```
/* blok funkcyjny - nazwa: SAMPLE */

#include <tasy0def.h>       /* potrzebna do zdefiniowania typów danych */
#include <grfb0255.h>       /* nagłówek źródłowy bloku funkcyjnego C */

/* definicja struktury zawierającej dane dla jednej instancji */

typedef struct {
        T_BOO overflow;  /*prawda: wart.zliczana>=wart. zaprogram. */
        T_ANA value;     /* aktualna wartość zliczana */
} str_data;

/* inicjalizacja: wymaga pamięci dla danych instancji */

word FBINIT_sample (uint16 hinstance)
{
        return (sizeof (str_data));
}

/* aktywacja: algorytm zliczania w górę */

void FBACT_sample (uint16 hinstance, str_data *data, str_arg *arg)
{
        if (R) data->value = 0;
        else if (CU && data->value < PV) (data->value)++;
        data->overflow = (data->value >= PV) ? (T_BOO)1 : (T_BOO)0;
}

/* operacja cast potrzebna do kopiowania parametrów do bufora ISaGRAF */

#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* odczytywanie: pobieranie wartości jednego zwracanego parametru */

void FBREAD_sample (uint16 hinstance, str_data *data, uint16 parno, void *value)
{
        switch (parno) {
```

```
            case FBLPNO_Q   : *BOO_VALUE = data->overflow; break;
            case FBLPNO_CV : *ANA_VALUE = data->value; break;
          }
}

/* usługa deklaracji służąca do dynamicznego połączenia z jądrem ISaGRAF */

ABP fbldef_sample (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "SAMPLE");
        *initproc = (IBP)FBINIT_sample;
        *readproc = (RBP)FBREAD_sample;
        return ((ABP)FBACT_sample);
}

/* koniec pliku */
```

C.7.5 Techniki kompilowania i łączenia

Pakiet ISaGRAF nie zawiera żadnego kompilatora języka "C" ani konsolidatora. W rozdziale tym opisano jednak główne techniki, jakie można zastosować do łatwego wykorzystania plików stworzonych przez program Biblioteki ISaGRAF i do przekazania ich do innych narzędzi takich, jak kompilatory i konsolidatory.

- Pliki źródłowe w języku "C"

Pliki źródłowe w języku "C" zawierające konwersje, funkcje i bloki funkcyjne są umieszczane przez program Biblioteki ISaGRAF w katalogach **ISAWIN\LIB\DEFS** i **ISAWIN\LIB\SRC**. Nazwa pliku źródłowego jest kojarzona w bibliotece ISaGRAF z numerem odpowiadającej mu konwersji, funkcji lub bloku funkcyjnego. Stosowane są następujące nazwy plików:

\isawin\lib\defs\TACN0DEF.H **plik definicyjny funkcji konwersji**
\isawin\lib\src\GRCN0nnn.H **plik źródłowy funkcji konwersji**
\isawin\lib\defs\GRUS0nnn.H **plik definicyjny funkcji**
\isawin\lib\src\GRUS0nnn.C **plik źródłowy funkcji**
\isawin\lib\defs\GRFB0nnn.H **plik definicyjny bloku funkcyjnego**
\isawin\lib\src\GRFB0nnn.C **plik źródłowy bloku funkcyjnego**
(**nnn** jest numerem konwersji, funkcji lub bloku funkcyjnego)

Ostrzeżenie: Zmieniając nazwę lub kopiując elementy biblioteki ISaGRAF nie uaktualnia tekstu ani wierszy programu stosownie do nowej nazwy i numeru logicznego elementu. Zmiany te muszą zostać dokonane ręcznie w pliku źródłowym w języku "C".

Plik **\ISAWIN\LIB\USPNUMS** podaje zależności pomiędzy nazwami i numerami logicznymi funkcji "C" istniejących w bibliotece ISaGRAF. Może to być na przykład taki plik:

```
1       funct_A
10      funct_B
16      funct_C
```

Plik **\ISAWIN\LIB\FBLNUMS** podaje zależności pomiędzy nazwami i numerami logicznymi bloków funkcyjnych "C" istniejących w bibliotece ISaGRAF. Może to być na przykład taki plik:

0 fbl_A
1 fbl_B
2 fbl_C

Plik **\ISAWIN\LIB\CNVNUMS** podaje zależności pomiędzy nazwami i numerami logicznymi funkcji konwersji istniejących w bibliotece ISaGRAF. Poniżej podano w charakterze przykładu, zawartość takiego pliku funkcji konwersji ze standardowej biblioteki:

0 SCALE
1 BCD

Pliki te są automatycznie uaktualniane przez program Biblioteki ISaGRAF po każdym utworzeniu, skopiowaniu, usunięciu lub zmianie nazwy konwersji, funkcji lub bloku funkcyjnego. Podczas budowania aplikacji Generator Kodu ISaGRAF automatycznie generuje następujące pliki:

\isawin\apl\ppp\GRCN0LIB.C Deklaracja jako tablica wszystkich funkcji konwersji stosowanych w projekcie.
\isawin\apl\ppp\GRUS0LIB.C Deklaracja jako tablica wszystkich funkcji stosowanych w projekcie.
\isawin\apl\ppp\GRFB0LIB.C Deklaracja jako tablica wszystkich bloków funkcyjnych stosowanych w projekcie.

(**ppp** jest nazwą projektu ISaGRAF)

Pliki te mogą być użyte podczas operacji konsolidacji do zbudowania nowego jądra ISaGRAF przeznaczonego dla danego projektu zawierającego tylko konwersje, funkcje i bloki funkcyjne stosowane w tym projekcie.

- Ładowanie plików źródłowych do specyficznego systemu

Pliki źródłowe w języku "C" i pliki definicyjne utworzone przez program Biblioteki ISaGRAF mogą być załadowane do systemu oprogramowania wbudowanego ISaGRAF, o ile obsługuje on specyficzne narzędzie do kompilacji. Można do tego celu użyć standardowego narzędzia **TERMINAL** dostarczanego z systemem Windows.

Jeśli w oprogramowaniu wbudowanym są stosowane pliki źródłowe, to pliki definicyjne muszą być uaktualnianie operacją ponownego załadowania po każdym zmodyfikowaniu interfejsu funkcji przy pomocy programu Biblioteki ISaGRAF.

Wiersze komendy do ładowania plików mogą zostać pogrupowane na przykład w plik wsadowy a potem uruchomione z menu Narzędzia pakietu ISaGRAF (patrz Podręcznik użytkownika: Zarządzanie programami)

- Korzystanie z kompilatora dedykowanego na inne środowisko

Pliki źródłowe mogą być również obsługiwane bezpośrednio w komputerze PC użytkownika, o ile oprogramowanie wbudowane jest zainstalowane w PC lub jest dostępny kompilator dedykowany na inne środowisko pracujący na PC i generujący kod dla oprogramowania wbudowanego.

W takim przypadku użytkownik może przy pomocy programu Biblioteki ISaGRAF skompletować i zmodyfikować kody źródłowe konwersji, funkcji lub bloków funkcyjnych. Wiersze komend do uruchomienia kompilatora i konsolidatora mogą zostać pogrupowane, na

przykład w plik wsadowy, a potem wykonane z menu Narzędzia pakietu ISaGRAF (patrz Podręcznik użytkownika: Zarządzanie programami)

Gdy konwersje, funkcje i bloki funkcyjne są kompilowane na PC, to przed uruchomieniem aplikacji użytkownik musi po prostu załadować nowo wygenerowane jądro ISaGRAF (skojarzone z nowymi komponentami) do oprogramowania wbudowanego. Jeśli oprogramowanie wbudowane pracuje na innym PC, to nowo wygenerowane jądro ISaGRAF można załadować do PC z oprogramowaniem wbudowanym przy użyciu dyskietki lub poprzez sieć.

- Łączenie z bibliotekami jądra ISaGRAF

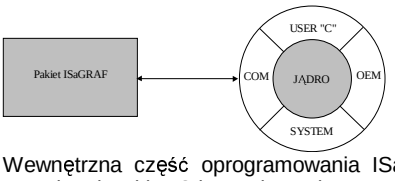
Ostrzeżenie:
Podane poniżej informacje ogólne nie muszą odpowiadać dokładnie oprogramowaniu wbudowanemu pracującemu u użytkownika. Szczegółowe informacje są podane w pliku readme i plikach .TXT dostarczonych na dysku z oprogramowaniem wbudowanym.

Dyskietka z oprogramowaniem wbudowanym ISaGRAF zawiera wiele plików usługowych do kompilowania i konsolidacji konwersji, funkcji i bloków funkcyjnych z bibliotekami jądra ISaGRAF.

Istnieją dwie implementacje ISaGRAF:
- jednozadaniowa: wszystkie funkcje są wykonywane w tym samym programie
- wielozadaniowa: komunikacja jest odrębnym zadaniem.

W każdym przypadku komponenty "C" są grupowane w tych samych bibliotekach: dla programisty "C" nie ma różnicy, czy system jest jedno czy wielozadaniowy. W przypadku wersji jednozadaniowej biblioteki użytkowe "C" są łączone z jednym zadaniem (generalnie nazywanym **isa**), podczas gdy dla wersji wielozadaniowej biblioteki są łączone z zadaniem jądra (generalnie nazywanym **isaker**).

System do pisania programów **Oprogramowanie wbudowane**

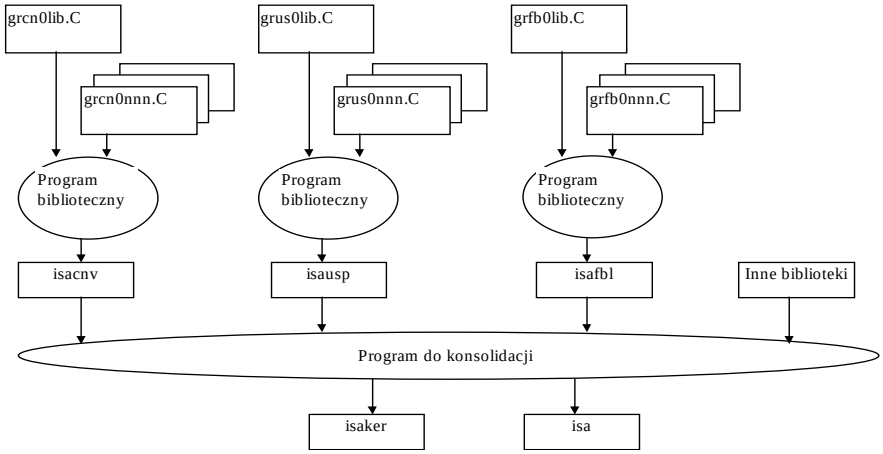


Wewnętrzna część oprogramowania ISaGRAF jest niezależna od sprzętu. Działa ona w oparciu o języki IEC i ma własną bazę zmiennych.

W procesie konsolidacji z jądrem pierwszym krokiem jest zbudowanie bibliotek wszystkich konwersji, funkcji i bloków funkcyjnych potrzebnych do realizacji określonego projektu:

Biblioteka	Zawartość
ISAUSP	- plik pośredni GRUS0LIB (tablica zadeklarowanych funkcji) - plik pośredni każdej zintegrowanej funkcji
ISAFBL	- plik pośredni GRFB0LIB (tablica zadeklarowanych bloków funkcyjnych) - plik pośredni każdego zintegrowanego *bloku funkcyjnego*
ISACNV	- plik pośredni GRCN0LIB (tablica zadeklarowanych konwersji) - plik pośredni każdej zintegrowanej funkcji konwersji

Następnie programista musi skonsolidować te nowe biblioteki z innymi plikami pośrednimi i bibliotekami jądra ISaGRAF. Różne fazy integracji dokonanych przez użytkownika opracowań w języku "C" są przedstawione na poniższym schemacie:



Poniżej przedstawiono dokładną listę modułów pośrednich i bibliotek, które muszą zostać przyłączone podczas konsolidacji:

Aby zbudować isaker:

Moduł pośredni: **tast0mai**
Moduł pośredni: **tats0com**

Biblioteka jądra: **isaker**
Biblioteka jądra: **isaoem**

Biblioteka użytkownika: **isausp** funkcje zdef. przez użytkownika
Biblioteka użytkownika: **isafbl** bloki funkcyjne zdef. przez użytk.
Biblioteka użytkownika: **isacnv** funkcje konwer. zdef. przez użytk

Biblioteka jądra: **isasys**

Biblioteki systemowe: (patrz podręcznik kompilatora języka "C")

Aby zbudować isa:

Moduł pośredni: **tast0mai**
Moduł pośredni: **tast0com**

Biblioteka jądra: **isaker**
Biblioteka jądra: **isatst**
Biblioteka jądra: **isaoem**

Biblioteka użytkownika:	**isausp**	funkcje zdef. przez użytkownika Biblioteka
użytkownika:	**isafbl**	bloki funkcji zdef. przez użytk.
Biblioteka użytkownika:	**isacnv**	funkcje konwer. zdef. przez użytk

Biblioteka jądra: **isasys**

Biblioteki systemowe: (patrz podręcznik kompilatora języka)

Programista powinien stosować podaną kolejność modułów pośrednich i bibliotek pokazaną na poprzednich rysunkach . Moduły pośrednie i biblioteki mają standardowe rozszerzenia ("**.lib**", "**.obj**", "**.l**", "**.r**"...) w zależności od oprogramowania wbudowanego.

- Niezbędne opcje kompilacji i konsolidacji

Odpowiednie opcje można wybrać podczas kompilowania i konsolidacji. Zależą one od typu operacji wykonywanych w konwersjach, funkcjach i blokach funkcyjnych. Niektóre operacje wymagają podczas konsolidacji innych bibliotek systemowych (matematycznej, graficznej...).

Wszystkie pliki źródłowe "C" jądra ISaGRAF zostały skompilowane przy użyciu modelu pamięci **LARGE**. Do skompilowania konwersji, funkcji i bloków funkcyjnych programista musi zastosować ten sam model.

Kompilowanie komponentów biblioteki "C" wymaga zdefiniowania specjalnej stałej. Stała ta wskazuje typ oprogramowania wbudowanego i procesory, aby źródła konwersji, funkcji i bloków funkcyjnych mogły być niezależne od systemu. Poniżej podane są nazwy tych wartości stałych:

DOS dla systemów pracujących pod DOS (procesor INTEL)
ISAWNT dla systemów pracujących pod Windows-NT (procesor INTEL)
OS9 dla systemu OS9 (procesor MOTOROLA)
VxWorks dla systemu VxWorks (procesor MOTOROLA)

Plik komend programu usługowego (do kompilacji i konsolidacji) dostarczony z oprogramowaniem wbudowanym ISaGRAF określa sposób definiowania odpowiedniej wartości stałej w wierszu komendy kompilatora.

- Obsługiwane kompilatory

Do opracowania konwersji, funkcji i bloków funkcyjnych oraz skonsolidowania ich z jądrem ISaGRAF można wykorzystać następujące kompilatory:

Kompilator Microsoft MSC 7.00 dla oprogramowania wbudowanego pracującego pod DOS
Kompilator Microsoft MSVC 4.00 dla oprogramowania wbudowanego pracującego pod Windows-NT
Kompilator Microware ULTRA-C dla oprogramowania wbudowanego pracującego pod OS-9
Tornado 1.0; GNU Toolkit 2.6 dla oprogramowania wbudowanego pracującego pod VxWorks

W sprawie innych kompilatorów należy się skontaktować z CJ International.

- Podsumowanie

Poniżej przedstawiono zestawienie operacji, jakie należy wykonać opracowując nową konwersję, funkcje lub blok funkcyjny.

⇨**1.** Korzystając z programu Biblioteki ISaGRAF należy utworzyć nowy element: podać jego nazwę i tekst komentarza. Ramka pliku źródłowego w języku "C" jest generowana automatycznie.

⇨**2.** Używając programu Biblioteki ISaGRAF opisać interfejs (parametry wywołania i zwracane), jeśli element jest funkcją lub blokiem funkcyjnym. Plik nagłówkowy w języku "C" jest generowany automatycznie.

⇨**3.** Przy pomocy programu Biblioteki ISaGRAF wprowadzić tekst szczegółowego opisu technicznego elementu.

⇨**4.** Przy pomocy programu Biblioteki ISaGRAF skompletować plik źródłowy w języku "C" przez wprowadzenie programu algorytmu konwersji, funkcji lub bloku funkcyjnego. W ten sposób kod źródłowy elementu będzie już kompletny. Należy zwrócić uwagę, że może być użyty inny edytor.

⇨**5.** Wybrać opcję programu Biblioteki "**Pokaż numer logiczny**", aby poznać numer logiczny skojarzony z nowym elementem. Ten numer jest podawany w nawiasach okrągłych odpowiednich plików źródłowych ".C" i ".H".

⇨**6.** Skopiować / załadować pliki .C i .H do oprogramowania wbudowanego (w przypadku kompilatora narodowego) lub do odpowiedniego środowiska (w przypadku kompilatora dedykowanego na inne środowisko), w którym są zainstalowane biblioteki i zadania oprogramowania wbudowanego ISaGRAF.

⇨**7.** Uruchomić kompilator języka "C" dla nowego pliku źródłowego i skorygować ewentualne błędy składniowe.

⇨**8.** Wprowadzić nazwę usługi deklaracji nowego elementu do pliku źródłowego "**GR??0LIB.C**" określającego tablicę wstawionych elementów.

⇨**9.** Uruchomić kompilator języka "C" w celu skompilowania pliku "**GR??0LIB.C**".

⇨**10.** Wprowadzić nazwę modułu pośredniego do listy plików pośrednich służących do zbudowania odpowiedniej biblioteki.

⇨**11.** Wykonać procedurę tworzenia biblioteki programów w języku "C". Uruchomić konsolidator języka "C" w celu utworzenia nowego jądra.

⇨**12.** Zainstalować nowo utworzone jądro w komputerze z oprogramowaniem wbudowanym.

⇨**13.** Napisać przykładową aplikację ISaGRAF, która przetestuje aktywację i interfejs nowego elementu.

C.8 Łącze Modbus

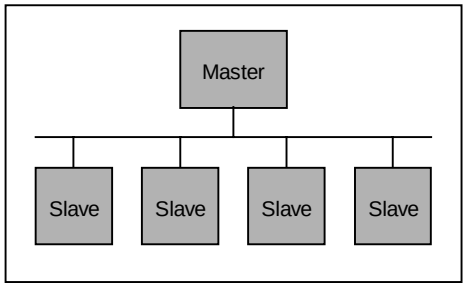
Po kompletnym opracowaniu i przetestowaniu aplikacji można włączyć ją do systemu wizualizacji procesu.

ISaGRAF jest systemem otwartym oferującym dużą różnorodność funkcji sieciowych. Najprostszą siecią przemysłową jest standardowy protokół MODBUS/MODICON, który jest dostępny w niemal każdym systemie wizualizacji procesu i do pracy którego jest potrzebne jedynie łącze szeregowe (RS232, RS485, Pętla prądowa).

Protokół komunikacyjny debagera ISaGRAF jest kompatybilny z MODBUS i dzięki temu jest możliwe odczytywanie/zapisywanie zmiennych ze stacji master Modbus.

C.8.1 Sieć i protokół MODBUS

Sieć Modbus jest zbudowana z tylko jednej stacji master (obejmującej zwykle system wizualizacji procesu) i jednej lub kilku stacji podrzędnych (zwykle sterowników PLC).



Stacja master wysyła do sterownika jednorazowo jedno żądanie (używając numeru sterownika) i oczekuje na odpowiedź ze sterownika, a dopiero później wysyła następne zapytanie. Inne nie pytane sterowniki nie odpowiadają.

Każda ramka zawiera numer sterownika, numer żądania i odpowiednie dane oraz 16-bitową sumę kontrolną (CRC).

Jeśli w ciągu ustalonego limitu czasu nie nadejdzie żadna odpowiedź, to zanim stacja master uzna sterownik za 'odłączony' pytanie jest powtarzane kilka razy. Długość limitu czasu i liczba powtórzeń muszą być ustawione w stacji master tak, aby odpowiadały wymaganiom sterownika (zależnie od aplikacji, itd.).

Jeśli w czasie przetwarzania żądania wystąpi błąd, to zamiast wysłania spodziewanej ramki odpowiedzi sterownik może przekazać komunikat o błędzie.

Modbus jest protokołem Modicon, ale nie jest standardem międzynarodowym. Jest wiele różnych implementacji protokołów kompatybilnych z 'Modbus', różniących się takimi cechami, jak:

– Lista zaimplementowanych kodów funkcji
– Mapowanie adresów
– Protokół RTU (kody binarne) lub ASCII
itd.

C.8.2 Implementacja ISaGRAF

Korzystanie ze zmiennych aplikacji

Łącze komunikacyjne ISaGRAF rozpoznaje pięć kodów funkcji Modbus:

1	czytaj N bitów
3	czytaj N słów
5	zapisz 1 bit
6	zapisz 1 słowo
16	zapisz N słów

Ze zmiennych aplikacji ISaGRAF można korzystać w oparciu o ich 'adresy sieciowe', jeśli oczywiście zostały one zdefiniowane w słowniku pakietu ISaGRAF. Mogą być stosowane następujące typy zmiennych:
- Zmienne binarne lub analogowe
- Wejścia, wyjścia lub zmienne wewnętrzne
- Zmienne lokalne lub globalne.

Aby zapisać zmienną binarną można zastosować funkcję 5, 6 lub 16. Wartością PRAWDA dla zapisu jest każda wartość niezerowa.

Aby odczytać zmienną dwustanową można zastosować funkcję 1 lub 3. Funkcja 1 wyszukuje wartości w polu bitowym, funkcja 3 wyszukuje je w bajtach (wartość PRAWDA odpowiada wielkości 0xFFFF).

Aby zapisać zmienną analogową można zastosować albo funkcję 6 albo 16. Wartość jest 16 bitową liczbą integer z przedziału od -32768 aż do +32767 (zmienne oprogramowania wbudowanego ISaGRAF są 32 bitowe).

Aby odczytać zmienną analogową należy zastosować funkcję 3. Odczytana wartość jest 16 bitową liczbą integer z przedziału od -32768 aż do +32767. W oprogramowaniu wbudowanym zmienne analogowe są 32 bitowe i dlatego w oprogramowaniu wbudowanym wielkość przekraczająca długością 16 bitów (dodatnia lub ujemna) zostanie odczytana jako wartość o maksymalnej długości 16 bitów (dodatnia lub ujemna).

W implementacji Modbus nie można korzystać ze zmiennych typu real.

Ostrzeżenie:
Implementacja ISaGRAF nie obsługuje kodów błędów takich, jak ‘nieznany adres Modbus’.

Oznaczenia umowne:

slv	numer sterownika
nbw	liczba słów
nbb	liczba bajtów
nbi	liczba bitów
addH	adres sieciowy (starszy bajt)
addL	adres sieciowy (młodszy bajt)
vH	wartość (starszy bajt)
vL	wartość (młodszy bajt)
V	wartość w bajtach
bfd	postać bitowa (nbb bajtów)
crcH	suma kontrolna (starszy bajt)
crcL	suma kontrolna (młodszy bajt)

FUNKCJA 1: czytaj N bitów

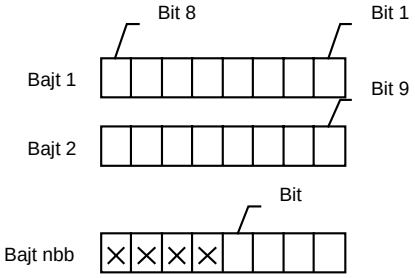
Odczytuje nbi bitów (dwustanowo) zaczynając od adresu sieciowego addH/addL

Pytanie	slv	01	addH	addL	00	nbi	crcH	crcL

Odpowiedź	slv	01	nbb	bfd	...		crcH	crcL

Bajt 1 Bajt nbb

bfd jest polem bitowym złożonym z nbb bajtów o następującym formacie:



Bit 1 odpowiada wartości zmiennej pod adresem sieciowym addH/addL.
Bit nbi odpowiada wartości zmiennej pod adresem sieciowym addH/addL + nbi -1.
X oznacza wartość nieokreśloną.

FUNKCJA 3: czytaj N słów

Odczytuje nbw słów zaczynając od adresu sieciowego addH/addL

Pytanie	slv	03	AddH	addL	00	nbw	crcH	crcL

Odpowiedź .	slv	03	Nbb	vH	vL	...	crcH	crcL

nbb odpowiada liczbie vH, vL bajtów

FUNKCJA 5: zapisz 1 bit

Zapisuje jeden bit (dwustanowo) pod adresem sieciowym addH/addL

Pytanie	slv	05	addH	addL	vH	00	crcH	crcL

Odpowiedź	slv	05	addH	addL	vH	00	crcH	crcL

FUNKCJA 6: zapisz 1 słowo

Zapisuje jedno słowo pod adresem sieciowym addH/addL

Pytanie	slv	06	addH	addL	VH	vL	crcH	crcL

Odpowiedź	slv	06	addH	addL	VH	vL	crcH	crcL

FUNKCJA 16: zapisz N słów

Zapisuje nbw słów w pamięci zaczynając od adresu sieciowego addH/addL (nbb = 2nbw)

Pytanie	slv	10	addH	addL	00	nbw	Nbb	vH	vL	...	crcH	crcL

Odpowiedź	slv	10	addH	addL	00	nbw	CrcH	crcL

Przykłady:

Funkcja 1: odczyt 15 bitów zaczynając od adresu sieciowego 0x1020 w sterowniku 1

Pytanie	01	01	10	20	00	0F	79	04

Odpowiedź	01	01	02	00	12	39	F1

Odczytaną wartością jest 0x0012, co daje 00000000 00010010 w postaci bitowej. W tym przykładzie zmienne 0x1029 i 0x102C są typu PRAWDA, wszystkie inne są FAŁSZEM.

Funkcja 16: zapis 2 słów pod adresem 0x2100 w sterowniku 1, zapisanymi wartościami są 0x1234 i 0x5678.

Pytanie	01	10	21	00	00	02	04	12	34	56	78	1C	CA

Odpowiedź	01	10	21	00	00	02	4B	F4

- Przekazywanie plików

W porównaniu do nowoczesnych magistrali protokół Modbus oferuje zestaw bardzo ubogich usług, jeśli nie zostanie rozszerzony o kody funkcji określonych producentów.

W naszej sytuacji wykonywanie ISaGRAF na bazie komputerów o dużej mocy obliczeniowej podlega dwóm ograniczeniom dotyczącym protokołu Modbus:
Możliwe jest korzystanie tylko ze zmiennych ISaGRAF
Trudno jest dokonać szybkiego transferu dużych ilości danych

Z tego powodu ISaGRAF oferuje zbiór żądań przesłania plików podobnych do tych, jakie realizuje Modbus oraz protokół 'zdalnego zarządzania plikami'. Własności te zostały zrealizowane w systemie ISaGRAF, aby umożliwić:
Ładowanie plików binarnych lub ASCII
Pobieranie plików binarnych lub ASCII
Dynamiczną wymianę danych poprzez pliki udostępniane wirtualnie lub fizycznie.

Dlatego przy pomocy łącza komunikacyjnego ISaGRAF każda aplikacja « niezależna od ISaGRAF » może z łatwością komunikować się ze zdalnym oprogramowaniem wbudowanym.

Protokół jest oparty na następujących założeniach:
Plik po stronie oprogramowania wbudowanego ISaGRAF jest nazywany **plikiem zdalnym**
Plik w komputerze master jest nazywany **plikiem lokalnym**
Każdy bajt w dowolnym pliku jest udostępniany w oparciu o 32 bitowy **adres bazowy** plus 16 bitowy **adres bajtowy**.

Żądania wybrania nazwy pliku zdalnego, wybrania adresu bazowego, odczytu lub zapisu danych z pliku zdalnego są wykonywane w oparciu o 16 bitowy adres bajtowy.

FUNKCJA 17: zapisz dane

nbb odpowiada liczbie vH, vL bajtów

Pytanie	slv	11	addH	addL	00	nbb	nbb	vH	vL	...	CrcH	crcL

Odpowiedź	slv	11	addH	addL	00	nbb	crcH	crcL

Znaczenie tego żądania jest różne zależnie od przedziału adresowego addH/addL:

0xF000: Zainicjuj nazwę pliku zdalnego
nbb odpowiada liczbie znaków w nazwie pliku określonej w polach vH vL (w tym przypadku bajty starszy i młodszy są nieznaczące) **łącznie z \0** dla oznaczenia końca ciągu. Jeśli taki plik nie istnieje, to zostanie utworzony wraz z atrybutami zapisu + odczytu + wykonywania.

0xF002: Zmień adres bazowy na określoną wartość
nbb powinno być równe 4. Pierwszy bajt vH/vL odpowiada starszemu słowu określonej wartości. Dozwolona jest dowolna wartość 32 bitowa.
Ten adres bazowy będzie wykorzystywany przez wszystkie przyszłe żądania odczytu lub zapisu. Gdy żądania to nie zostanie zastosowane to domyślnym adresem bazowym będzie zero.

0xF004: Usuń plik
nbb powinno być równe zero.

Plik zostanie wykryty o ile istnieje i o ile jest to możliwe.

Większy od 0xF004: Zastrzeżony

Mniejszy od 0xF000: Zapisz bajty
Określony adres, pod który należy zapisać bajty, jest podany w addH/addL. Musi on być mniejszy od F000. Określone bajty (nbb bajtów określonych w polach vH vL, gdzie bajty starsze i młodsze mogą już dłużej nie być znaczące) są zapisywane w podanej kolejności (z lewej do prawej) i oznaczone poprzednio wybraną nazwą pliku zdalnego. Początkowym adresem, pod który dane są zapisane, jest określony adres dodany do poprzednio wybranego adresu bazowego. Jeśli uzyskane w rezultacie zaadresowane pole przekracza aktualny rozmiar pliku, to plik zostanie powiększony. Nie można zmniejszyć rozmiaru pliku.

FUNCTION 18: czytaj dane

Pytanie	slv	12	addH	addL	00	nbb	crcH	crcL

Odpowiedź	slv	12	nbb	V	V	...	crcH	crcL

Określony adres, spod którego należy odczytać bajty, jest podany w addH/addL. Musi on być mniejszy od F000. Funkcja odczytuje określoną liczbę bajtów (nbb) z pliku zdalnego o poprzednio wybranej nazwie, zaczynając od określonego adresu (addH/addL z dowolną 16 bitową wartością) dodanego do poprzednio wybranego adresu bazowego. Wartości są pobierane (V pól od lewej do prawej) w kolejności ich odczytywania w pliku.

Przykład:

Wybór nazwy pliku zdalnego: 'target.fil'.

Pytanie	01	11	F0	00	00	0B	0B	74	...	00	25	9F

Odpowiedź	01	11	F0	00	00	0B	8F	0E

Wybór adresu bazowego: 0x10000.

Pytanie	01	11	F0	02	00	04	04	00	01	00	00	76	11

Odpowiedź	01	11	F0	02	00	04	6E	CA

Zapis 4 bajtów: adres bezwzględny 0x107D0, wartości 01,02,03,04.

Pytanie	01	11	07	D0	00	04	04	01	02	03	04	28	6F

Odpowiedź	01	11	07	D0	00	04	FC	87

Odczyt 4 bajtów: adres bezwzględny 0x107D0.

Pytanie	01	12	07	D0	00	04	B8	87

Odpowiedź	01	12	04	01	02	03	04	58	7D

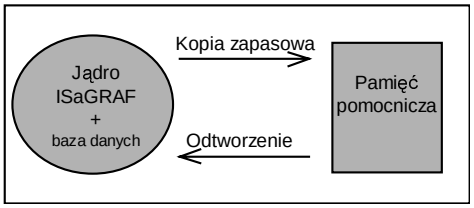
C.9 Postępowanie w przypadku awarii zasilania

C.9.1 Informacje podstawowe

Postępowanie w przypadku awarii zasilania ma dla aplikacji niezmiernie istotne znacznie z trzech powodów:
– Zależy od parametrów technicznych procesu
– Zależy od możliwości sprzętowych
– Zależy od metod programowania

Dlatego sposób reagowania ISaGRAF w przypadku awarii zasilania nie jest kompletną czy bezwzględnie uniwersalną metodą, ale jest zbiorem zasad, metod i narzędzi, jakie trzeba połączyć w szczególny sposób dla każdej aplikacji lub przynajmniej dla określonego zestawu sprzętowego.

Aby umożliwić poprawne ponowne uruchomienie systemu sterowania procesem po wystąpienie awarii zasilania, należy rozwiązać trzy problemy:



– Wykonywanie kopii zapasowych danych
– Wykrywanie przy uruchamianiu faktu wystąpienia awarii zasilania
– Odtwarzanie danych z kopii zapasowych

Drugi problem nie może mieć standardowego rozwiązania programowego, ale dostawca systemu może zapewnić niezbędne narzędzia umożliwiające dostęp do danych o stanie sprzętu z aplikacji ISaGRAF lub programu w języku C.

Ponadto ważne jest podjęcie decyzji o tym, które dane należy zachować i potem odtworzyć. Zdefiniujmy dwa rodzaje takich danych:
– Zmienne aplikacji
  Są to zmienne technologiczne takie, jak liczba przetwarzanych elementów, data wystąpienia awarii zasilania, wartości parametrów aplikacji, itd.
  Są to też zmienne programowe takie, jak liczniki, parametry czasowe, wartości pośrednie i flagi.

– Dane o stanie programu:
  Są to takie dane, jak wykaz aktywnych kroków, status każdego programu w języku C, itd.

W kolejnych rozdziałach są analizowane oba rodzaje danych i możliwości ich zachowywania przez ISaGRAF.

C.9.2 Kopie zapasowe zmiennych aplikacji

Zachowywane zmienne

Edytor zmiennych pakietu ISaGRAF oferuje możliwość wyboru atrybutu ‘zachowania’ każdej zmiennej wewnętrznej (nie będącej zmienną We/Wy).

Pod koniec każdego cyklu pracy oprogramowania wbudowanego wartości zachowywanych zmiennych są kopiowane do określonego miejsca w pamięci. Tym miejscem w pamięci jest generalnie podtrzymywana akumulatorami pamięć RAM.

Przy uruchamianiu, jeśli przynajmniej jedna zmienna ma atrybut "zachowania", to ISaGRAF szuka zachowywanych zmiennych:
- Jeśli poprzednio wykonywana była ta sama aplikacja, ISaGRAF rozpoznaje zapamiętane wartości i przyporządkowuje je każdej ‘zachowanej’ zmiennej.
- Jeśli poprzednio wykonywana była inna aplikacja lub nie była wykonywana żadna, to ISaGRAF rozpoznaje, że ‘zachowane’ wartości nie są prawidłowe i ustawia wszystkie ‘zachowywane’ zmienne na zera.

Dane dotyczące obszaru pamięci służącego do odtworzenia różnych typów zmiennych są określone w pakiecie ISaGRAF, w menu **Kompilacja**: **opcja pracy aplikacji; zachowywane zmienne.**
Określony łańcuch musi mieć następujący format:

boo_add , boo_size , ana_add , ana_size , tmr_add , tmr_size , msg_add , msg_size

gdzie:

boo_add: Adres szesnastkowy służący do zapamiętywania zmiennych dwustanowych. Zawsze musi być różny od zera.

boo_size: Podany szesnastkowo, w bajtach, rozmiar pamięci dostępnej pod tym adresem. Do zapamiętania jednej zmiennej dwustanowej potrzebny jest jeden bajt pamięci.

ana_add: Adres szesnastkowy służący do zapamiętywania zmiennych analogowych. Zawsze musi być różny od zera.

ana_size: Podany szesnastkowo, w bajtach, rozmiar pamięci dostępnej pod tym adresem. Do zapamiętania zmiennych analogowych zawsze potrzeba minimum cztery bajty plus cztery bajty na każdą zmienną.

tmr_add: Adres szesnastkowy służący do zapamiętywania zmiennych typu timer. Zawsze musi być różny od zera.

tmr_size: Podany szesnastkowo, w bajtach, rozmiar pamięci dostępnej pod tym adresem. Do zapamiętania każdej zmiennej timer potrzeba pięć bajtów.

msg_add: Adres szesnastkowy służący do zapamiętywania zmiennych w postaci komunikatów. Zawsze musi być różny od zera.

msg_size: Podany szesnastkowo, w bajtach, rozmiar pamięci dostępnej pod tym adresem. Do zapamiętania każdej zmiennej tekstowej potrzeba 256 bajtów.

Wymagania:
Muszą zostać określone wszystkie pola wszystkich typów nawet, jeśli może nie być potrzeby wykonywania kopii zapasowych wszystkich typów zmiennych. W takim przypadku dla każdego niepotrzebnego typu zmiennej trzeba określić zerową wielkość pola (oprócz zmiennych analogowych, dla których trzeba określić pola wielkości czterech bajtów) i dowolny adres różny od zera.

Przykład:
Przypuśćmy, że trzeba wykonać kopię zapasową następujących zmiennych:
20 zmiennych dwustanowych
0 zmiennych analogowych
0 zmiennych timer
3 zmienne w formie komunikatu
Pamięć pomocnicza, w której są zapisywane kopie zmiennych jest zlokalizowana pod adresem szesnastkowym 0xA2F200.
Przypuśćmy, że:
Zmienne dwustanowe zostaną zapamiętane pod adresem 0xA2F200 w polu wielkości dokładnie 20 bajtów.
Zmienne analogowe wymagają minimum po 4 bajty i zostaną zapamiętane pod adresem 0xA2F214.
Pustym adresem zmiennej timer będzie 0xA2F200 a pole będzie miało rozmiar zero.
Komunikaty zostaną zapamiętane pod adresem 0x A2F218 w polu, którego wymagana wielkość wynosi dokładnie 256*3 bajty.
Wówczas ciąg wprowadzony w pakiecie ISaGRAF powinien mieć postać:

A2F200,14,A2F214,4,A2F200,0,A2F218,300

- Wywołanie funkcji SYSTEM

Jeśli większość zmiennych aplikacji musi zostać zapamiętywana, to aby obsłużyć pełny zbiór zmiennych powinny zostać użyte mechanizmy funkcji SYSTEM (więcej informacji o funkcji SYSTEM podano w Podręczniku użytkownika). Proszę zwrócić uwagę, że wykonywaniem kopii zapasowych i odtwarzaniem wartości zmiennych zajmuje się programista na poziomie aplikacji.

Przede wszystkim trzeba zdefiniować miejsce w pamięci, w którym będą zapamiętywane kopie zapasowe określonego typu zmiennych lub wszystkich typów zmiennych:

<new_address> := SYSTEM(SYS_INITxxx,<address>);

gdzie:
<address> jest adresem pamięci pomocniczej (wielkość 16# oznacza format szesnastkowy). Musi to być adres parzysty, aby operacja została wykonana.
SYS_INITxxx może mieć następującą postać:
SYS_INITBOO, aby zdefiniować lokalizację pamięci pomocniczej dla wszystkich zmiennych dwustanowych.
SYS_INITANA, aby zdefiniować lokalizację pamięci pomocniczej dla wszystkich zmiennych analogowych.
SYS_INITTMR, aby zdefiniować lokalizację pamięci pomocniczej dla wszystkich zmiennych typu timer.
SYS_INITALL, aby zdefiniować lokalizację pamięci pomocniczej dla wszystkich zmiennych dwustanowych, analogowych i timer.

<new_address> podaje następny wolny adres, to znaczy <address> + wielkość zmiennych zachowywanych (w bajtach) zgodnie z SYS_INITxxx. Umożliwia to sprawdzenie wielkości potrzebnej pamięci pomocniczej. Jeśli operacja nie zakończy się pomyślnie, to <new_address> jest ustawiany na zero.

Następnie należy wykonać kopie zapasowe. Procedurę wywoływania kopii zapasowej można wywołać w dowolnym czasie wykonywani aplikacji, a kopia zapasowa zostanie wykonana pod koniec aktualnego cyklu i tylko jeden raz. Jeśli ze sprzętu zostanie nadejdzie sygnał wejściowy binarny lub funkcja C informująca użytkownika o terminie wystąpienia awarii zasilania i do awaryjnego zatrzymania ISaGRAF pozostanie przynajmniej jeden cykl opóźnienia, to kopia zapasowa będzie mogła zostać wykonana po wykryciu awarii zasilania:

<error> :=SYSTEM(SYS_SAVxxx,0);

gdzie:
SYS_SAVxxx może mieć następującą postać:
SYS_SAVBOO, aby zostały wykonane kopie zapasowe wszystkich zmiennych dwustanowych.
SYS_SAVANA, aby zostały wykonane kopie zapasowe wszystkich zmiennych analogowych.
SYS_SAVTMR, aby zostały wykonane kopie zapasowe wszystkich zmiennych timer.
SYS_SAVALL, aby zostały wykonane kopie zapasowe wszystkich zmiennych dwustanowych, analogowych i timer.
<error> podaje status błędu różny od zera, gdy operacja nie została wykonana pomyślnie (nie wywołano SYS_INITxxx).

Na koniec trzeba odtworzyć zachowane zmienne. Procedura ta może zostać wywołana w dowolnym czasie wykonywania aplikacji a odtworzenie wartości zmiennych nastąpi pod koniec aktualnego cyklu i tylko jedne raz. Aby zapewnić poprawność zachowanych danych, należy pewną zmienną analogową ustawić na stałą wartość uważaną za sygnaturę:

<error> := SYSTEM(SYS_RESTxxx,0);

gdzie:
SYS_RESTxxx może mieć następującą postać:
SYS_RESTBOO, aby odtworzyć wszystkie zmienne dwustanowe.
SYS_RESTANA, aby odtworzyć wszystkie zmienne analogowe.
SYS_RESTTMR, aby odtworzyć wszystkie zmienne timer.
SYS_RESTALL, , aby odtworzyć wszystkie zmienne dwustanowe, analogowe i timer.
<error> podaje status błędu różny od zera, gdy operacja nie została wykonana pomyślnie (nie wykonano (SYS_INITxxx).

Poniżej przedstawiono zestawienie komend funkcji SYSTEM obsługującej kopie zapasowe zmiennych:

Komenda		**Znaczenie**
Ustalone słowo kluczowe	**Wartość**	
SYS_INITBOO	16#20	inicjalizuj kopię zapasową zm. dwustan.
SYS_SAVBOO	16#21	zapamiętaj zmienne dwustanowe
SYS_RESTBOO	16#22	odtwórz zmienne dwustanowe
SYS_INITANA	16#24	inicjalizuj kopię zapasową zm. analog.
SYS_SAVANA	16#25	zapamiętaj zmienne analogowe
SYS_RESTANA	16#26	odtwórz zmienne analogowe

SYS_INITTMR	16#28	inicjalizuj kopię zapasową zm. timer
SYS_SAVTMR	16#29	zapamiętaj zmienne timer
SYS_RESTTMR	16#2A	odtwórz zmienne timer
SYS_INITALL	16#2C	inicjalizuj kopie zap. wszystkich typów
SYS_SAVALL	16#2D	zapamiętaj wszystkie typy
SYS_RESTALL	16#2E	odtwórz wszystkie typy

Komenda (ustalone słowo kluczowe)	**Argument**	**Wartość zwracana**
SYS_INITxxx	adres pamięci	Następny wolny adres
SYS_SAVxxx	0	zero jest OK
SYS_RESTxxx	0	zero jeśli OK

- Implementacja specjalna

I wreszcie korzystając z funkcji lub bloków funkcyjnych C użytkownik może opracować swoje specjalne procedury zapewniające dostęp do pamięci zasilanej akumulatorem w celu zapamiętania i odtworzenia zmiennych w dowolnym momencie pracy aplikacji.

Przykłady:

1) Procedura przeznaczona dla aplikacji:

backup, restore_temp, restore_date, restore_cpt mogą być procedurami użytkownika w języku C.

backup(temperature, date, cnt); zapamiętuje 3 krytyczne dane

temperature := **restore_temp()**; odtwarza temperaturę
date := **restore_date()**; odtwarza datę
cnt := **restore_cnt()**; odtwarza licznik

2) Procedura ogólnego przeznaczenia:

backup_init, backup, backup_link, restore mogą być procedurami użytkownika w języku C.

save_id := **backup_init**(address, size); przydziela pole w pamięci pomoc.
backup(save_id, cpt1, 3); zapamiętuje cpt1 jako 3-ci element.

rest_id := **backup_link**(address, size) dołącza pamięć pomocniczą.
cpt1 := **restore**(rest_id, 3); odtwarza z kopii wartość cpt1.

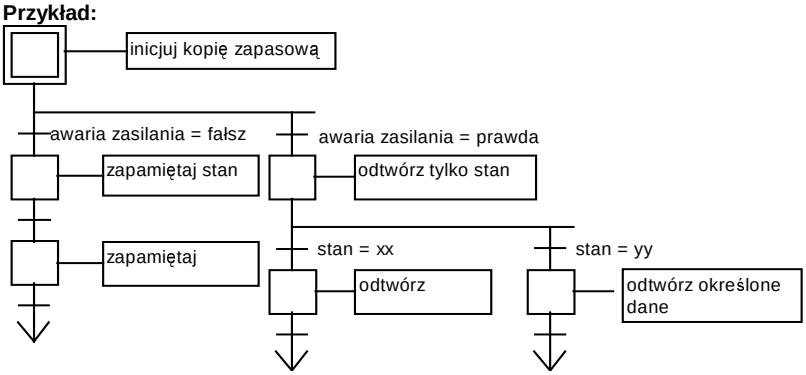
C.9.3 Kopia zapasowa danych o stanie programu

Możliwe jest odtworzenie dowolnego stanu każdego programu aplikacji, ale niebezpieczne wydaje się odtwarzanie programu do stanu, w jakim znajdował się on w chwili wykonywania ostatniej kopii zapasowej. Są tego trzy powody:

- Niektóre procesy wymagają specjalnych operacji przed ponownym uruchomieniem
- Zajmowanie się każdym stanem całej aplikacji jest żmudne
- Niektóre zasoby zewnętrzne, jak programy w języku C, urządzenia peryferyjne itp. nie mogą zostać ponownie uruchomione w sposób automatyczny.

Najlepszym rozwiązaniem wydaje się być wykonanie kopii zapasowej zmiennych analogowych lub dwustanowych, aby opisać stan procesu w momencie, gdy zdaniem programisty dane te będą mogły zostać wykorzystane przy ponownym uruchomieniu programu. Potem w oparciu niekompletny, ale inteligentny 'obraz' procesu powinno być możliwe uruchomienie lub zamknięcie programów SFC i zainicjowanie zmiennych w celu doprowadzenia aplikacji do odpowiedniego stanu. ISaGRAF nie zapewnia jednak żadnej procedury automatycznego uruchomienia.

Przykład:



C.10 Załącznik: Wykaz błędów i ich opisy

Lista błędów:

Kod	Komunikat	Typ
1	Nie można zaalokować pamięci dla bazy danych aplikacji	systemowy
2	Nieprawidłowa baza danych aplikacji lub zła CRC	aplikacyjny
3	Nie można zaalokować skrzynki komunikacyjnej	systemowy
4	Nie można połączyć się z bazą danych jądra	systemowy
5	Przekroczenie limitu czasu - wysyłanie zapytania do jądra	systemowy
6	Przekroczenie limitu czasu - oczekiwanie na odpowiedź z jądra	systemowy
7	Nie można zainicjować komunikacji	systemowy
8	Nie można zaalokować pamięci dla zmiennych zachowywanych	aplikacyjny
9	Aplikacja zatrzymana	aplikacyjny
10	Zbyt wiele równoczesnych akcji N lub P.	aplikacyjny
11	Zbyt wiele równoczesnych akcji ustawiania	aplikacyjny
12	Zbyt wiele równoczesnych akcji kasowania	aplikacyjny
13	Nieznana instrukcja TIC	aplikacyjny
16	Nie można odpowiedzieć na żądanie czytania danych	systemowy
17	Nie można odpowiedzieć na żądanie wpisania danych	systemowy
18	Nie można odpowiedzieć na żądanie sesji debagera	systemowy
19	Nie można odpowiedzieć na żądanie modbus	systemowy
20	Nie można odpowiedzieć na żądanie aplikacji debagera	systemowy
21	Nie można odpowiedzieć debagerowi	systemowy
22	Przekroczony stos	systemowy
23	Nieznany kod żądania	systemowy
24	Błąd komunikacji Ethernet	systemowy
25	Błąd synchronizacji komunikacji	systemowy
28	Nie można zaalokować pamięci dla aplikacji	systemowy
29	Nie można przydzielić pamięci dla aktualizacji aplikacji	systemowy
30	Nieznany kod OEM	aplikacyjny
31	Nie można zainicjować karty wejść binarnych	aplikacyjny
32	Nie można zainicjować karty wejść analogowych	aplikacyjny
33	Nie można zainicjować karty wejść komunikatowych	aplikacyjny
34	Nie można zainicjować karty wyjść binarnych	aplikacyjny
35	Nie można zainicjować karty wyjść analogowych	aplikacyjny
36	Nie można zainicjować karty wyjść komunikatowych	aplikacyjny
37	Nie można odczytać karty wejść binarnych	aplikacyjny
38	Nie można odczytać karty wejść analogowych	aplikacyjny
39	Nie można odczytać karty wejść komunikatowych	aplikacyjny
40	Nie można wystawić zmiennej wyjściowej binarnej	aplikacyjny
41	Nie można wystawić zmiennej wyjściowej analogowej	aplikacyjny
42	Nie można wystawić zmiennej wyjściowej komunikatowej	aplikacyjny
43	Nie można obsłużyć zmiennej wyjściowej binarnej	aplikacyjny

44	Nie można obsłużyć zmiennej wyjściowej analogowej	aplikacyjny
45	Nie można obsłużyć zmiennej wyjściowej komunikatowej	aplikacyjny
46	Nie można otworzyć karty	aplikacyjny
47	Nie można zamknąć karty	aplikacyjny
50	Nie można nadpisać zmiennej wyjściowej binarnej	programowy
51	Nie można nadpisać zmiennej wyjściowej analogowej	programowy
52	Nie można nadpisać zmiennej wyjściowej komunikatowej	programowy
61	Nieznany kod żądania SYSTEMU	programowy
62	Okres próbkowania przekroczony	programowy
63	Funkcja użytkownika nie zaimplementowana	aplikacyjny
64	Integer podzielony przez zero	programowy
65	Funkcja konwersji nie zaimplementowana	aplikacyjny
66	Blok funkcyjny nie zaimplementowany	aplikacyjny
67	Funkcja standardowa nie zaimplementowana	aplikacyjny
68	Dzielenie przez zero (real)	programowy
69	Niepoprawne parametry operacji	aplikacyjny
72	Symbole aplikacji nie mogą być modyfikowane	aplikacyjny
73	Nie można uaktualnić: inny zbiór zmiennych binarnych	aplikacyjny
74	Nie można uaktualnić: inny zbiór zmiennych analogowych	aplikacyjny
75	Nie można uaktualnić: inny zbiór zmiennych typu timer	aplikacyjny
76	Nie można uaktualnić: inny zbiór zmiennych komunikatowych	aplikacyjny
77	Nie można uaktualnić: nie można znaleźć nowej aplikacji	aplikacyjny
> 100	Szczególny kod błędu OEM. Należy poprosić dostawcę o szczegóły	

Opisane poniżej trzy typy błędów odpowiadają różnym źródłom problemów:

– Błędy systemowe:
Takie błędy są prawdopodobnie spowodowane oprogramowaniem wbudowanym lub sprzętem, a nie ustawieniami aplikacji czy wykonywaniem programu.
Należy spróbować przeprowadzić zerowanie sprzętowe (wyłączenie zasilania) z komputera z oprogramowaniem wbudowanym i uruchomić inną aplikację.
Błędy te powinny być zgłaszane serwisowi ISaGRAF.

– **Błędy aplikacyjne:**
Takie błędy są spowodowane parametrami, wielkością lub zawartością aplikacji.
Powinny zniknąć po załadowaniu znanej i poprzednio sprawdzonej aplikacji. Jeśli mimo to problem nadaj się utrzymuje, to jest to błąd opisany powyżej systemowy.

– **Błędy programowe:**
Takie błędy są spowodowane szczególną sekwencją programową.
Powinny zniknąć po uruchomieniu aplikacji w trybie 'cykl po cyklu' lub po zatrzymaniu krytycznego programu.

Opis błędów:

1. Nie można zaalokować pamięci dla bazy danych aplikacji	***systemowy***

Brak pamięci. Proszę sprawdzić sprzęt.

2. Nieprawidłowa baza danych aplikacji lub zła CRC	***aplikacyjny***

Plik aplikacji lub jego kopia zapasowa nie są poprawne. Błąd ten pojawia się wtedy, gdy aplikacja jest generowana dla procesora INTEL a załadowana do procesora MOTOROLA (i odwrotnie) lub jeśli plik został zmieniony.

3. Nie można zaalokować skrzynki komunikacyjnej	***systemowy***

Ten błąd występuje w zadaniu komunikacji, jeśli nie może ono przydzielić obszaru pamięci 3 do komunikacji między zadaniami.

4. Nie można połączyć się z bazą danych jądra	**systemowy**

Ten błąd występuje w zadaniu komunikacji, jeśli nie może ono znaleźć jądra o numerze sterownika określonym w wierszu komend.

5. Przekroczenie limitu czasu - wysyłanie zapytania do jądra	***systemowy***

Zadanie komunikacji nie może wysłać żądania do jądra. Jądro prawdopodobnie nie pracuje lub jest zajęte.

6. Przekroczenie limitu czasu - oczekiwanie na odpowiedź z jądra	***systemowy***

Zadanie komunikacji nie może otrzymać z jądra żadnej odpowiedzi. Jądro prawdopodobnie nie pracuje lub jest zajęte.

7. Nie można zainicjować komunikacji	***systemowy***

Ostrzeżenie to jest generowane wtedy, gdy warstwa komunikacyjna nie może uaktywnić fizycznego łącza. Ostrzeżenie jest też wyświetlane wtedy, gdy nie została określona ścieżka komunikacyjna. Nie stanowi to przeszkody dla poprawnej pracy oprogramowania wbudowanego, które nie może nawiązać łączności.

8. Nie można zaalokować pamięci dla zmiennych zachowywanych	***aplikacyjny***

ISaGRAF nie może obsługiwać zachowywanych zmiennych. Problem może wynikać z dwóch przyczyn:
- ciąg przekazany jako parametr do oprogramowania wbudowanego nie jest syntaktycznie (składniowo) poprawny
- wielkość pamięci określona dla każdego bloku nie jest wystarczająca.
Należy skorygować składnię parametru ‘zachowywanej zmiennej’ i spróbować ze zmniejszoną liczbą zachowywanych zmiennych.

9. Aplikacja zatrzymana	***aplikacyjny***

To ostrzeżenie pojawia się zawsze wtedy, gdy aplikacja zostaje zatrzymana przez debagera.

10. Zbyt wiele równoczesnych akcji N lub P	***aplikacyjny***

Ten błąd występuje wtedy, gdy jeden z cykli oprogramowania wbudowanego musi wykonać zbyt wiele działań bez zapamiętywania lub zbyt wiele bloków cyklicznych. Problem można zlokalizować w trybie CC. W programie SFC dopuszcza się maksymalnie 2 + 4 równoczesne operacje.

11. Zbyt wiele równoczesnych akcji ustawiania	***aplikacyjny***

Ten błąd występuje wtedy, gdy jeden z cykli oprogramowania wbudowanego musi wykonać zbyt wiele operacji ustawiania (wykonywanych po uaktywnieniu etapu). Postępować jak powyżej.

12. Zbyt wiele równoczesnych akcji kasowania	***aplikacyjny***

Ten błąd występuje wtedy, gdy jeden z cykli oprogramowania wbudowanego musi wykonać zbyt wiele zerowania (wykonywanych gdy etap przestaje być aktywny). Postępować jak powyżej.

13. Nieznana instrukcja TIC	***aplikacyjny***

Jądro wykryło pewną nieprawidłowość w kodzie aplikacyjnym (Target Independent Code) w programie. Są dwa ewentualne wyjaśnienia tego błędu:
- program zewnętrzny dokonuje wpisu do kodu aplikacyjnego. Należy spróbować zlokalizować zatrzymanie w trybie CC i upewnić się, czy żaden interfejs We/Wy nie zawiera niewłaściwych parametrów.
- oprogramowanie wbudowane zredukowało zbiór instrukcji , a w aplikacji jest wykorzystana niedozwolona instrukcja lub niedozwolony typ zmiennej.

16. Nie można odpowiedzieć na żądanie czytania danych	***systemowy***

W odpowiedzi na kod funkcji 18 (czytanie pliku) określonego żądania Modbus ISaGRAF został wykryty błąd komunikacji. Należy sprawdzić połączenie i konfigurację systemu zarówno po stronie oprogramowania wbudowanego jak i stacji master.

17. Nie można odpowiedzieć na żądanie wpisania danych	***systemowy***

W odpowiedzi na kod funkcji 17 (zapis pliku) określonego żądania Modbus ISaGRAF został wykryty błąd komunikacji. Należy sprawdzić połączenie i konfigurację systemu zarówno po stronie oprogramowania wbudowanego jak i stacji master.

18. Nie można odpowiedzieć na żądanie sesji debagera	***systemowy***

W odpowiedzi na żądanie debagera został wykryty błąd komunikacji. Należy sprawdzić połączenie i konfigurację systemu zarówno po stronie oprogramowania wbudowanego jak i stacji master.

19. Nie można odpowiedzieć na żądanie modbus	***systemowy***

W odpowiedzi na żądanie Modbus został wykryty błąd komunikacji. Należy sprawdzić połączenie i konfigurację systemu zarówno po stronie oprogramowania wbudowanego jak i stacji master.

20. Nie można odpowiedzieć na żądanie aplikacji debagera	***systemowy***

W odpowiedzi na żądanie debagera został wykryty błąd komunikacji. Należy sprawdzić połączenie i konfigurację systemu zarówno po stronie oprogramowania wbudowanego jak i stacji master.

21. Nie można odpowiedzieć debagerowi	***systemowy***

W odpowiedzi na żądanie debagera został wykryty błąd komunikacji. Należy sprawdzić połączenie i konfigurację systemu zarówno po stronie oprogramowania wbudowanego jak i stacji master.

23. Nieznany kod żądania	***systemowy***

Żądanie debagera nie ma sensu.

24. Błąd komunikacji Ethernet	***systemowy***

Ten błąd pojawia się przy każdorazowym zamknięciu połączenia po zamknięciu debagera: system pracuje poprawnie. W przeciwnym razie oznacza błąd komunikacji poprzez Ethernet. Należy sprawdzić połączenie i konfigurację systemu zarówno po stronie oprogramowania wbudowanego jak i stacji master.
Jest podawane drugie pole, którym może być:
1: błąd podczas wysyłania lub otrzymywania
2: błąd podczas tworzenia gniazda
3: błąd podczas łączenia lub przesłuchiwania gniazda
4: błąd podczas akceptowania nowego klienta

25. Błąd synchronizacji komunikacji	***systemowy***

Zła synchronizacja pomiędzy zadaniem komunikacji w oprogramowaniu wbudowanym i stacji master. Należy sprawdzić połączenie i konfigurację systemu (parametry komunikacji) zarówno po stronie oprogramowania wbudowanego jak i stacji master.

28. Nie można zaalokować pamięci dla aplikacji	***systemowy***

Brak pamięci. Należy sprawdzić możliwości sprzętowe pod kątem wielkości aplikacji.

29. Nie można zaalokować pamięci do aktualizacji aplikacji	***systemowy***

Brak pamięci. Należy sprawdzić możliwości sprzętowe pod kątem wielkości aplikacji.

30. Nieznany kod OEM	***aplikacyjny***

Aplikacja wykorzystuje kartę, której kod fabryczny nie jest rozpoznawany przez oprogramowanie wbudowane. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i zastosować atrybut 'WIRTUALNA' celu zlokalizowania niewłaściwej karty. Biblioteka pakietu ISaGRAF może nie odpowiadać wersji oprogramowania wbudowanego.

31. Nie można zainicjować karty wejść binarnych	***aplikacyjny***

Nie powiodła się inicjalizacja kart wejść binarnych. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart wejść binarnych.

32. Nie można zainicjować karty wejść analogowych	***aplikacyjny***

Nie powiodła się inicjalizacja kart wejść analogowych. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart wejść analogowych.

33. Nie można zainicjować karty wejść komunikatowych	***aplikacyjny***

Nie powiodła się inicjalizacja kart wejść dla komunikatów. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart wejść typu komunikat.

34. Nie można zainicjować karty wyjść binarnych	***aplikacyjny***

Nie powiodła się inicjalizacja kart wyjść binarnych. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart wyjść binarnych.

35. Nie można zainicjować karty wyjść analogowych	***aplikacyjny***

Nie powiodła się inicjalizacja kart wyjść analogowych. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart wyjść analogowych.

36. Nie można zainicjować karty wyjść komunikatowych	***aplikacyjny***

Nie powiodła się inicjalizacja kart wyjść typu komunikat. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart wyjść typu komunikat.

37. Nie można odczytać karty wejść binarnych	***aplikacyjny***

Podczas odświeżania karty wejść binarnych został wykryty błąd. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart.

38. Nie można odczytać karty wejść analogowych	***aplikacyjny***

Podczas odświeżania karty wejść analogowych został wykryty błąd. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF jak również parametry kart.

39. Nie można odczytać karty wejść komunikatowych	***aplikacyjny***

Podczas odświeżania karty wejść typu komunikat został wykryty błąd. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart.

40. Nie można wystawić zmiennej wyjściowej binarnej	***aplikacyjny***

Podczas uaktualniania wyjściowej zmiennej binarnej został wykryty błąd. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart.

41. Nie można wystawić zmiennej wyjściowej analogowej	***aplikacyjny***

Podczas uaktualniania wyjściowej zmiennej analogowej został wykryty błąd. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart.

42. Nie można wystawić zmiennej wyjściowej komunikatowej	***aplikacyjny***

Podczas uaktualniania wyjściowej zmiennej typu komunikat został wykryty błąd. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF i parametry stosowanych kart.

43. Nie można obsłużyć zmiennej wyjściowej binarnej	***aplikacyjny***

Podczas wykonywania wywołania OPERATE do zmiennej binarnej został wykryty błąd. Należy zweryfikować parametry OPERATE i opis użytkownika karty.

44. Nie można obsłużyć zmiennej wyjściowej analogowej	***aplikacyjny***

Podczas wykonywania wywołania OPERATE do zmiennej analogowej został wykryty błąd. Należy zweryfikować parametry OPERATE i opis użytkownika karty.

45. Nie można obsłużyć zmiennej wyjściowej komunikatowej	***aplikacyjny***

Podczas wykonywania wywołania OPERATE do zmiennej typu komunikat został wykryty błąd. Należy zweryfikować parametry OPERATE i opis użytkownika karty.

46. Nie można otworzyć karty	***aplikacyjny***

Aplikacja stosuje odsyłacz do karty, jaki nie jest rozpoznawany przez oprogramowanie wbudowane. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF. Biblioteka pakietu ISaGRAF może nie odpowiadać wersji oprogramowania wbudowanego.

47. Nie można zamknąć karty	***aplikacyjny***

Aplikacja stosuje odsyłacz do karty, jaki nie jest rozpoznawany przez oprogramowanie wbudowane. Należy sprawdzić połączenie We/Wy w pakiecie ISaGRAF.

50. Nie można nadpisać zmiennej wyjściowej binarnej	***programowy***

Ta sama wyjściowa zmienna binarna jest zapisywana przez dwie sekwencje SFC w tym samym cyklu oprogramowania wbudowanego. Należy unikać takich sytuacji, aby zapobiec niebezpiecznemu stanowi wejść/wyjść. W przypadku takiego konfliktu priorytet otrzymuje program znajdujący się najwyżej w hierarchii. Jeśli dwa programy SFC znajdują się na tym samym poziomie, rezultat jest nieprzewidywalny.

51. Nie można nadpisać zmiennej wyjściowej analogowej	***programowy***

Ta sama wyjściowa zmienna analogowa jest zapisywana przez dwa programy SFC w tym samym cyklu oprogramowania wbudowanego. Patrz komentarz powyżej.

52. Nie można nadpisać zmiennej wyjściowej komunikatowej	***programowy***

Ta sama wyjściowa zmienna typu komunikat jest zapisywana przez dwa programy SFC w tym samym cyklu oprogramowania wbudowanego. Patrz komentarz powyżej.

61. Nieznany kod żądania SYSTEMU	***programowy***

Program stosuje wywołanie SYSTEM z niewłaściwym kodem.

62. Okres próbkowania przekroczony	***programowy***

Cykl oprogramowania wbudowanego jest dłuższy niż określony w menu pakietu ISaGRAF.
W systemie wielozadaniowym oznacza to brak czasu jednostki centralnej na wykonanie cyklu nawet, jeśli ‘aktualny czas trwania cyklu’ jest krótszy od określonego.
W systemie jednozadaniowym oznacza to zawsze zbyt wiele operacji w jednym cyklu oprogramowania wbudowanego.

Jest wiele sposobów uniknięcia tego błędu:
– zmniejszenie liczby operacji wykonywanych w instancji, w której wykryto błąd,
– zmniejsz7enie liczby znaczników, dopuszczalnych przejść oraz optymalizacja złożonych operacji przetwarzania, itd.,
– zmniejszenie obciążenia jednostki centralnej innymi zadaniami, aby zapewnić więcej czasu dla ISaGRAF,
– zmniejszenie natężenie ruchu komunikacyjnego, aby zapewnić więcej czasu dla ISaGRAF,
– zastosowanie dynamicznej modyfikacji czasu trwania cyklu, aby dostosować czas trwania cyklu do różnych etapów procesu,
– ustawienie czasu trwania cyklu na zero, aby jądro ISaGRAF mogło pracować z maksymalną szybkością bez sprawdzania przepełnienia.

63. Funkcja użytkownika nie zaimplementowana	***aplikacyjny***

Program korzysta z funkcji C, która nie jest rozpoznawana przez oprogramowanie wbudowane. Biblioteka pakietu ISaGRAF może nie odpowiadać wersji oprogramowania wbudowanego.

64. Integer podzielony przez zero	***programowy***

Program próbuje podzielić wielkość analogową typu integer przez zero. Aplikacja powinna zapobiegać tego typu zdarzeniom, których skutki mogą być nieprzewidziane.
W takim przypadku, jako rezultat dzielenia ISaGRAF poda maksymalną wartość analogową. Jeśli operand jest ujemny, to wynik zostanie odwrócony.

65. Funkcja konwersji nie zaimplementowana	***aplikacyjny***

Program korzysta z funkcji konwersji C, która nie jest rozpoznawana przez oprogramowanie wbudowane. Biblioteka pakietu ISaGRAF może nie odpowiadać wersji oprogramowania wbudowanego.
W takim przypadku ISaGRAF nie dokona konwersji wartości.

66. Blok funkcyjny nie zaimplementowany	***aplikacyjny***

Program korzysta z bloku funkcyjnego C, który nie jest rozpoznawany przez oprogramowanie wbudowane. Biblioteka pakietu ISaGRAF może nie odpowiadać wersji oprogramowania wbudowanego.

67. Funkcja standardowa nie zaimplementowana	***aplikacyjny***

Program korzysta z *bloku funkcyjnego* C, który nie jest rozpoznawany przez oprogramowanie wbudowane, chociaż powinien być obsługiwany przez większość implementacji oprogramowania wbudowanego. Należy skontaktować się dostawcą.

68. Dzielenie przez zero (real)	***programowy***

Program próbuje podzielić wielkość analogową typu real przez zero. Aplikacja powinna zapobiegać tego typu zdarzeniom, których skutki mogą być nieprzewidziane.
W takim przypadku, jako rezultat dzielenia ISaGRAF poda maksymalną wartość analogową typu real.
Jeśli operand jest ujemny, to wynik zostanie odwrócony.

69. Nieprawidłowe parametry operacji	***aplikacyjny***

Aplikacja stosuje wywołanie OPERATE ze złymi parametrami. Normalnie jest to wyłapywane przez kompilator. Należy użyć parametru typu timer lub zmiennej, która nie jest zmienną wejściową ani wyjściową.

72. Symbole aplikacji nie mogą być modyfikowane	***aplikacyjny***

Po próbie uaktualnienia aplikacji ta zmodyfikowana aplikacja nie może zostać uruchomiona z powodu różnych symboli. Może dodano, usunięto lub zmodyfikowano jedną lub więcej zmiennych lub instancji bloków funkcyjnych w porównaniu z bieżącą aplikacją.

73. Nie można uaktualnić: inny zbiór zmiennych binarnych	***aplikacyjny***

Zmodyfikowana aplikacja nie może zostać uruchomiona, ponieważ w aktualnej aplikacji dodano lub usunięto pewne zmienne binarne.

74. Nie można uaktualnić: inny zbiór zmiennych analogowych	***aplikacyjny***

Zmodyfikowana aplikacja nie może zostać uruchomiona, ponieważ w aktualnej aplikacji dodano lub usunięto pewne zmienne analogowe.

75. Nie można uaktualnić: inny zbiór zmiennych typu timer	***aplikacyjny***

Zmodyfikowana aplikacja nie może zostać uruchomiona, ponieważ w aktualnej aplikacji dodano lub usunięto pewne zmienne typu timer.

76. Nie można uaktualnić: inny zbiór zmiennych komunikatowych	***aplikacyjny***

Zmodyfikowana aplikacja nie może zostać uruchomiona, ponieważ w aktualnej aplikacji dodano lub usunięto pewne zmienne typu komunikat.

77. Nie można uaktualnić: nie można znaleźć nowej aplikacji	***aplikacyjny***

Nie można znaleźć w pamięci zmodyfikowanej aplikacji. Coś nieprawidłowego musiało się zdarzyć podczas ładowania aplikacji.

D. Glosariusz

Adres sieciowy Opcjonalny adres szesnastkowy swobodnie definiowany dla każdej zmiennej. Adres ten jest używany przez protokół Modbus gdy oprogramowanie wbudowane jest połączone z systemem zewnętrznym.

Aktywność kroku Atrybut kroku oznaczany znacznikiem SFC. Działania powiązane z krokiem są wykonywane wtedy, gdy krok jest aktywny.

Analogowe Typ zmiennych. Są to zmienne typu integer lub real.

Argument (IL) Wyrażenie zmienne lub stałe przetwarzane przez elementarną instrukcję IL.

Atrybut Klasa zmiennych. Stosowanymi atrybutami zmiennych są zmienne wewnętrzne, wejściowe i wyjściowe.

Biblioteka Zbiór zasobów sprzętowych i programowych, które mogą zostać wstawione w dowolnej aplikacji.

Binarne Typ zmiennych. Takie zmienne mogą jedynie przyjmować wartość prawdy lub fałszu.

Blok funkcyjny Element graficzny języka FBD reprezentujący elementarną funkcję standardową z bibliotek ISaGRAF.

Błąd wykonania Błąd aplikacji wykryty przez oprogramowanie wbudowane ISaGRAF w czasie jego wykonywania.

Cewka Element graficzny programu LD reprezentujący przyporządkowanie zmiennej wyjściowej.

Cykliczny Atrybut programu zawsze wykonywanego.

Cykl sterownika Zbiór operacji wykonywanych po każdorazowym uaktywnieniu oprogramowania wbudowanego IsaGRAF. Cykle są przełączane w oparciu o programowalny czas cyklu.

Czas cyklu Czas trwania cyklu wykonywania aplikacji.

Czyszczenie przejścia Operacja polegająca na usunięciu wszystkich znaczników znajdujących się poprzednich krokach. Znacznik zostanie utworzony w każdym następnym kroku.

Decyzja (FC) (Również nazywana testem) Symbol grafu przepływowego powiązany z wyrażeniem binarnym. Przepływ jest kierowany do wyjścia TAK albo NIE symbolu, w zależności od stanu wyrażenia.

Dziennik Plik tekstowy zawierający wszystkie opisy zmian wprowadzonych w jednym programie. Każdy opis jest uzupełniony datą edycji.

Etykieta (IL) Identyfikator umieszczany na początku linii instrukcji IL. Etykieta służy do identyfikowania instrukcji i może być użyta jako argument w operacjach skoku (JMP).

FBD Skrót od Functional Block Diagram.

FC Skrót od Flow Chart.

Flow Chart Znaczy: Graf Przepływowy. Język graficzny służący do projektowania przepływu. Działanie grafu polega na wykonywaniu operacji i podejmowaniu decyzji pozwalających na wybór różnych dróg przepływu. Język grafu przepływowego umożliwia wstawianie pętli realizowanych w następujących po sobie cyklach.

Functional Block Diagram Znaczy: Diagram Bloków Funkcyjnych. Język graficzny, w którym równania są budowane z elementarnych bloków standardowych z biblioteki ISaGRAF. Bloki połączone ze sobą tworzą diagram.

Funkcja C Funkcja napisana w języku "C", wywoływana w sposób synchroniczny z programów IsaGRAF (napisanych w innych językach). Funkcje C są dostarczane przez CJ International lub opracowywane przez użytkownika.

Funkcja konwersji Funkcja napisana w języku "C" opisująca konwersję. Taka konwersja może być powiązana z dowolną analogową zmienną wejściową lub wyjściową.

Globalna Zakres zmiennych lub zdefiniowanych słów. Takie obiekty mogą być użyte w dowolnym programie jednego projektu.

Hierarchia Architektura projektu podzielonego na kilka programów. Drzewo hierarchiczne reprezentuje połączenia pomiędzy programami rodzicami a programami potomnymi.

Identyfikator Charakterystyczne słowo służące do reprezentowania zmiennej lub wyrażenia stałego w kodzie programowym.

IL Skrót od Instruction List.

Instrukcja ST Podstawowa kompletna operacja ST.

Instrukcja Elementarna operacja programu IL wprowadzana w jednej linii tekstu.

Instruction List Znaczy: Lista Instrukcji. Prosty język niższego rzędu wprowadzany jako sekwencyjna lista elementarnych operacji.

Integer Klasa zmiennych analogowych zapamiętywanych w formacie 32-bitowej liczby całkowitej ze znakiem.

Język C Język wyższego rzędu służący do opisywania operacji komputerowych takich,

jak funkcje i funkcje konwersji C.

Kanał We/Wy Pojedynczy punkt przyłączeniowy na karcie We/Wy. Kanał We/Wy może otrzymać jedną zmienną We/Wy.

Karta rzeczywista Fizyczna karta We/Wy połączona w sterowniku z urządzeniem We/Wy.

Karta We/Wy Część zasobu sprzętowego. Kartę We/Wy charakteryzuje typ i kierunek sygnału (wejściowa lub wyjściowa). Parametry karty We/Wy są opisane w bibliotece ISaGRAF.

Karta wirtualna Karta We/Wy, która nie jest w sterowniku fizycznie połączona z urządzeniem We/Wy.

Kod OEM (karta We/Wy) Szesnastkowy kod 16 bitowy definiowany dla każdej karty We/Wy biblioteki ISaGRAF. Kod OEM identyfikuje dostawcę karty.

Kod źródłowy C Plik tekstowy zawierający kod źródłowy funkcji lub funkcji konwersji w języku "C".

Komentarz Tekst zawarty w programie, nie mający żadnego wpływu na wykonywanie programu.

Komentarz (SFC) Tekst powiązany z przejściem lub krokiem SFC, nie mający żadnego wpływu na wykonanie programu.

Komórka Elementarne pole macierzy w językach graficznych takich, jak SFC, FBD lub LD.

Komunikat Typ zmiennej. Zmienne tego typu zawierają ciągi znaków o zmiennej długości.

Konwersja Filtr powiązany z analogową zmienną wejściową lub wyjściową. Konwersja jest automatycznie stosowana przy każdym wprowadzeniu lub wyprowadzeniu zmiennej.

Krok Podstawowy element graficzny języka SFC. Krok reprezentuje ustaloną sytuację procesu i jest oznaczany kwadratem. Odsyłaczem do kroku jest liczba. Funkcją kroku jest sterowanie wykonywaniem odpowiadających mu operacji.

Koniec kroku makro Ostatni krok w treści kroku makro SFC. Krok kończący nie jest powiązany z żadnym z następującym po nim przejściem.

Krok makro Element graficzny SFC. Krok makro jest charakterystyczną grupą kroków i przejść opisanych osobno i reprezentowanych w grafie głównym charakterystycznym symbolem.

Początek kroku makro Pierwszy krok w treści kroku makro. Krok początkowy nie jest powiązany z żadnym poprzedzającym go przejściem.

Krok wstępny	Specjalny krok programu SFC uaktywniany w chwili uruchamiania programu.
Ladder Diagram	Znaczy: Diagram Drabinkowy.(lub język stykowy) Język graficzny stosujący styki i cewki, służący do projektowania równań binarnych.
LD	Skrót od Ladder Diagram.
Lokalna	Zakres zmiennych lub zdefiniowanych słów. Obiekty tego typu mogą być używane w tylko jednym programie jednego projektu.
Łańcuch	Zbiór znaków zapamiętanych w zmiennej typu komunikat.
Macierz	Podczas edytowania programu napisanego w języku graficznym jest to logiczny podział edytowanego pola na prostokątne komórki.
Modbus	Protokół Master-Slave. Oprogramowanie wbudowane ISaGRAF jako 'slave' korzysta z Modbus do połączenia się z systemem zewnętrznym (takim, jak system wizualizacji) tworząc kompletną architekturę.
Modyfikator (IL)	Pojedynczy znak umieszczany na końcu słowa kluczowego operacji IL w celu zmodyfikowania znaczenia tej operacji.
Numer odniesienia (SFC)	Liczba dziesiętna (z przedziału od 1 do 65535) identyfikująca krok lub przejście SFC w programie SFC.
Odsyłacze	Informacje wyliczane przez pakiet ISaGRAF dla słownika zmiennych i określające miejsce użycia zmiennych w projekcie.
Operacja	Lista instrukcji lub przyporządkowań wykonywanych po uaktywnieniu kroku programu SFC. (akcja)
Operacja (IL)	Podstawowa instrukcja języka IL. Operacja jest generalnie skojarzona z argumentem instrukcji.
Operacja binarna	Operacja SFC, w której zmienna binarna jest przyporządkowana do sygnału aktywności kroku.
Operacja (FC)	Symbol na diagramie grafu przepływowego. Operacja jest listą instrukcji, które zostaną wykonane, gdy w procesie dynamicznego przepływu napotkany zostanie symbol operacji.
Operacja impulsowa	Operacja SFC: jest to lista instrukcji wykonywanych tylko jeden raz po uaktywnieniu skojarzonego z nią kroku.
Operacja nie zapisywana	Operacja SFC: jest to lista instrukcji wykonywanych w każdym cyklu oprogramowania wbudowanego po uaktywnieniu odpowiadającego jej kroku.
Operacja z opóźnienie	Operacja programu IL, która jest wykonywana dopiero po pojawieniu się w dalszej części programu instrukcji "(".

m (IL)

Opis kodu źródłowego C Plik tekstowy zawierający definicje i typy w języku "C" niezbędne do zaprogramowania funkcji konwersji lub funkcji C.
(Plik nagłówkowy C).

Opis techniczny Poradnik użytkownika dotyczący elementu bibliotek ISaGRAF (funkcji C lub bloku funkcyjnego, funkcji konwersji lub karty We/Wy). Opis jest tworzony przez projektanta danego elementu.

Parametr OEM (karta We/Wy) Parametr karty We/Wy zdefiniowany przez projektanta karty. Może być parametrem stałym lub zmiennym wprowadzanym przez użytkownika podczas połączenia We/Wy.

Parametr (funkcja C) Wartość nadana funkcji "C" przy jej wprowadzaniu. Parametr charakteryzuje się typem.

Parametr (karta We/Wy) Zdefiniowany przez użytkownika lub stały parametr standardowej karty We/Wy. Parametr zdefiniowany przez użytkownika jest wprowadzany przez programistę podczas połączenia We/Wy.

Podprogram Program napisany w dowolnym języku oprócz SFC i wywoływany przez inny program nazywany programem rodzicem.

Połączenie (FC) Element graficzny FC reprezentujący połączenie pomiędzy punktem grafu przepływowego a operacją lub testem FC. Symbolem graficznym skoku jest małe kółko, oznaczone numerem odniesienia elementu docelowego.

Połączenie We/Wy Definicja powiązania pomiędzy zmiennymi aplikacji a kanałami na kartach używanych w sterowniku.

Poziom 1 SFC Podstawowy opis programu SFC. Na poziomie 1 znajduje się graf (kroki i przejścia) i dotyczące go komentarze.

Poziom 2 SFC Szczegółowy opis programu SFC. Jest to opis operacji wykonywanych w ramach kroków i warunków binarnych powiązanych z przejściami.

Praca w czasie rzeczywistym Normalny tryb pracy: cykle oprogramowania wbudowanego są przełączane po upływie zaprogramowanego czasu cyklu.

Program Podstawowa jednostka programowa w projekcie. Program jest napisany w pewnym języku i jest umieszczany w architekturze hierarchicznej projektu.

Program najwyższego poziomu Program umieszczony na szczycie drzewa. Program najwyższego poziomu jest uaktywniany przez system.

Program potomny SFC Program SFC sterowany przez inny program SFC zwany rodzicem.

Program rodzic	Program napisany w dowolnym języku, sterujący innym programem nie napisanym w SFC i nazywanym jego podprogramem.
Program-rodzic SFC	Program SFC sterujący innymi programami SFC zwanymi programami potomnymi.
Projekt	Struktura kodowa grupująca wszystkie informacje (programy, zmienne, kod oprogramowania wbudowanego, itd.) dotyczące jednej aplikacji ISaGRAF.
Przejście	Podstawowy element graficzny SFC. Przejście reprezentuje warunek pomiędzy różnymi krokami SFC. Do przejścia odwołuje się za pomocą liczby. Z każdym przejściem jest związany pewien warunek binarny.
Pułapka	Oznaczenie umieszczane przez użytkownika w czasie debagowania na kroku lub przejściu SFC. Gdy znacznik SFC znajdzie się w pułapce, to system oprogramowania wbudowanego zatrzymuje się.
Real	Klasa zmiennych analogowych zapisywanych jako 32 bitowe słowa zmiennoprzecinkowe pojedynczej precyzji (wg IEEE).
Rejestr (IL)	Bieżący rezultat sekwencji IL.
Sekcja	Grupa programów wykonywanych przy zastosowaniu tych samych zasad dynamicznych.
Sekcja końcowa	Grupa cyklicznych programów wykonywanych pod koniec każdego cyklu oprogramowania wbudowanego.
Sekcja początkowa	Grupa cyklicznych programów wykonywanych na początku każdego cyklu oprogramowania wbudowanego.
Sekcja sekwencyjna	Grupa programów projektu. Wykonywanie tych programów opiera się na zasadach dynamicznych języka SFC.
Separator	Znak specjalny (lub grupa znaków) służących do oddzielania identyfikatorów w prostym języku programowania. Separator może reprezentować operację.
Sequential Function Chart	Znaczy: Graf Funkcji Sekwencyjnych. W języku graficznym proces jest opisany jako zbiór kroków połączonych przejściami. Operacje są powiązane z krokami. Przejścia są opisane warunkami binarnymi.
SFC	Skrót od Sequential Function Chart.
Skok do kroku	Element graficzny SFC reprezentujący połączenie pomiędzy przejściem a krokiem. Symbolem graficznym skoku jest strzałka oznaczona numerem odniesienia kroku docelowego.
Słownik	Zbiór zmiennych wewnętrznych, wejściowych lub wyjściowych oraz zdefiniowanych słów używanych w programach jednego projektu.
Słowo	Słowo zastrzeżone dla danego języka.

kluczowe	
Sprawdzenie poprawności przejścia	Atrybut przejścia. Przejście jest możliwe wtedy, gdy są uaktywnione wszystkie poprzedzające je kroki.
ST	Skrót od Structured Text.
Sterownik	Urządzenie, na którym pracuje oprogramowanie wbudowane ISaGRAF, i które obsługuje jądro ISaGRAF.
Structured Text	Znaczy: Tekst Strukturalny. Jest to prosty strukturalny język wyższego rzędu, łączący w sobie przyporządkowania, struktury wyższego rzędu takie, jak If/Then/Else i wywołania funkcji.
Styk	Element graficzny programu LD. Reprezentuje status zmiennej wejściowej.
Sytuacja pierwotna	Zbiór kroków wstępnych programu SFC, reprezentujący środowisko pracy programu w chwili jego uruchamiania.
Szyna zasilająca	Główna pionowa szyna znajdująca się na skraju diagramu drabinkowego po jego lewej i prawej stronie.
Tabela konwersji	Zbiór punktów definiujących konwersję liniową (segmentami). Taka konwersja może być zastosowana do dowolnej analogowej zmiennej wejściowej lub wyjściowej.
Test (FC)	(Również nazywany decyzją) Symbol grafu przepływowego powiązany z wyrażeniem binarnym. Przepływ jest kierowany do wyjścia TAK albo NIE symbolu, w zależności od stanu wyrażenia.
Timer	Typ zmiennej. Zmienne tego typu zawierają wartości czasowe i mogą być uaktualniane automatycznie przez system IsaGRAF w czasie jego pracy.
Toolbox	Małe okienko w oknie graficznych narzędzi edycyjnych, w którym są zgrupowane główne przyciski wybierania elementów graficznych.
Tryb cykl po cyklu	Tryb pracy: w tym trybie cykle są wykonywane pojedynczo, zgodnie z poleceniami wydawanymi przez użytkownika debagera.
Typ	Klasa zmiennych tego samego formatu. Podstawowymi typami zmiennych są zmienne binarne, analogowe, typu timer i komunikat.
Wartość zwracana podprogramu	Wartość zwracana przez podprogram pod koniec jego wykonywania. Wartość zwracana jest używana w operacjach programu rodzica.
Warunek (dla przejścia)	Wyrażenie binarne skojarzone z przejściem SFC. Gdy warunek ma wartość fałsz, to przejście nie może zostać wyczyszczone.
Wejściowa	Atrybut zmiennej. Zmienne tego typu są powiązane z urządzeniem wejściowym.

Wewnętrzna Atrybut zmiennej nie powiązanej z urządzeniem wejścia ani wyjścia.

Wspólne Zakres użycia zdefiniowanych słów. Takie obiekty mogą być użyte w dowolnym programie projektu.

Wyjściowa Atrybut zmiennej. Zmienne takie są powiązane z urządzeniem wyjściowym w sterowniku.

Wynik bieżący (IL) Rezultat wykonania instrukcji w programie IL. Wynik bieżący może być modyfikowany przez pewną instrukcję lub zostać użyty do ustawienia zmiennej.

Wyrażenie Zbiór operatorów i identyfikatorów umożliwiających dokonanie oceny wartości.

Wyrażenie stałe Proste wyrażenie służące do opisania wartości stałej. Wyrażenie stałe dotyczy tylko jednego typu.

Zablokowane We/Wy Zmienna wejściowa lub wyjściowa odłączona logicznie od skojarzonego z nią urządzenia We/Wy komendą "Zablokuj" wysłaną przez użytkownika z debagera.

Zakres (Miejsce) Zbiór programów, jakie mogą korzystać z danego obiektu. W IsaGRAF programy mogą mieć zakres globalny i lokalny.

Zbocze Zmiana zmiennej binarnej. Zbocze narastające oznacza zmianę z fałszu na prawdę. Zbocze opadające oznacza zmianę z prawdy na fałsz.

Zdefiniowane słowo Charakterystyczny identyfikator służący do zastąpienia w programie dowolnego wyrażenia.

Zmienna Charakterystyczna reprezentacja elementarnych danych przetwarzanych w programach projektu.

Zmienna We/Wy Zmienna skojarzona z urządzeniem wejścia lub wyjścia. Zmienna We/Wy musi być połączona z kanałem na karcie We/Wy.

Znacznik (SFC) Znacznik graficzny służący do wskazania aktywnych kroków programu SFC.

E. Indeks

-, B-77
%, A-96, B-11
&, B-74
*, B-78
/, B-79
:=, A-146
:= (ST, przyporządkowanie), B-58
+, B-77
<, B-82
<=, B-83
<>, B-86
=, B-86
=1, B-76
>, B-84
>=, B-85
>=1, B-75
1 do 1, B-73
1 gain, B-73

A

ABS, B-114
ACOS, B-118
Adres sieciowy, A-82, A-84, A-87, D-1
Akcja, A-46, A-51, B-33, B-35
Aktualizacja, A-117
Aktywność kroku, B-15, B-16, B-30, B-63, D-1
ANA, B-88
Analogowe, B-8, B-9, B-12
AND_MASK, B-80
appli.tst, C-6, C-16, C-29, C-38
appli.x8m, C-6, C-38
Archiwum, A-22, A-155, A-162, A-171
Archiwum - napęd, A-163
Archiwum - plik, A-164
ARCREATE, B-142
Arcus cosinus, B-118
Arcus sinus, B-119
Arcus tangens, B-119
Argument, A-159
Argument (IL), B-70, D-1
ARREAD, B-143
ARWRITE, B-144
ASCII, B-133
ASIN, B-119
ATAN, B-119
Atrybut, D-1

B

Bezpośredni styk, B-44
Bezpośrednia cewka, B-46
Biblioteka, A-29
Biblioteka, A-22, A-28, A-94, A-95, A-113, A-141, A-152, A-162, C-47, D-1
binarna akcja, A-43
Binarne, B-12, D-1
Bitmapa, A-129
Błąd, A-106
Błąd trybu pracy, A-30
Błąd wykonania, D-1
Błąd wykonywania, A-119
błędy wykonywania, B-92
blok funkcyjny, B-55
Blok funkcyjny, A-24, A-27, A-56, A-65, A-70, A-81, A-85, A-154, B-5, B-38, C-60
Blok funkcyjny, D-1
Blok funkcyjny C, C-47
Blokowanie, A-118, A-173
BOO, B-87
BY, B-61

C

C Blok funkcyjny, A-160
CAŁKA, B-109
CASE, B-60
CAT, B-91

Cewka, A-55, A-65, D-1
Cewka bezpośrednia, B-46
Cewka kasująca, B-48
Cewka ustawiająca, B-47
Cewka zanegowana, B-47
Cewka, zbocze narastające, B-48
Cewka, zbocze opadające, B-49
CHAR, B-134
CLKRATE, C-20
CMP, B-104
COS, B-120
Cosinus, B-120
CTD, B-99
CTU, B-98
CTUD, B-100
Cykl, B-2, B-6
Cykl po cyklu, A-30, A-118
Cykl sterownika, D-1
Cykle, A-148, C-2
Cykliczny, B-2
Cykliczny, D-1
Czas cyklu, A-30, A-118, A-143, B-92, C-7, C-40, C-49, C-53, C-60, D-1
Czas rzeczywisty, A-30, A-117
Czas trwania aktywności, B-16
Czcionka, A-168
Czyszczenie przejścia, B-30, D-1

D

DAY_TIME, B-142
DDE, A-124
DDE (sterownik NT), C-38, C-42, C-45
Debager, A-32, A-115, A-139
Decyzja, A-46, A-50, A-51, B-33
Decyzja, D-1
Deklaracja, A-26, A-80
DELETE, B-134
Diagnostyka, A-139
Diagram Bloków Funkcyjnych, B-38
Diagram Drabinkowy, B-42
Długość komunikatu, B-137
Długość łańcucha, B-137
DO, B-60, B-61
Dodawanie, B-77
Dodawanie komunikatów, B-91
Dokument, A-165
Dokumentacja, A-21, A-31
Dokumentacja projektu, A-165
Drukowanie, A-21, A-31, A-83, A-147, A-165, A-167
Dzielenie, B-79
Dziennik, A-26
Dziennik, D-2
Dziesiętne, B-9

E

Edycja opisu projektu, A-20
edytor SFC, A-35
edytor Tekstowy, A-73
Eksportowanie bloku funkcyjnego, A-29
Eksportowanie funkcji, A-29
Ekstrakcja podłańcucha (środek), B-139
Ekstrakcja podłańcucha (z lewej), B-138
Ekstrakcja podłańcucha (z prawej), B-141
ELSE, B-59, B-60
ELSIF, B-59
EN, A-56
End, A-150
END_CASE, B-60
END_FOR, B-61
END_IF, B-59
END_REPEAT, B-61
END_WHILE, B-60
ENO, A-57
EPROM, C-16, C-28
Ethernet, A-33
Etykieta, A-66, B-39, B-50
Etykieta (IL), B-69, D-2
Etykieta szczebla, A-58
Etykiety, A-149
EXIT, B-62
EXPT, B-115

F

F_CLOSE, B-147
F_EOF, B-148

F_ROPEN, B-145
F_TRIG, B-97
F_WOPEN, B-146
FA_READ, B-149
FA_WRITE, B-151
FALSE, A-84
FBD, A-64, B-38, C-54, C-62, D-2
FBD edytor, A-64, A-75
FBD komentarz, A-67
FC, A-46, B-32, D-2
FC edytor, A-46
FC komentarz, A-48
FC podprogram, A-24, B-34
FC połączenie, A-49
FEDGE, B-57
FIND, B-136
Flow, B-35
Flow Chart, A-46, B-32, D-2
Flow Chart edytor, A-46
FM_READ, B-153
FM_WRITE, B-155
FOR, B-61
Funkcja, A-24, A-27, A-154, A-158, B-4
Funkcja C, A-160, C-47, C-53, D-2
Funkcja konwersji, A-160, C-47, C-48, D-2

G

Galeria, A-45
Generacja kodu, A-102
Generator sygnału, B-112
GFREEZE, B-66
GKILL, B-66
Globalna, B-10, D-2
Gniazdo, A-94, A-97
Górny poziom, A-23
Goto, A-149
Graf Funkcji Sekwencyjnych, B-15
Grafika, A-129, A-134
GRST, B-67
Grupowanie, A-132
Grupy projektów, A-21
GSTART, B-65
GSTATUS, B-67

H

Hasło, A-21, A-97, A-153, A-169
Hierarchia, A-23, A-27, B-2, B-30
Hierarchia, D-2
Histereza, B-107, B-108
Historia, A-20, A-31
HYSTER, B-107

I

I, B-74
Identyfikator, D-2
If, A-150
IF, B-36, B-59
Ikona, A-130
Ikony, A-13
IL, A-73, A-128, B-27, B-29, D-2
IL edytor, A-75
IL21, B-70
IL22, B-70
IL23, B-70
IL24, B-70
IL25, B-71
IL26, B-71
IL27, B-71
IL28, B-71
IL29, B-70
Importowanie bloku funkcyjnego, A-28
Importowanie funkcji, A-28
Impulsowa operacja, A-42
INSERT, B-135
Instancja, A-81, A-85
Instancja bloku funkcyjnego, C-60
Instrukcja, D-2
Instrukcja, B-53, D-2
Instrukcje, B-69
Integer, A-85, B-8, D-2
Interfejs, A-27, A-159
ISA.EXE, C-4
ISA.O (VxWorks), C-19
isa_main, C-21, C-24
isa_register_slave, C-20
ISAGRAF.INI (sterownik NT), C-32
ISAKERET.O (VxWorks), C-19

ISAKERSE.O (VxWorks), C-19
ISAMOD (VxWorks), C-19
ISAMOD.EXE, C-4
ISASSR.O (VxWorks), C-19
ISAx0, C-15
ISAx1, C-15
ISAx2, C-15
ISAx3, C-15
ISAx4, C-15
ISAx5, C-15
ISAx6, C-6, C-15, C-28, C-37

J

Jednostka czasu, B-9
Język C, C-47, C-50, C-51, C-55, C-63, C-74, D-2
Język stykowy, B-42
Języki, A-24, B-6

K

Kanał, A-95, A-96, A-97, A-156, A-171
Kanał We/Wy, D-3
Kanały We/WY, OBSŁUGA, B-93
Karta, A-93, A-94
Karta rzeczywista, A-94
Karta rzeczywista, D-3
Karta We/Wy, A-156
Karta We/Wy, D-3
Karta wirtualna, A-94, A-173, D-3
Karta wirtualna (symulacja na NT), C-42, C-45
Karta wirtualna (symulacja na NT), C-35
Katalog, A-177
Klawisze wyjścia (na sterowniku), C-7, C-40
Kod C, A-106, A-154
Kod OEM, A-156
Kod OEM, D-3
Kod źródłowy, A-154
kod źródłowy C, C-51, C-56, C-74, D-3
kodu tworzenie, A-29
Kolejność wykonywania, A-69
Komentarz, B-13, B-35
Komentarz, D-3
Komentarz (SFC), B-15, B-16, D-3
Komentarz do szczebla, A-58
Komentarz kanału, A-95
Komentarz programu, A-26
Komentarz szczebla, A-62
Komórka, D-3
Kompilacja, A-29, A-102, A-154, A-158
Kompilator C, C-47, C-74
Kompilatora opcje, A-30
Kompresja, A-163
Komunikacja, A-33, A-119, A-136, A-176, C-4, C-9, C-10, C-11, C-14, C-19, C-32, C-42
Komunikat, A-85, A-127, B-9, B-13
Komunikat, D-3
Komunikat błędu, A-79
Komunikaty kompilatora, A-106
końcowy krok, A-39
Konfiguracja We/Wy, A-19
Konfigurator We/Wy, A-93, A-155
Koniec, A-23, B-32
Koniec kroku makro, D-3
Kontrola końca cyklu (VxWorks), C-21, C-24
Konwersja, A-99
Konwersja, D-3
Konwersja ASCII -> znak, B-134
Konwersja do binarnej, B-87
Konwersja do integer, B-88
Konwersja do komunikatu, B-90
Konwersja do real, B-89
Konwersja do timer, B-90
Konwersja znaku -> ASCII, B-133
Kopia zapasowa, A-22, A-155, A-162, A-163, A-171
Kopia zapasowa aplikacji (VxWorks), C-20, C-23
Kopiowanie biblioteki, A-153
Kopiowanie FBD, A-69
Kopiowanie FC, A-50
Kopiowanie LD, A-61
Kopiowanie programu, A-28
Kopiowanie SFC, A-40
Kopiowanie tekstu, A-73

Kopiowanie zmiennej, A-84
Krok, A-35, A-41, A-120, B-15
Krok, D-3
Krok kończący, B-21
Krok makro, A-37, A-39, B-21, D-3
Krok początkowy, A-37, A-39, B-30
Krok początkowy, B-16
Krok rozpoczynający, B-21
Krok wstępny, D-4
Krzywa, A-129, A-130, A-134

L

Łącze (LD), B-42
Łącze (SFC), B-17
Łączenie komunikatów, B-91
łącznik, A-49
Ladder Diagram, D-4
Łańcuch, B-9, D-4
LD, A-44, A-52, A-55, A-64, B-42, D-4
LD edytor, A-55, A-75
LEFT, B-138
Liczba losowa, B-131
LIM_ALRM, B-108
LIMIT, B-127
Limit czasu, A-33
Limit wielkości aplikacji, C-8
Lista instrukcji, D-2
Lista Instrukcji, B-69
Lista projektów, A-19, A-21
Lista zmiennych, A-126, A-128, A-132
LOG, B-115
Logarytm, B-115
Lokalna, B-10
Lokalna, D-4
Lokalne, A-159
LUB, B-75
LUB wyłączające, B-76

M

Macierz, D-4
Maksimum, B-126
Maska bitowa integer (I), B-80
Maska bitowa integer (lub), B-80
Maska bitowa integer (nie), B-82
Maska bitowa integer integer (lub wyłaczające), B-81
MAX, B-126
Menadżer biblioteki, A-152, C-47, C-49, C-54, C-61
Metafile, A-129
MID, B-139
MIGANIE, B-111
MIN, B-126
Minimum, B-126
MLEN, B-137
Mniejsze lub równe, B-83
Mniejsze niż, B-82
Mnożenie, B-78
MOD, B-128
Modbus, D-4
MODBUS, A-87, C-80
Modulo, B-128
Modyfikacja natychmiastowa, A-117, A-121
Modyfikacja śledzeń, A-71
Modyfikacja zmiennych, A-83
Modyfikator (IL), B-69, B-70, D-4
MSG, B-90
Multiplekser 4-wejściowy, B-129
Multiplekser 8-wejściowy, B-130
MUX4, B-129
MUX8, B-130

N

N kwalifikator, A-43
Narastające zbocze cewki, B-48
Narastające zbocze, styk, B-45
Narzędzia menu, A-31
Nawias, B-54, B-70
Nazwa zmiennej, B-10
Negacja, B-73
Negacja (FBD), B-40
New Zmienna, A-83
Nie przechowana, A-43
Nie równe, B-86
NOT, A-66, A-68
NOT_MASK, B-82

Nowa funkcja, A-25
Nowy blok funkcyjny, A-25
Nowy element biblioteki, A-152
Nowy program, A-25
Nowy projekt, A-19
Nowy szczebel, A-58
NT (klucz zabezpieczający), A-15
Numer logiczny zadania komunikacji, C-12
Numer logiczny zadania komunikacji, C-11, C-12, C-23
Numer odniesienia, B-15, B-16, B-17, B-21
Numer odniesienia, D-4
Numer sterownika, A-33, C-5, C-9, C-10, C-11, C-12, C-20, C-23, C-32, C-42, C-44
Numer sterownika, C-10
Numer wersji, A-117

O

Obcinanie części dziesiętnej, B-117
Obrót w lewo, B-122
Obrót w prawo, B-123
OBSŁUGA kanału We/Wy, B-93
Odblokowanie, A-118
Odczytywanie tablicy, B-143
Odczytywanie z pliku, B-149, B-153
ODD, B-131
Odejmowanie, B-77
Odsyłacze, A-30, A-113
Odsyłacze, D-4
OF, B-60
Okno wyjściowe, A-79
Okresy aktywności, B-63
On Line, A-115
Opadające zbocze, cewka, B-49
Opadające zbocze, styk, B-46
Opcje kompilatora, A-30, A-103, A-136
Operacja, B-22, B-27
Operacja, D-4
Operacja (IL), B-70, D-4
Operacja binarna, B-22
Operacja binarna, D-4
Operacja impulsowa, B-24
Operacja impulsowa, D-4
Operacja nie przechowywana, B-24
Operacja nie zapisywana, D-4
Operacja z opóźnieniem (IL), D-5
Operacja ze zwłoką (IL), B-70
Operacje (IL), B-69
Operacje specyficzne We/Wy, B-33, B-35
Operator (IL), B-69
Operator EQ (IL), B-86
Operator GE (IL), B-85
Operator GT (IL), B-84
Operator LE (IL), B-83
Operator LT (IL), B-82
Operator NE (IL), B-86
Opis, D-5
Opis, A-19, A-31, A-95, A-154, C-48, C-49, C-54, C-61
Opis projektu, A-20
Oprogramowanie sterownika, A-103
Optymalizator, A-104
OR, A-65
OR_MASK, B-80
Otwarcie projektu, A-20
Otwieranie pliku, B-145, B-146
Otwieranie programu, A-26
Otwórz program, A-114

P

P kwalifikator, A-42
P0 kwalifikator, A-43
P1 kwalifikator, A-43
Pamięć, A-12
Panel sterowania, A-115
Parametr, A-27, A-159
Parametr (funkcja C), C-54, C-62, D-5
Parametr (karta We/Wy), D-5
Parametr karty, A-95, A-156
Parametr OEM, A-157
Parametr OEM (karta We/Wy), D-5
Parzystość, A-33
Pierwiastek kwadratowy, B-117
Plik

koniec wykrywania pliku, B-148
plik nagłówkowy C, C-50, C-55, C-63, C-74, D-5
Pobieranie, A-135
Pobieranie (opcje), A-136
Pobieranie (przygotowanie), A-136
POCHODNA, B-110
Początek, A-23, B-32
Początek kroku makro, D-3
Podłączenie, A-66, A-68, A-69
Podłączenie We/Wy, A-30
Podprogram, A-24, B-4, B-26, B-29, B-34, B-41
Podprogram, D-5
Podstawa, B-8
Połączenie, A-33, A-66, A-68, A-69, A-119, A-136, A-176, B-33, B-35, B-37
Połączenie, D-5
Połączenie (FBD), B-39
połączenie szeregowe, A-33
Połączenie We/Wy, D-5
Pole bitowe, A-131
Porównanie, B-104
Postać wykładnicza, B-115
Potęgowanie, B-116
potomny program, A-24
POW, B-116
Powiększenie, A-53, A-62, A-71
powłoka OS-9 (shell), C-18
Powrót, A-56, A-66
POWRÓT, B-39
Poziom 1 SFC, B-15, B-16
Poziom 1 SFC, D-5
Poziom 2, A-41, A-51
Poziom 2 SFC, B-22
Poziom 2 SFC, D-5
Poziom priorytetu (sterownik NT), C-35
Poziomy zabezpieczeń, A-169
Praca w czasie rzeczywistym, D-5
Prawo dostępu, A-169
Prędkość transmisji, A-33
PrintTime, A-147
Priorytet, C-42
Profiler cyklu, A-143
Program, A-23, A-75, A-143, B-2, D-5
Program najwyższego poziomu, D-5
Program potomny, B-3
Program potomny SFC, B-31
Program potomny SFC, D-5
Program rodzic, D-6
Program- rodzic SFC, D-6
Projekt, A-19, A-162
Projekt, D-6
projektów grupy, A-21
projektów Separator, A-19
projektu dokumentacja, A-21, A-31
projektu opis, A-31
PROM, C-16, C-28
Przejście, A-35, A-41, A-120, B-16
Przejście, D-6
Przejście włączone, B-30
Przejście wyłączone, B-30
Przenieś SFC, A-40
Przenoszenie FBD, A-67
Przenoszenie FC, A-49
Przenoszenie karty, A-93
Przenoszenie programu, A-27
Przenoszenie projektu, A-19
Przenumerowanie, A-51
Przenumeruj, A-40
Przepływ, A-49, B-33, B-37
Przesuń synoptykę, A-131
Przesunięcie w lewo, B-124
Przesunięcie w prawo, B-125
Przyporządkowanie, B-73
Przyporządkowanie (w ST, =), B-58
Przywracanie z kopii, A-22, A-155, A-162, A-164, A-171
Pułapka, A-118, A-120
Pułapka, D-6
Pulpit roboczy debagera, A-32
Punkt, A-99, A-100

R

R_TRIG, B-96
RAND, B-131
Real, A-85, B-9, D-6
REAL, B-89

REDGE, B-56
Rejestr (IL), D-6
REPEAT, B-36, B-61
REPLACE, B-140
RETURN, B-50, B-58
RIGHT, B-141
Róg, A-66
ROL, B-122
ROR, B-123
Równe, B-86
Rozbieżność, B-18
Rozejścia, A-39
Rozejścia, A-36, A-39
Rozejście (FBD), B-39
Rozgrupowanie, A-132
Różniczkowanie, B-110
RS, B-95
Rysunek w tle, A-129
rzeczywisty debager, A-32

S

Sekcja, D-6
Sekcja końcowa, D-6
Sekcja początkowa, D-6
Sekcja sekwencyjna, D-6
Sekcje, A-23
sekwencja $, B-10
Sekwencyjnie, A-23, B-15
Sekwencyjny, B-2
SEL, B-132
Selektor binarny, B-132
SEMA, B-98
SEMAFOR, B-98
Separator, B-53, D-6
Separator projektów, A-19
Sequential Function Chart, D-6
SET, typ cewki, B-47
SFC, A-35, A-104, A-120, A-167, B-15, C-54, D-6
SFC edytor, A-75
SFC galeria, A-45
SFC potomek, B-3
SFC potomny, A-44
SFC potomny program, A-24
SFC zasady, B-30
SFC61, B-28
SFC62, B-28
SFC63, B-28
SFC64, B-28
SHL, B-124
SHR, B-125
Siatka, A-58
SIG_GEN, B-112
SIN, B-121
Sinus, B-121
Skocz, A-41
Skocz do, A-73
Skok, A-51, A-56, A-66, B-39, B-50
Skok do kroku, A-37, B-17
Skok do kroku, D-6
Skrypt, A-144, A-146
SlavesLink, C-26
Śledzenie, A-126, A-128, A-129
Śledzenie zmiennych, A-126
Słownik, A-26, A-80, A-114, A-159, C-49, C-60, D-6
Słowo kluczowe, B-10, B-70
Słowo kluczowe, D-7
Słupek, A-130
Sortowanie, A-84
Spis treści, A-165
Sprawdzanie parzystości/nieparzystości, B-131
Sprawdzenie poprawności przejścia, D-7
SQRT, B-117
SR, B-94
ŚREDNIA, B-106
SSR[x][1].space, C-28
ST, A-44, A-73, A-128, B-53, C-54, C-61, D-7
ST edytor, A-75
STACKINT, B-105
Sterownik, D-7
Stos wielkości analogowych integer, B-105
Strona, A-167
Structured Text, D-7
Styk, A-55, A-65, B-44
Styk, D-7

Styk bezpośredni, B-44
Styk wyzwalany zboczem narastającym, B-45
Styk wyzwalany zboczem opadającym, B-46
Styk zanegowany, B-45
Styl, A-71, A-132
Styl modyfikowany, A-71
Styl normalny, A-71
Styl skasowany, A-71
Symbole (symbole aplikacji), C-6, C-15, C-28, C-37
Symulator, A-32, A-140, A-142, A-144, C-49, C-53, C-60
Synoptyka, A-129
SYSTEM, B-92
Sytuacja pierwotna, D-7
Sytuacja początkowa, B-16, B-30
Szczebel, A-55, A-56, A-61, A-68
Szybki LD, A-44
Szyna zasilająca, A-55, A-56, A-64, B-42
Szyna zasilająca, D-7

T

Tabela konwersji, D-7
Tablica konwersji, A-99, A-101
Tablica symboli, A-179
TAN, B-122
Tangens, B-122
Tekst Strukturalny, B-53
Test, A-46, A-50, A-51, A-115, A-140, B-33, D-7
THEN, B-59
Timer, A-85, B-9, B-13, B-103, D-7
Timer włączany, B-102
Timer wyłączany, B-102
TMR, B-90
TO, B-61
TOF, B-102
TON, B-102
Toolbox, D-7
TP, B-103
Treść kroku makro, B-21
TRUE, A-84
TRUNC, B-117
Tryb cykl po cyklu, D-7
Tryb pracy, A-30
Tryb terminalowy, C-18
TSK_FUNIT, C-20, C-23
TSK_NBTCKSCHED, C-21, C-24, C-29, C-30
tst_main_ex, C-24
TSTART, B-64
TSTOP, B-65
Tworzenie kodu aplikacji, A-102
Tworzenie tablicy, B-142
Typ, A-80, A-82, A-93, A-113, A-156, B-8, D-7
Typ cewki, A-60
Typ karty, A-94
Typ styku, A-60

U

Uaktywnianie, A-116
UNTIL, B-61
Uruchamianie, A-116
Usuwanie biblioteki, A-153
Usuwanie FBD, A-69
Usuwanie FC, A-50
Usuwanie karty, A-94
Usuwanie LD, A-61
Usuwanie podłańcucha, B-134
Usuwanie programu, A-28
Usuwanie SFC, A-40
Usuwanie tekstu, A-73
Uznawanie źródeł za modyfikowane, A-29, A-103

W

Wait, A-148
Wartość analogowa, C-48, C-49, D-1
Wartość bezwzględna, B-114
Wartość zwracana, D-7
Warunek, B-33
Warunek (dla przejścia), D-7
Warunek (dla przejścia), B-28

Warunki (dla przejścia), B-29
We/Wy, A-30, A-93, A-94, A-96, A-114, A-118, A-140, A-155, A-156, A-171, A-173
We/Wy konfiguracja, A-19
Wejście, A-93, A-114, A-140, A-142, A-156, B-6
Wejściowa, D-7
Weryfikacja, A-29, A-102, A-158
Wewnętrzna, D-8
WHILE, B-36, B-60
Większe lub równe, B-85
Większe niż, B-84
Wielkość aplikacji, C-45
Wielozadaniowość, C-15, C-27, C-37
WISAKER.EXE (NT), C-31
Wklejanie FBD, A-69
Wklejanie FC, A-50
Wklejanie LD, A-61
Wklejanie SFC, A-40
Wklejanie tekstu, A-73
Wklejanie zmiennej, A-84
Wspólne, D-8
Wspólne dane, A-162
Wstawianie cewki, A-59
Wstawianie elementu FBD, A-67
Wstawianie elementu FC, A-47
Wstawianie FBD, A-70
Wstawianie gniazda, A-93
Wstawianie pliku, A-74
Wstawianie podłańcucha, B-135
Wstawianie styku, A-59
Wstawianie szczebla, A-60
Wstawianie zmiennej, A-42
Wycinanie FBD, A-69
Wycinanie FC, A-50
Wycinanie LD, A-61
Wycinanie SFC, A-40
Wycinanie tekstu, A-73
Wycinanie zmiennej, A-84
Wydruk programu, A-77
Wyjście, A-93, A-114, A-140, A-156, B-6
Wyjściowa, D-8
Wykładnicze, B-9
Wykonywanie 1 cyklu, A-118
Wynik bieżący (IL), B-70, D-8
Wynik bieżący(IL), B-69
Wyrażenie, D-8
Wyrażenie stałe, B-8
Wyrażenie stałe, D-8
Wyświetlanie tekstowe, A-129

X

XOR_MASK, B-81

Z

Zabezpieczanie, A-97, A-153, A-169
Zabezpieczenia poziomy, A-169
Zabezpieczenie, A-21
Zablokowane We/Wy, D-8
Zachowywanie, C-88
zadanie ISA (OS9), C-9, C-20
zadanie ISAKER (OS9), C-10
zadanie ISANET (OS9), C-11
zadanie ISATST (OS9), C-10
Zakres, A-80, A-82
Zakres, D-8
Załadowanie, A-117
Zamień, A-41, A-51, A-61, A-69, A-73
Zamiennik nazwy, A-62
Zamykanie pliku, B-147
Zanegowana cewka, B-47
Zanegowane połączenie, A-66, A-68
Zanegowany styk, B-45
Zapisywanie listy, A-126
Zapisywanie tablicy, B-144
Zapisywanie w pliku, B-151, B-155
Zarządzanie programami, A-23
Zarządzanie projektami, A-19
Zasoby, A-30
Zastępowanie podciągu, B-140
Zatrzymywanie, A-116
Zawieranie kodu źródłowego, A-136
Zaznaczanie elementu FBD, A-67
Zaznaczanie elementu FC, A-48
Zaznaczanie synoptyki, A-131
Zbieżność, B-18

Zbocze, D-8
Zbocze wyzwalające styk, B-45, B-46
Zdefiniowane słowo, A-81, A-84, A-85, B-14
Zdefiniowane słowo, D-8
Zegar systemowy (VxWorks), C-20
Zejścia, A-39
Zejścia, A-36, A-39
Zespoły We/Wy, A-156
Zliczane w górę, B-98
Zliczanie w dół, B-99
Zliczanie w górę /w dół, B-100
Zmiana nazwy biblioteki, A-153
Zmiana wymiarów FC, A-50
Zmień rozmiary synoptyki, A-131
Zmienna, A-26, A-42, A-60, A-66, A-70, A-73, A-80, A-113, A-114, A-119, A-159, A-171, B-10, B-38
Zmienna, D-8
Zmienna binarna, A-85
Zmienna o reprezentacji bezpośredniej, B-11
Zmienna reprezentowana wprost, A-96
zmienna We/Wy, C-48, C-49
Zmienna We/Wy, D-8
Znacznik (SFC), B-15, D-8
Znajdowanie podłańcucha, B-136
Znajdź, A-41, A-51, A-61, A-69, A-73, A-79
Zrzut, A-127