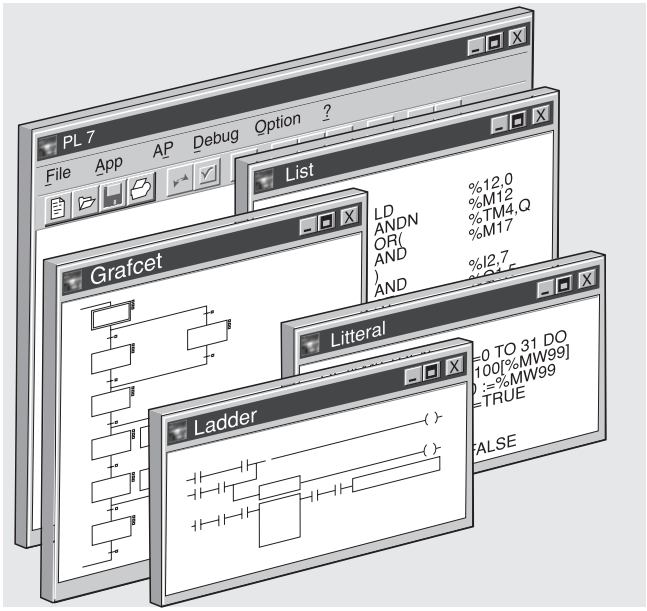
PL7 Micro / Junior / Pro

Instrukcja Użytkownika
(wersja V3. ■)

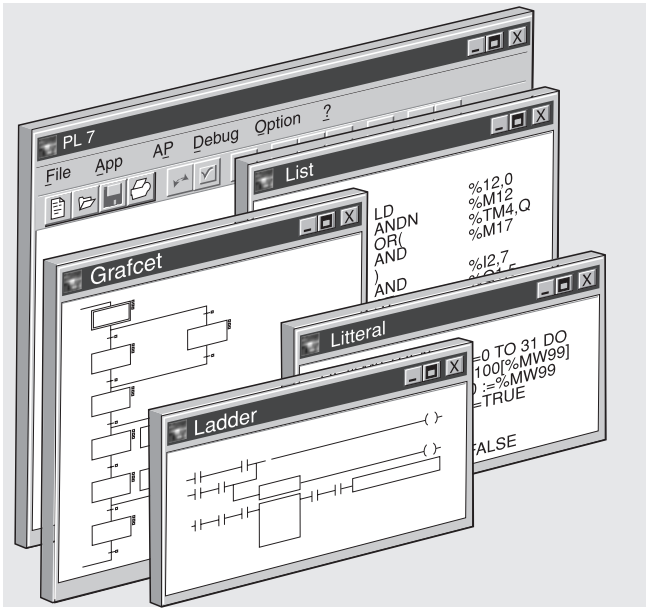


GROUPE SCHNEIDER

■ Merlin Gerin ■ Modicon ■ Square D ■ Telemecanique

PL7 Micro / Junior / Pro

Instrukcja Użytkownika
(wersja V3. ■)



GROUPE SCHNEIDER

■ Merlin Gerin ■ Modicon ■ Square D ■ Telemecanique

REF. **TLX DR PL7 30E**

ART. **87727**

Schneider Automation Inc.
One High Street
North Andover, MA 01845
Tél.: (1) 508 794 0800
Fax : (1) 508 975 9010

Schneider Automation S.A.
245, route des Lucioles - BP 147
F-06903 Sophia Antipolis
Tél. : (33) (0)4 92 96 20 00
Fax : (33) (0)4 93 65 37 15

Schneider Automation GmbH
Steinheimer Straße 117
D-63500 Seligenstadt
Tél. : (49) 6182 81 2584
Fax : (49) 6182 81 2860

Tłumaczenie - BATI (0-63)243-45-81

Lipiec 1998 (837)

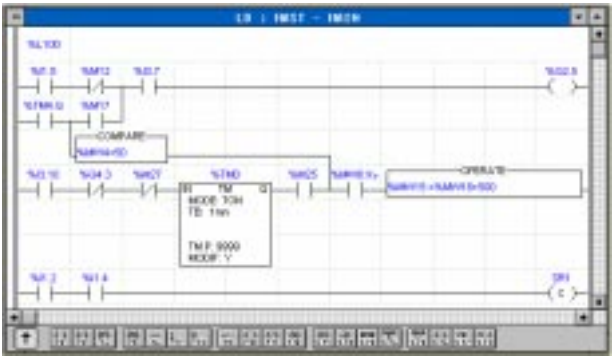
1.1 Język PL7 - przedstawienie

1.1-1 Charakterystyka wstępna

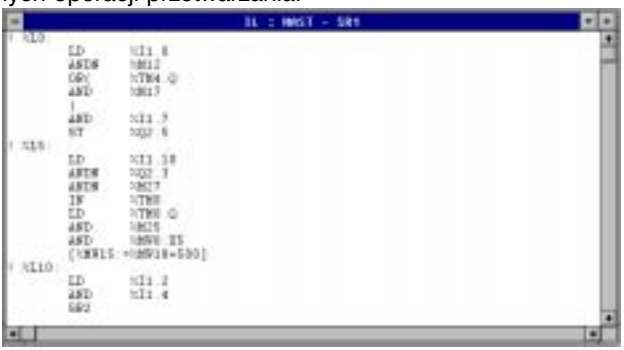
Język PL7 Junior stanowi oprogramowanie dla sterowników TSX 37 i TSX/PMX/PCX 57 pracujących w środowisku Windows. Natomiast język PL7 Micro służy do programowania tylko sterowników TSX 37.
Język PL7 Pro dodatkowo daje, w odniesieniu do języka PL7 Junior, możliwość tworzenia bloków funkcyjnych typu DFB (*Derived Function Block* - blok tworzony przez użytkownika) oraz ekranów kontroli przebiegu procesu sterowania (*runtime screen*).
Oprogramowanie PL7 Prodyn stanowi narzędzie użytkowe sterowników serii TSX 37 oraz TSX/PMX/PCX 57 umożliwiające sterowanie, diagnostykę i ich obsługę. Nie może być ono użyte do tworzenia czy też modyfikowania aplikacji.

Oprogramowanie PL7 (1) tworzą:

- Graficzny język *Ladder* (LD), służący do definiowania diagramów drabinkowych (*Ladder diagrams*), przeznaczonych szczególnie do przetwarzania kombinacyjnego. Przy tworzeniu diagramów korzysta się ze styków (*contact*), cewek (*coil*) oraz bloków (*block*). Obliczenia numeryczne zapisuje się wewnątrz bloków.



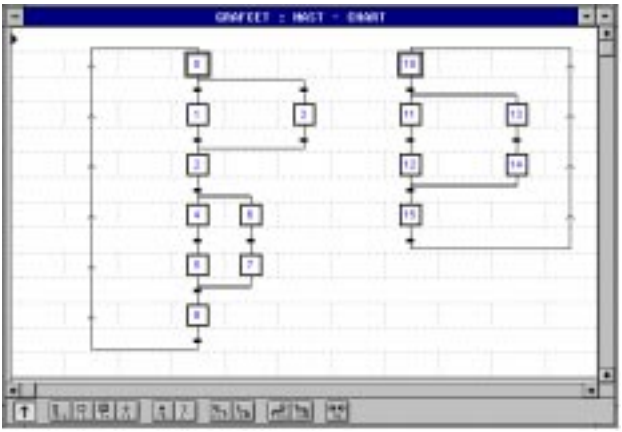
- Język logiczny List (IL), który stanowi kod maszynowy dla zapisywania logicznych i numerycznych operacji przetwarzania.



(1) Nazwa PL7 używana w niniejszej dokumentacji odnosi się do PL7 Junior, PL7 Micro, PL7 Pro oraz PL7 Prodyn.

- Strukturalny język tekstowy (ST) umożliwiający strukturalny zapis przetwarzania logicznego i numerycznego.

- Graficzny język *Grafcet* służący do graficznego i strukturalnego przedstawiania sekwencji systemu sterowania.



Wymienione powyżej języki zawierają zdefiniowane wstępnie bloki funkcyjne (zegary, liczniki, itp.), które można wykorzystać przy tworzeniu danej aplikacji (komunikacja analogowa, odliczanie, itp.) oraz w funkcjach specjalnych (zarządzanie czasem, łańcuchy znaków, itp.).

Obiektom językowym można przypisywać symbole.

Oprogramowanie PL7 jest zgodne z normą IEC 1131-3. Warunki, przy spełnieniu których zostaje zachowany standard zestawiono w tabelach zgodności zamieszczonych w *Dodatku*: część B, rozdział 6.

1.1-2 Struktura jednozadaniowa

Jest to struktura domyślna. Program zawiera jedno zadanie - główne (*master task*).

Zadanie główne *Master*
Zadanie to może być cykliczne (ustawienie domyślne) lub okresowe. W przypadku działania cyklicznego kolejne "przebiegi" zadania są łączone jeden z drugim, bez przerw. Przy działaniu okresowym "przebiegi" zadania następują po sobie w zdefiniowanym przez użytkownika okresie.

1.1-3 Struktura wielozadaniowa

Struktura wielozadaniowa w przypadku sterowników TSX 37 oraz TSX 57 pozwala na uzyskanie lepszych efektów poprzez zastosowanie aplikacji pracujących w czasie rzeczywistym, w których każdej aplikacji przypisany jest specjalny program. Każdy z tych programów jest zarządzany przez zadanie.
Zadania te są w stosunku do siebie niezależne i są wykonywane równolegle przez główny procesor, który decyduje zarówno o ich priorytecie jak i o ich wykonaniu.

Zalety struktury tego typu:
- optymalne wykorzystanie mocy procesora,
- uproszczenie projektowania; każde zadanie zapisuje się i poprawia niezależnie.od pozostałych,
- nadanie aplikacji określonej struktury; każde zadanie pełni unikatową funkcję,
- optymalizacja dostępności.

System wielozadaniowy może składać się z zadania głównego MASTER, szybkiego FAST oraz 8 - 64 (w zależności od rodzaju procesora) zadań wyzwalanych zdarzeniami.

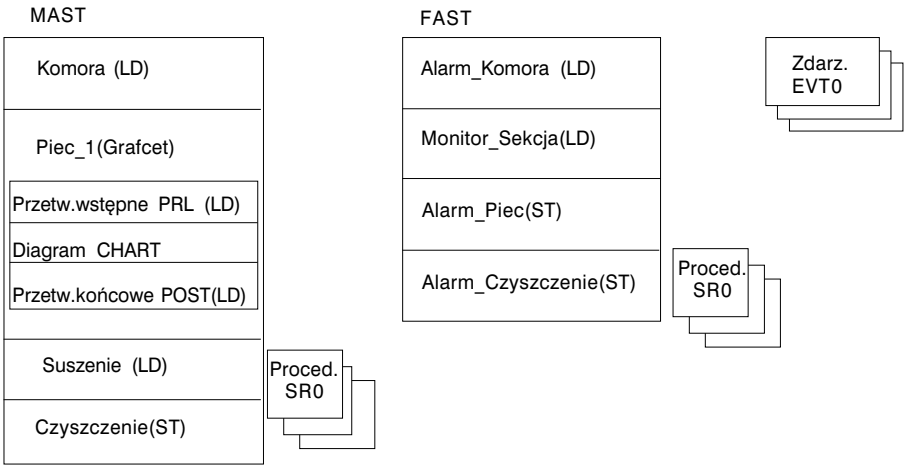
Zadanie szybkie FAST
Zadanie szybkie (opcjonalne) jest zadaniem okresowym i służy do realizacji szybkich operacji przetwarzania z wyższym priorytetem w stosunku do zadania głównego MASTER. Zaprogramowane zadanie jest automatycznie uruchamiane przez system podczas inicjacji (*start-up*). Można je zatrzymywać oraz uruchamiać za pośrednictwem bitu systemowego.

Zadania wyzwalane zdarzeniami (*event-triggered task*)
W odróżnieniu od zadania opisanego powyżej zadania tego typu nie są okresowe. Są one wyzwalane (uruchamiane) przez pewne moduły. Te zadania mają najwyższy priorytet. Ich przetwarzanie odbywa się odpowiednio szybko tak, że nie zakłócają wykonywania innych zadań.

1.1-4 Programowane strukturalne i modułowe

Zadania w języku PL7 składają się z kilku części (zwanych sekcjami) oraz procedur.

Każdą z sekcji programuje się w języku, który najlepiej odpowiada rodzajowi zastosowanego procesu.



Podział na sekcje oznacza, że można stworzyć program stukturalny oraz można z łatwością generować lub też dołączać moduły programu.

Procedury (podprogramy) mogą być wywoływane z dowolnej sekcji zadania, do którego należą oraz z innych procedur tego samego zadania.

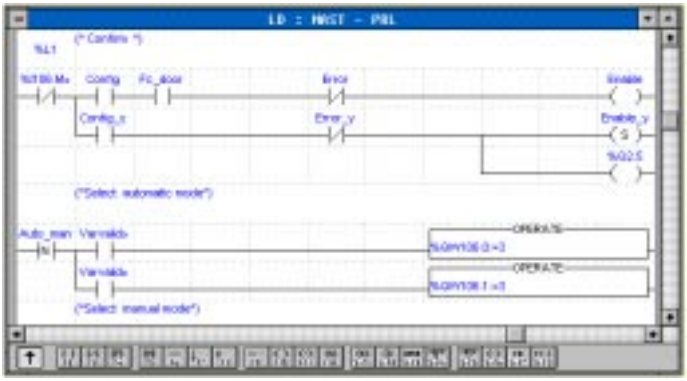
1.1-5 Programowanie symboliczne

Użytkownik może wprowadzać lub wyświetlać obiekty za pomocą:

- ich adresów (na przykład: %Q2.5),
- łańcucha znaków (max. 32 znaki) rozpoznawanego jako symbol (na przykład Fc_door).

Uwaga
Obiekty sprzężone z blokiem typu DFB (blok własny użytkownika) są obiektami czysto symbolicznymi.

Przykład: symboliczne wyświetlanie obiektów w języku *Ladder*.



W języku *Ladder* można jednocześnie wyświetlać adresy i symbole.

Wykorzystywane symbole można wprowadzać przed lub podczas edycji programu.

Baza symboli, zarządzana przy pomocy edytora zmiennych (*variables*) ma charakter globalny dla danej stacji (sterownika).

Uwaga
Niektóre moduły specjalne umożliwiają automatyczne przypisywanie symboli obiektom, które są z nimi sprzężone.

1.1-6 Instrukcje języka PL7

Wszystkie języki oprogramowania PL7 posługują się tymi samymi instrukcjami.

Wszystkie instrukcje są szczegółowo opisane w części B niniejszej dokumentacji. W celu uproszczenia instrukcje zostały podzielone na dwie grupy: instrukcje podstawowe oraz instrukcje złożone.

Instrukcje podstawowe

Instrukcje te obejmują podstawowe instrukcje logiczne, wstępnie zdefiniowane bloki funkcyjne oraz arytmetyczne i logiczne operacje na liczbach całkowitych.

Instrukcje złożone

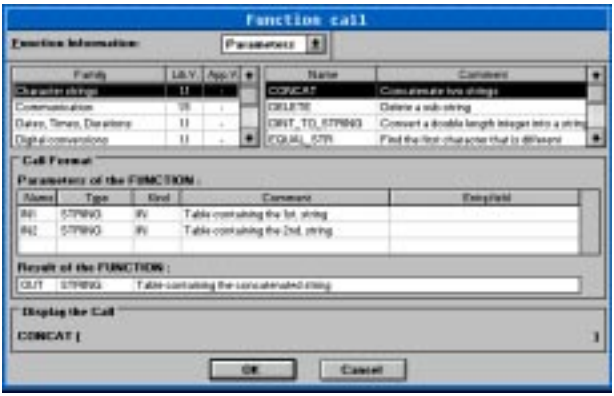
Instrukcje te używane są przy programowaniu zaawansowanym.

Instrukcje te dają dwojaki efekt:

- w stosunku do języka PL7: zwiększają możliwości języka poprzez zastosowanie funkcji specjalnych (działanie na łańcuchach znaków, zarządzanie czasem, itp.),
- w stosunku do aplikacji: umożliwiają zaprogramowanie specjalnych funkcji dla danej aplikacji. Mogą to być na przykład funkcje wykorzystywane do nawiązania komunikacji:
  - PRINT do wysyłania komunikatu zawierającego łańcuch standardowych znaków do terminala albo na drukarkę,
  - SEND do wysyłania komunikatu do aplikacji,
  - PID: funkcja regulatora PID.

Wywołanie funkcji - ekran pomocy

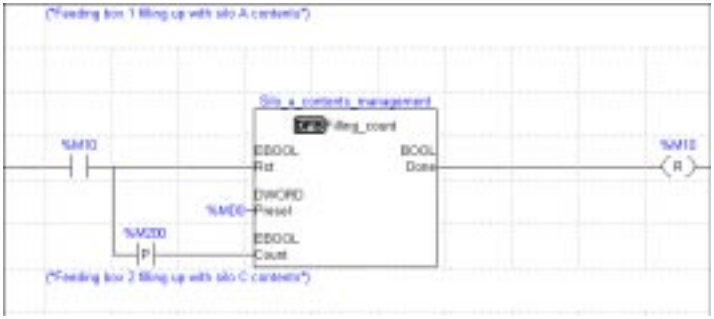
Ekran wywołania funkcji umożliwia dostęp do wszystkich funkcji języka. Można go wywoływać w każdym momencie, nawet podczas programowania.



1.1-7 Bloki funkcyjne

Do tworzenia bloków typu DFB dla sterowników serii Premium służy język PL7 Pro. Tworzy się je w strukturalnym języku tekstowym (ST) i mogą być one użyte w sekcji lub procedurze bez względu na to w jakim języku są one napisane (można je stosować również w języku PL7 Junior).

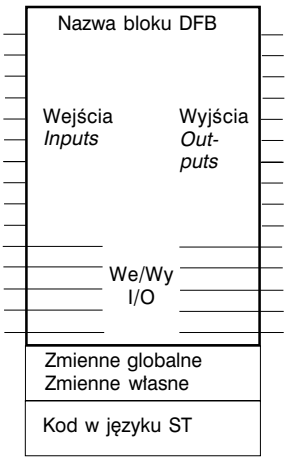
Przykład zastosowania bloku DFB w programie napisanym w języku *Ladder*.



Blok funkcyjny DFB składa się z następujących elementów:

- nazwy
- parametrów wejść i wyjść
- zmiennych globalnych i własnych
- kodu w języku strukturalnym ST.

Blok typu DFB może mieć maksymalnie 15 wejść i/lub We/Wy (I/O) oraz 15 wyjść i/lub We/Wy (I/O).



Raz utworzony blok DFB może być wykorzystywany w aplikacji wielokrotnie. Użytkownik programuje dla danego języka model DFB (rozpoznawany jako typu bloku DFB), a przed każdym użyciem definiuje nazwę bloku korzystając przy tym z edytora zmiennych lub też ekranu wywołania funkcji.

1.2 Adresy obiektów

1.2-1 Definiowanie głównych obiektów logicznych

Bity I/O (wejść i wyjść)

Bity te stanowią logiczne odwzorowanie elektrycznych stanów wejść i wyjść.

Bity wewnętrzne (*internal bits*)

Bity wewnętrzne %Mi służą do zapamiętywania stanów pośrednich w trakcie wykonywania programu.

Uwaga: Nie używane bity I/O nie mogą być wykorzystane jako bity wewnętrzne.

Bity systemowe

Bity systemowe %S0 do %S127 umożliwiają monitorowanie poprawności działania sterownika oraz przebiegu realizacji programu. Rola oraz sposób wykorzystywania tych obiektów zostały szczegółowo opisane w rozdziale 3.1 części B.

Bity bloków funkcyjnych

Wartości bitów bloków funkcyjnych odpowiadają stanom wyjść bloków. Wyjścia bloków mogą być podłączone bezpośrednio lub wykorzystywane jako obiekty.

Bity wydzielone ze słów (*wors extract bits*)

Przy użyciu PL7 możliwe jest wydzielenie dowolnego z 16 bitów tworzących słowo.

Bity statusu kroków i skoków (*macro-step*) języka *Grafcet*

Do identyfikowania statusu kroku *i* diagramu, skoku *j* oraz kroku *i* w skoku *j* służą odpowiednie bity statusu: dla kroków %Xi, dla skoków %XMj oraz dla kroków w skokach %Xi.j (dla wejść i wyjść kroków w skokach - odpowiednio Xj.IN oraz Xj.OUT).

Lista argumentów operacji na bitach

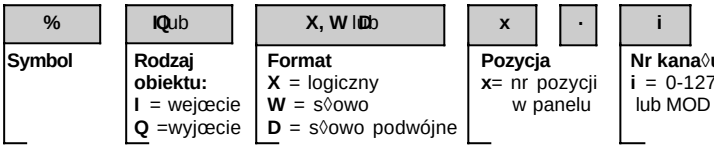
W tabeli poniżej zestawiono listę wszystkich typów argumentów logicznych.

Rodzaj	Adres (lub wartość)	Dostęp w trybie zapisu **(1)**	Patrz	
			Rozdz.	Część
Wartość bezpośr.	0 lub 1 (Fałsz lub Prawda)	–	1.2-4	A
Bity wejść Bity wyjść	%Ix.i lub %IXx.i %Qx.i lub %QXx.i	nie tak	1.2-2 1.2-3	A
Internal bits	%Mi lub %MXi	tak		–
System bits	%Si	w zależności od *i*	3.1	B
Bity bloków funkcyjnych	np.: %TMi.Q %DRi.F.....	nie	1.2-5	A
Bity *Grafcet-u*	%Xi, %XMj, %Xj.i...	tak	5.2	A
Bity ze słów	np.: %MW10:X5	zal. od typu słowa	1.2-4	A

(1) Zapisywanie przez program lub w trybie poprawiania (*adjust mode*) za pomocą terminala.

1.2-2 Adresowanie obiektów związanych z modułem I/O TSX 37

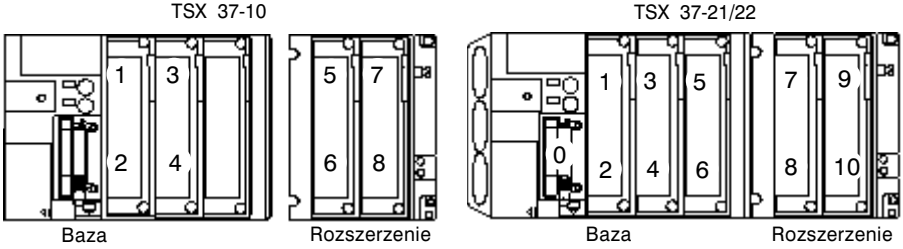
Adresy podstawowych słów oraz bitów obiektów dla modułów I/O (We/Wy) definiuje się w następujący sposób:



- **Rodzaj obiektu**
  I lub **Q**: Fizyczne wejścia i wyjścia modułu dokonującego wymiany w sposób ukryty (niejawny) przy każdym wykonaniu zadania, do którego są one podłączone.
  Uwaga: Jeżeli istnieje taka potrzeba, to na żądanie (*request*) aplikacji (patrz opis w instrukcji funkcji aplikacji), można dokonywać wymiany danych pozostałych typów (status, słowa poleceń, itp.).

- **Format**
  Dla obiektów zapisanych w formacie logicznym można pomijać *x*. Formaty pozostałych obektów takich, jak bajt, słowo, słowo podwójnej precyzji zostały opisane w rozdziale 1.2-4.

- **Numer i pozycja kanału**
  Sterowniki TSX`37 mają budowę modułową 2-adresową. Pozycję każdego z modułów sterownika TSX 37 (modułu bazowego *Base* i modułu rozszerzenia *Extension*) pokazano na rysunku poniżej.



Moduły standardowe adresuje się jak dwa moduły półformatowe (patrz tabela poniżej).

Na przykład moduł zawierający 64 We/Wy jest widziany jak dwa moduły: 32-wejściowy moduł umieszczony w slocie 5 oraz 32-wyjściowy, umieszczony w slocie 6.

Moduł	Półformat		Format standardowy				
	4 Q	8 Q	12 I	28 I/O	32 I	32 Q	64 I/O
Nr kanału: i	0 to 3	0 do 7	0 do 11	0 do 15 0 do 11	0 do 15 0 do 15	0 do 15 0 do 15	0 do 31 0 do 31
Pozycja i nr kan. (x = pozycja)	x.0 do x.3	x.0 do x.7	x.0 do x.11	x.0 do x.15 (x+1).0 do (x+1).11	x.0 do x.15 (x+1).0 do (x+1).15	x.0 do x.15 (x+1).0 do (x+1).15	x.0 do x.31 (x+1).0 do (x+1).31

Uwaga:
Numer kanału można zastąpić symbolem "MOD", co umożliwia dostęp do danych globalnych modułu.

- **Ranga:** za numerem kanału można wstawić dodatkowy indeks. Umożliwia to rozróżnienie obiektów tego samego typu sprzężonych z tym samym kanałem (patrz opis funkcji aplikacji w odpowiedniej instrukcji).
  ERR: sygnalizuje błąd modułu lub kanału.
  Przykłady: %I4.MOD.ERR : informacja o błędzie w module 4
  %I4.3.ERR : informacja o błędzie w kanale 3 modułu 4.

Uwaga:
Podczas adresowania operacji pomędzy elementami sieci oraz podczas adresowania zdalnego wejść i wyjść należy do numeru pozycji w module dodać kompletną ścieżkę dostępu do stacji.

Przykłady

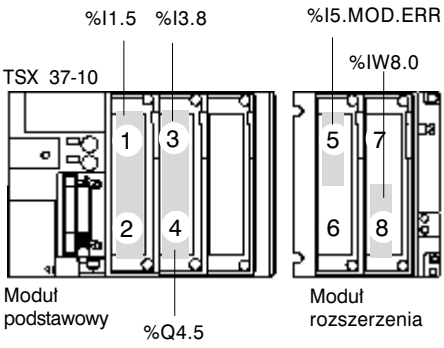
%I1.5 kanał wejściowy nr 5 modułu umieszczonego na pozycji nr 1.

%I3.8 kanał wejściowy nr 8 modułu o formacie zwykłym umieszczonego na pozycjach nr 3 i 4.

%Q4.5 kanał wyjściowy nr 5 modułu o formacie standardowym umieszczonego na pozycjach nr 3 i 4.

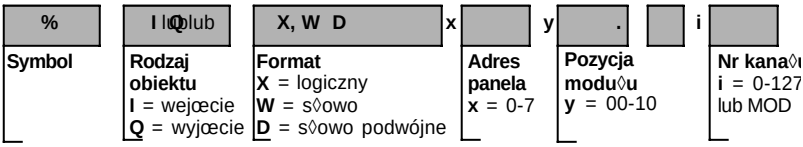
%I5.MOD.ERR informacja o błędzie modułu umieszczonym na pozycji nr 5.

%IW8.0 kanał wejściowy nr 0 modułu półformatowego umieszczonego na pozycji nr 8.

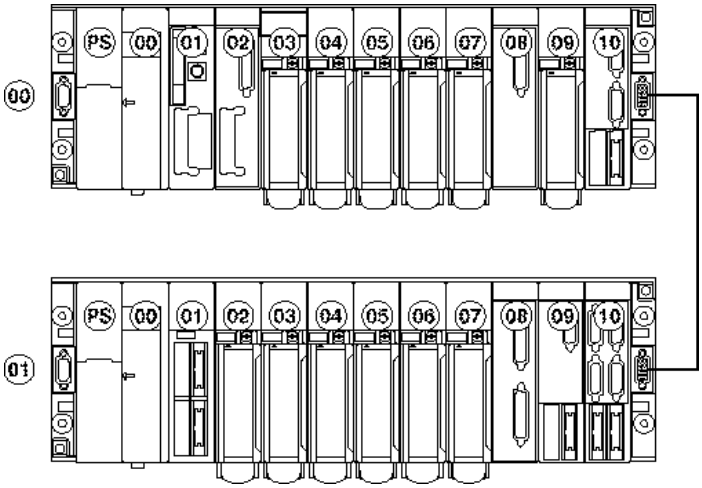


1.2-3 Adresowanie obiektów modułu I/O TSX/PMX/PCX 57

Składnia adresów słów podstawowych i bitów obiektów w module I/O jest następująca:



- **Rodzaj obiektu**
  I lub **Q**: Fizyczne wejścia i wyjścia modułu dokonującego wymiany w sposób ukryty (niejawny) przy każdym wykonaniu zadania, do którego są one podłączone.
  Uwaga: Jeżeli istnieje taka potrzeba, to na żądanie (*request*) aplikacji (patrz opis funkcji specjalnych aplikacji), można dokonywać wymiany danych pozostałych typów (status, słowa poleceń, itp.).
- **Format** (rozmiar)
  Dla obiektów zapisanych w formacie logicznym można pomijać X. Formaty pozostałych obiektów (bajt, słowo, słowo podwójne) opisano w rozdziale 1.2-4.
- **Adresowanie kanałów**
  Adres kanału zależy od adresu panela (*rack*), fizycznej pozycji modułu w panelu oraz od numeru kanału.



Uwaga:

- Listy obiektów sprzężonych z poszczególnymi modułami należy szukać w instrukcji zawierającej opis funkcji aplikacji.
- Zdalne adresowanie wejść i wyjść I/O opisano w instrukcji funkcji związanych z aplikacjami zatytułowanej "*Common features of application-specific functions*".

Adres panela (x) i pozycja modułu (y)

Panele TSX-a	RKY 6	RKY 8	RKY 12	RKY 4EX	RKY 6EX	RKY 8EX	RKY 12EX
Adres panela: x	0	0	0	0 ÷ 7	0 ÷ 7	0 ÷ 7	0 ÷ 7
Pozycja modułu: y	00 ÷ 04	00 ÷ 06	00 ÷ 10	00 ÷ 02	00 ÷ 04	00 ÷ 06	00 ÷ 10

Uwaga:
Panel, który zawiera procesor ma zawsze adres 0.

Numer kanału (i)

Moduły TSX DEY ..../DSY	64 I/O	32 I/O	16 I/O	8 I/O
Numer kanału: i	0 do 63	0 do 31	0 do 15	0 do 7

Uwaga:
W celu uzyskania dostępu do informacji globalnych dla modułu, numer kanału zastępuje się symbolem "MOD".

- **Ranga:** za numerem kanału można wstawić dodatkowy indeks. Umożliwia to rozróżnienie obiektów tego samego typu sprzężonych z tym samym kanałem (patrz opis funkcji aplikacji w odpowiedniej instrukcji).
- **ERR**: sygnalizuje błąd modułu lub kanału.

Przykłady:
- %I104.MOD.ERR : informacja o błędzie w module umieszczonym na pozycji 4 panela o adresie 1.
- %I104.3.ERR : informacja o błędzie w kanale 3 modułu umieszczonego na pozycji 4 panela o adresie 1.

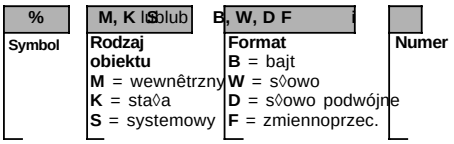
Uwaga:
Podczas adresowania operacji pomędzy elementami sieci oraz podczas adresowania zdalnego wejść i wyjść należy do numeru pozycji w module dodać kompletną ścieżkę dostępu do stacji.

Przykłady:

- **%I102.5** : Kanał wejściowy nr 5 modułu umieszczonego na pozycji 2 panela o adresie 1.
- **%Q307.2** : kanał wyjściowy nr 2 modułu umieszczonego na pozycji 7 panela o adresie 3.
- **%I102.MOD.ERR** : informacja o błędzie modułu umieszczonego na pozycji 2 panela o adresie 1.

1.2-4 Adresowanie słów

Adresowanie słów modułów I/O opisano w rozdziałach 1.2-2 oraz 1.2-3. Pozostałe słowa używane w języku PL7 (z wyjątkiem słów sieciowych oraz słów bloków funkcyjnych) adresuje się w sposób następujący:



- **Rodzaj obiektu**

M słowa wewnętrzne, czyli te, w których zapisywane są wartości w trakcie wykonywania programu. Słowa te są zapamiętywane w strefie danych (*data zone*) pamięci.

K słowa stałe służące do przechowywania wartości stałych i komunikatów alfanumerycznych. Ich zawartość może być zapisywana i modyfikowana jedynie przez terminal. Słowa te są przechowywane w tym samym miejscu co program. Stąd też mogą one wykorzystywać pamięć FLASH EPROM.

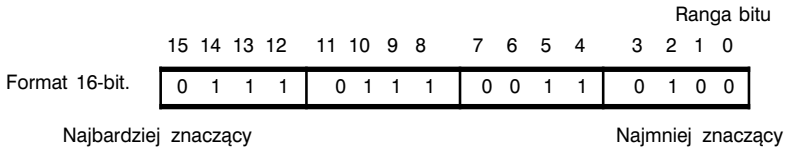
S słowa systemowe. Pełnią one różne funkcje:
- część zawiera informacje o statusie systemu, dostępne za pośrednictwem słów %SWi (czasy działania systemu i aplikacji).
- pozostałe służą do wykonywania odpowiednich operacji w aplikacji (tryby pracy, itp.). Słowa systemowe są szczegółowo opisane w rozdziale 3, część B.

- **Format**

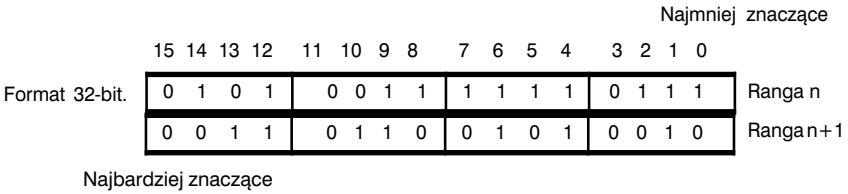
Słowa obiektowe można adresować w PL7 przy pomocy 4 różnych formatów:

B bajt: ten format jest używany wyłącznie do działań na łańcuchach słów.

W słowo pojedynczej precyzji: te 16-bitowe słowa mogą zawierać wartości algebraiczne z przedziału od -32 768 do 32 767.



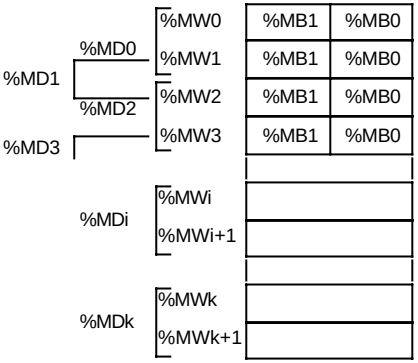
D słowa podwójnej precyzji: te 32-bitowe słowa mogą zawierać wartości algebraiczne z przedziału od - 2 147 483 648 do 2 147 483 647. Są one zapisywane do pamięci w dwu następujących po sobie słowach pojedynczej precyzji.



F słowo zmiennoprzecinkowe: stosowany format zmiennoprzecinkowy jest zgodny z normą IEEE Std 754-1985 (równoważny w stosunku do IEC 559). Słowa mają długość 32 bitów, co odpowiada długości pojedynczych słów zmiennoprzecinkowych. Przykładowe wartości zmiennoprzecinkowe:
1285.28
12.8528E2

Nakładanie się obiektów:
Bajty, słowa pojedynczej i podwójnej precyzji oraz słowa zmiennoprzecinkowe są zapamiętywane w strefie danych, w tym samym obszarze pamięci.
Stąd też może wystąpić nakładanie się następujących obiektów:

- słów podwójnej precyzji %MDi oraz słów pojedynczych %MWi, %MWi+1 (słowo %MWi zawierające mniej znaczące bity i słowo %MWi+1 zawierające bardziej znaczące bity słowa %MDi).
- słów pojedynczej precyzji %MWi oraz bajtów %MBj oraz %MBj +1 (gdzie j=2.i).
- słów zmiennoprzecinkowych %MFk i słów pojedynczej precyzji %MWk oraz %MWk+1.



Przykłady:
%MD0 odpowiadają %MW0 i %MW1
%MW3 odpowiadają %MB7 i %MB6
%KD543 odpowiadają %KW543 i %KW544
%MF10 odpowiadają %MW10 i %MW11.

Wartości bezpośrednie (*immediate values*)

Są to wartości algebraiczne, których format jest podobny do formatu słów pojedynczych i podwójnych (słowa 16- i 32-bitowe), które umożliwiają przypisywanie tym słowom wartości. Słowa te zapamiętywane są w pamięci programu. Składnia jest następująca:

Typ		Składnia	Ograniczenie od dołu	Ograniczenie od góry
Logiczny		0 lub 1 (Fałsz, Prawda)		
Całkowita (p.10) (dziesiętna)	pojedyncza	1506	-32768	+32767
	podwójna	578963	-2 147 483 648	2 147 483 647
Całkowita (p.2) (binarna)	pojedyncza	2#1000111011111011011	2#10...0	2#01...1
	podwójna	2#1000111011111011011 1111111011111011111	2#10...0	2#01...1
Całkowita (p.16) (hexadecym.)	pojedyncza	16#AB20	16#0000	16#FFFF
	podwójna	16#5AC10	16#000000000	16#FFFFFFFF
Zmiennoprzecinkowa		-1.32E12	-3.402824E+38(1) 1.175494E-38(1)	-1.175494E-38(1) 3.402824E+38(1)
Łańcuch znaków		'aAbBcC'		

(1) z wyłączeniem wartości ograniczeń

Adresowanie słów w sieci

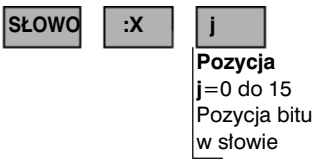
Adresowanie słów w sieci opisano w instrukcji dotyczącej komunikacji.

W sieci posługujemy się specyficznym rodzajem obiektów - słowami wspólnymi (*common words*). Są to pojedyncze słowa obiektowe (16-bitowe) wspólne dla wszystkich stacji podłączonych do sieci komunikacyjnej.

Adres: %NW{i.j}k
gdzie: *i* = nr sieci od 0 do 127, *j* = nr stacji 0 do 31 oraz *k* = nr słowa 0 do 3

Bity wydzielone ze słów

PL7 umożliwia wydzielenie dowolnego z 16 bitów ze słowa pojedynczego. Do adresu słowa dołącza się pozycję bitu, który ma być wydzielony. Składnia jest następująca:



Przykład : %MW10:X4 = bit nr 4 słowa wewnętrznego %MW10
%QW5.1:X10 = bit nr 10 słowa wyjściowego %QW5.1

Uwaga: Bity mogą być wydzielane również ze słów indeksowanych.

Zestawienie podstawowych słów obiektowych oraz sprzężonych z nimi bitów

R w poniższej tabeli oznacza odczyt, natomiast W - zapis.

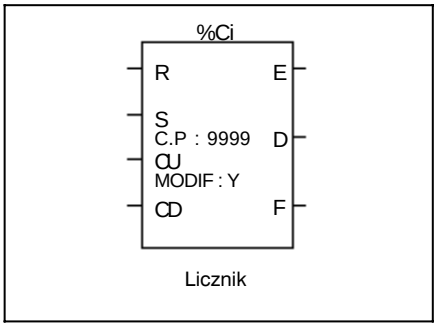
Sprzężone słowa i bity	Rodzaj	Adres	Ograniczenie	Możliwości
Słowa wewn.	pojedyncze	%MWi	(1)	R/W
	podwójne	%MDi	(1)	R/W
	zmiennoprzec.	%MFi	(1)	R/W
	bajt (2)	%MBi	(1)	R/W
Słowa stałe	pojedyncze	%KWi	(1)	R/W (3)
	podwójne	%KDi	(1)	R/W (3)
	zmiennoprzec.	%KFi	(1)	R/W (3)
	bajt (2)	%KBi	(1)	R/W (3)
Słowa modułu We/Wy (I/O)	pojedyncze, wejściowe	%IWxy.i	0⎮i⎮127	R
	podwójne, wejścowe	%IDxy.i	0⎮i⎮126	R
	pojedyncze, wyjściowe	%QWxy.i	0⎮i⎮127	R/W
	podwójne, wyjściowe	%QDxy.i	0⎮i⎮126	R/W
Słowa *Grafcet-u*	pojedyncze	%Xi.T	0⎮i⎮249	R
	podwójne	%Xj.i.T	0⎮j⎮63 0⎮i⎮249	R
Słowa wspólne	w sieci	%NW{i.j}k	0⎮i⎮127 0⎮j⎮31 0⎮k⎮3	R/W
Słowa systemowe	pojedyncze	%SWi	0⎮i⎮255	R/W (4)
	podwójne	%SDi	0⎮i⎮254	R/W (4)
Bity wydzielone ze słów	bit j słowa wewn.	%MWi:Xj	0⎮j⎮15	R/W
	bit j słowa stałego	%KWi:Xj	0⎮j⎮15	R/W (3)
	bit j słowa wejściowego	%IWi:Xj	0⎮j⎮15	R
	bit j słowa wyjściowego	%QWi:Xj	0⎮j⎮15	R/W
	bit j słowa systemowego	%SWi:Xj	0⎮j⎮15	R/W (4)
	bit j słowa wspólnego sieć 0	%NW{j}k:Xm	0⎮m⎮15	R/W

(1) Ograniczenie od góry zależy od rozmiaru dostępnej pamięci oraz liczby słów zadeklarowanej podczas konfigurowania oprogramowania.
(2) Te obiekty występują tylko w adresach początkowych łańcuchów znaków %MBi:L lub %KBi:L (patrz rozdział 2.8-1, część B).
(3) Zapisu można dokonać tylko z poziomu terminala.
(4) Zapis w zależności od *i*.

1.2-5 Obiekty związane z blokami funkcyjnymi

Bloki funkcyjne mają specyficzne bity i słowa.

- **Bity obiektowe:**
  Odpowiadają one wyjściom bloków. Są dostępne za pośrednictwem binarnych poleceń testowych.
- **Słowa obiektowe:**
  Odpowiadają one:
  - parametrom konfiguracyjnym bloku. Dostęp do nich odbywa się niekiedy za pomocą programu (np. wartości wstępnie nastawione) lub bez jego udziału (np. podstawa czasu),
  - wartościom bieżącym (np. %Ci,V - bieżąca wartość licznika).



Licznik

Lista bitów i słów bloków funkcyjnych dostępnych za pośrednictwem programu

Wstępnie zdef. bloki funkcyjne	Bity i słowa związane z blokiem		Adres	Dostęp do zapisu	Patrz Cz. B
Zegar *Timer* %TMi (i=0 do 63) (1)	Słowo	Wartość bieżąca	%TMi.V	nie	1.3-2
		Wartość nastawiona	%TMi.P	tak	
	Bit	Wyjście zegara	%TMi.Q	nie	
Licznik *counter* %Ci (i=0 do 31)	Słowo	Wartość bieżąca	%Ci.V	nie	1.3-3
		Wartość nastawiona	%Ci.P	tak	
	Bit	Licznik pusty (*empty*)	%Ci.E	nie	
		Na wyjściu jest wart. nast.	%Ci.D	nie	
		Licznik zapełniony (*full*)	%Ci.F	nie	
Przerzutnik %MNi (i=0 do 7)	Słowo	Wartość bieżąca	%MNi.V	nie	2.2-1
		Wartość nastawiona	%MNi.P	tak	
	Bit	Wartość bieżąca na wyjściu	%MNi.R	nie	
Rejestr %Ri (i= 0 do 3)	Słowo	Wejście rejestru	%Ri.I	tak	2.2-2
		Wyjście rejestru	%Ri.O	tak	
	Bit	Rejestr zapełniony	%Ri.F	nie	
		Rejestr pusty	%Ri.E	nie	
Bęben %DRi (i=0 do 7)	Słowo	Nr bieżącego kroku	%DRi.S	tak	2.2-3
		Status kroku j	%DRi.Wj	nie	
		Czas aktywności kroku	%DRi.V	nie	
	Bit	Ostatni zdefiniowany krok	%DRi.F	nie	
Zegar serii 7 %T (i=0 do 63) (1)	Słowo	Wartość bieżąca	%Ti.V	nie	2.2-4
		Wartość nastawiona	%Ti.P	tak	
	Bit	Wyjście zegara	%Ti.R	nie	
		Zegar wyliczony	%Ti.D	nie	

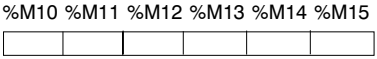
(1) Całkowita liczba zegarów %TMi i %Ti jest ograniczona do 64 dla TSX 37 i 255 dla TSX 57.
(2) Maksymalna liczba dotyczy TSX 37, dla TSX 57 i = 0 - 254 dla wszystkich bloków funkcyjnych.

1.2-6 Obiekty strukturalne

Tablice bitów

Tablice bitów, o określonej długości L, są sekwencjami sąsiadujących ze sobą bitów obiektowych tego samego typu.

Przykład tablicy bitów: %M10:6



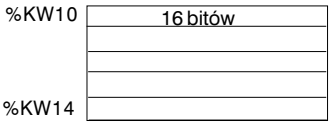
Rodzaj	Adres	Max. długość	Dostęp do zapisu
Bity wejść dyskretnych	%Ix.i:L	1 ≤ L ≤ m (1)	Nie
Bity wyjść dyskretnych	%Qx.i:L	1 ≤ L ≤m (1)	Tak
Bity wewnętrzne	%Mi:L	i+L ≤ n (2)	Tak
Bity *Grafcet-u*	%Xi:L, %Xj.i:L	i+L ≤ n (2)	Nie

(1) m = w zależności od typu modułu (np. 8 dla modułu z 8 wejściami lub 8 wyjściami),
(2) n - zmienia się w zależności od długości zadeklarowanej w konfiguracji.

Tablice słów

Tablica słów, o kreślonej długości L, jest sekwnecją sąsiadujących ze sobą słów tego samego typu.

Przykład tablicy słów: %KW10:5

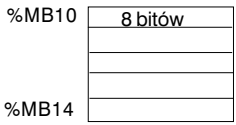


Rodzaj	Format	Adres	Maksymalna długość	Dostęp do zapisu
Słowa wewnętrzne	Pojedyncze Podwójne Zmiennoprzec.	%MWi:L %MDi:L %MFi:L	i+L ≤ Nmax (2) i+L ≤Nmax-1 (2) i+L ≤ Nmax-1 (2)	Tak Tak Tak
Słowa stałe	Pojedyncze Podwójne Zmiennoprzec.	%KWi:L %KDi:L %KFi:L	i+L ≤ Nmax (2) i+L≤ Nmax-1 (2) i+L ≤Nmax-1 (2)	Nie Nie Nie
Słowa *Grafcet-u*	Pojedyncze	%Xi.T:L, %Xj.i.T:L	i+L<Nmax-1 (2)	Nie
Słowa systemowe	Pojedyncze	%SW50:4 (3)		Tak

Łańcuchy znaków

Łańcuch znaków, o określonej długości L, jest sekwencją sąsiadujących ze sobą bajtów tego samego typu.

Przykład łańcucha znaków: %MB10:5



Rodzaj	Adres	Maksymalna długość	Dostęp do zapisu
Słowa wewnętrzne	%MBi:L (5)	1-i+L≤ Nmax (4)	Tak
Słowa stałe	%KBi:L (5)	1-i+L≤ Nmax (4)	Tak

(3) Zestawiać w formie tablicy można tylko słowa %SW50 do %SW53.
(4) *Nmax* = maksymalna liczba zdefiniowana w konfiguracji.
(5) *i* musi być liczbą parzystą.

Obiekty indeksowane

• Adresowanie bezpośrednie

Obiekty indeksowane adresuje się bezpośrednio wtedy, kiedy ich adresy są stałe i zdefiniowane w zapisanym programie.

Przykład: %MW26 (wewnętrzne słowo o adresie 26)

• Adresowanie indeksowe

Przy adresowaniu indeksowym do bezpośredniego adresu obiektu dodaje się indeks. W efekcie zawartość indeksu jest dodawana do adresu obiektu. Indeks definuje się za pomocą słów wewnętrznych %MWi, słów stałych %KWi lub wartości bezpośrednich (*immediate value*). Liczba słów indeksowanych jest nieograniczona.

Przykład : %MW108[%MW2] : bezp. adres słowa 108 + zawartość słowa %MW2. Jeśli zawartością %MW2 jest wartość 12, to zapis %MW108[%MW2] jest równoważny zapisowi %MW120.

Rodzaj	**Format**	**Adres**	**Maksymalny rozmiar**	**Dostęp do zapisu**
Bit wejściowy	Logiczny	%Ii[index]	0≤i+index≤m (1)	Nie
Bit wyjściowy	Logiczny	%Qi[index]	0≤i+index≤m (1)	Tak
Bit wewnętrzny	Logiczny	%Mi[index]	0≤i+index≤Nmax (2)	Tak
Bit *Grafcet-u*	Logiczny	%Xi[index] %Xj.i[index]	0≤i+index≤Nmax (2) 0≤i+index≤Nmax (2)	Nie Nie
Słowa wewnętrzne	Pojedynczy Podwójny Zmiennoprzec.	%MWi[index] %MDi[index] %MFi[index]	0≤i+index≤Nmax (2) 0≤i+index≤Nmax-1 (2) 0≤i+index≤Nmax-1 (2)	Tak Tak Tak
Słowa stałe	Pojedynczy Podwójny Zmiennoprzec.	%KWi[index] %KDi[index] %KFi[index]	0≤i+index≤Nmax (2) 0≤i+index≤Nmax-1 (2) 0≤i+index≤Nmax-1 (2)	Nie Nie Nie
Słowa *Grafcet-u*	Pojedynczy	%Xi .T[index] %Xj.i.T[index]	0≤i+index≤Nmax (2) 0≤i+index≤Nmax (2)	Nie Nie
Tablica słów	<Obiekt> [index]:L	index[index]:L %MDi[index]:L %KWi[index]:L %KDi[index]:L	0≤i+index+L≤Nmax (2) 0≤i+index+L≤Nmax (2) 0≤i+index+L≤Nmax (2) 0≤i+index+L≤Nmax (2)	Tak Tak Nie Nie

(1) *m* = zależy od typu modułu (np. 8 dla modułu z 8 wejściami lub 8 wyjściami). Indeksacja dotyczy tylko dyskretnych modułów I/O.

(2) *Nmax* = maksymalna liczba zdefiniowana w konfiguracji.

Taki sposób adresowania umożliwia przesuwanie się wzdłuż kolejnych elementów sekwencji obiektów tego samego typu (słowa wewnętrzne, słowa stałe). Zawartość indeksu dodawana jest do adresu bezpośredniego

Uwaga: Możliwe jest indeksowanie słów podwójnej precyzji (lub zmiennoprzecinkowych).

Przykład: %MD6[%MW100] bezpośredni adres słowa podwójnego 6 + 2-krotna zawartość słowa %MW100. Jeśli %MW100=10, adresowanym słowem będzie 6 + 2 x 10 -->%MD26.

• **Przekroczenie zakresu indeksu - bit systemowy %S20**

Przekroczenie zakresu indeksu występuje wtedy, kiedy adres indeksowanego obiektu wychodzi poza strefę zawierającą obiekty tego samego typu, co następuje, gdy:
- adres obiektu + wartość indeksu daje wartość mniejszą od 0,
- adres obiektu + wartość indeksu daje wartość większą od zdefiniowanej w konfiguracji wartości maksymalnej (patrz tabela na poprzedniej stronie).

Przy przekroczeniu zakresu indeksu bit systemowy %S20 przyjmuje wartość 1, a obiektowi przypisywany jest indeks o wartości 0.

Monitorowanie przekroczenia zakresu leży w gestii użytkownika - w celu przetworzenia bit %S20 musi być odczytany przez program użytkowy. Kasowanie bitu również jest obowiązkiem użytkownika.

%S20 (wartość początkowa = 0) :
- w przypadku przepełnienia indeksu: system nadaje mu wartość 1,
- potwierdzenie: nadanie bitowi wartości 0 po dokonaniu zmiany indeksu (użytkownik).

1.2-7 Obiekty języka Grafcet

Bity obiektowe

Użytkownik ma do dyspozycji następujące bity obiektowe:
- obiekt %Xi związany z krokami umożliwiający identyfikację statusu kroku *i* diagramu głównego.
- obiekt %XMj związany ze skokami (*macro-step*) umożliwiający identyfikację statusu skoku *j*.
- obiekt %Xj.i związany z krokiem *i* skoku *j* umożliwiający identyfikację statusu kroku *i* skoku *j* diagramu.
- obiekty %Xj.IN oraz %Xj.OUT związane odpowiednio z krokiem wejścia i wyjścia.

Bity te mają wartość 1, gdy krok (skok) jest aktywny, w innym razie mają wartość 0.

Słowa obiektowe

Z każdym krokiem związane jest jedno ze słów: %Xi.T, %Xj.i.T, %Xj.IN.T lub %Xj.OUT.T. Sygnalizuje ono czas, w którym krok *i* diagramu jest aktywny. Co 100ms wartość jest zwiększana o 1, przy czym przyjmuje wartości z przedziału od 0 do 9999.

1.2-8 Obiekty związane z blokiem typu DFB

Dostęp do parametrów wyjściowych oraz zmiennych globalnych (*public*) bloku typu DFB odbywa się za pośrednictwem programu. Mogą one być obiektami logicznymi, numerycznymi lub tablicami obiektów (patrz rozdział 6).
Obiekty te mają charakter czysto symboliczny i definiuje się je przy użyciu następującej składni: *DFB_nazwa.Parametr_nazwa*, gdzie *DFB_nazwa* oznacza nazwę użytego bloku funkcyjnego (max 32 znaki), a *Parametr_nazwa* jest nazwą parametru wyjściowego lub zmienną globalną (max 8 znaków).

1.2-9 Przypisywanie symboli

Symbole

Symbol to szereg maksymalnie 32 znaków alfanumerycznych, z których pierwszy jest literą alfabetu. Symbol zaczyna się od wielkiej litery, a pozostałą część symbolu zapisuje się literami małymi (np. Kociol_1). Po zdefiniowaniu symbolu można go później zapisywać zarówno wielkimi jak i małymi literami (np. KOCIOL_1). Program automatycznie przetworzy ten zapis na poprawny.

Dopuszczalne jest użycie następujących znaków:
- wielkich liter alfabetu:
  "A do Z" oraz "ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏDÑÒÓÔÕÖØÙÚÛÜYp"
  lub małych liter alfabetu:
  " a do z" oraz "áàâãåæçéèêëÌÍÎÏðñòóôõöøùúûüypßÿ"
- numerycznych: cyfry 0 to 9 (nie mogą być pierwszym znakiem symbolu).
- znaku "_" (nie może być ani pierwszym, ani ostatnim znakiem symbolu).

Istnieje pewna liczba słów, które są zarezerwowane i nie można ich używać jako symbolu. Pełną listę zamieszczono w rozdziale 5, w części B.

Symbole definiuje się i wiąże z obiektami za pomocą edytora zmiennych (patrz rozdział 5, część D). Do każdego symbolu można dołączyć komentarz, który może się składać maksymalnie z 508 znaków. Symbole wraz z komentarzami są zapisywane na dysku terminala, a nie w sterowniku.

Obiekty, którym można przypisywać symbole

Symbole można przypisywać niemal wszystkim obiektom języka PL7, z wyjątkiem obiektów indeksowych oraz obiektów strukturalnych typu tablice z tym, że gdy obiekt bazowy lub indeks mają przypisane symbole, to ten symbol jest wykorzystywany w obiekcie strukturalnym.

Przykłady:
- jeśli słowu %MW0 przypisano symbol "Temperatura", to tablica słów %MW0:12 ma przypisany symbol "Temperatura:12"
- jeśli słowu %MW10 przypisano symbol "Piec_1", to słowo indeksowe %MW0[%MW10] ma przypisany symbol "Temperatura[piec_1]".

Bitom obiektowym wydzielonym ze słów oraz bitom i słowom bloków funkcyjnych można nadać symbol przy czym jeśli go nie mają, to przyjmą symbol obiektu bazowego.
Przykłady:
- jeśli słowu %MW0 przypisano symbol "Stan_pompy" i jeśli wydzielony ze słowa bit %MW0:X1 nie ma symbolu, to przejmie on symbol słowa. Stąd symbolem bitu %MW0:X1 staje się "Stan_pompy:X1".
- jeśli blokowi funkcyjnemu %TM0 przypisano symbol "Piec1_zegar", a wyjście %TM0.D nie ma przypisanego symbolu, to przejmuje ono symbol bloku. Stąd symbolem wyjścia %TM0.D staje się "Piec_zegar.D".

Obiekty o charakterze czysto symbolicznym

Parametry bloków typu DFB są dostępne tylko jako symbole. Obiekty te są definiowane przy użyciu następującej składni:

DFB_nazwa.Parametr_nazwa.

gdzie **DFB_nazwa** jest nazwą bloku funkcyjnego DFB (maksymalnie 32 znaki), a **Parametr_nazwa** jest nazwą nadaną parametrowi wyjściowemu bloku, względnie zmienną globalną (maksymalnie 8 znaków).

Obiekty o symbolach przypisywanych automatycznie

Niektóre moduły umożliwiają automatyczne przypisywanie symboli obiektom z nimi związanym. Użytkownik podaje ogólny symbol kanału danego modułu %CHxy.i, a symbole obiektów związanych z tym kanałem mogą być automatycznie generowane.

Obiektom tym przypisuje się symbole w następujący sposób:

Przedrostek_użytkownika_Przyrostek_fabryczny

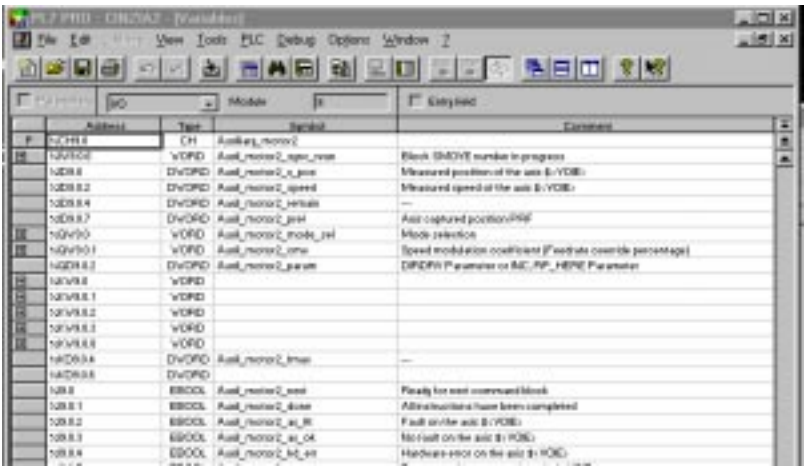
gdzie

Przedrostek_użytkownika jest symbolem ogólnym przypisanym kanałowi %CHxy.i (maksymalnie 12 znaków),

Przyrostek_fabryczny jest częścią symbolu nadawana przez system odpowiadającą bitowi lub słowu kanału (maksymalnie 20 znaków).

Oprócz symbolu generowany jest również automatycznie komentarz producenta zawierający krótką charakterystykę obiektu.

Przykład: *Auxil_motor2_param* (parametry silnika pomocniczego nr 2), gdzie *Auxil_motor2* stanowi przedrostek nadany przez użytkownika, *_param* jest przyrostkiem nadanym fabrycznie.



1.3 Pamięć użytkowa

1.3-1 Wprowadzenie

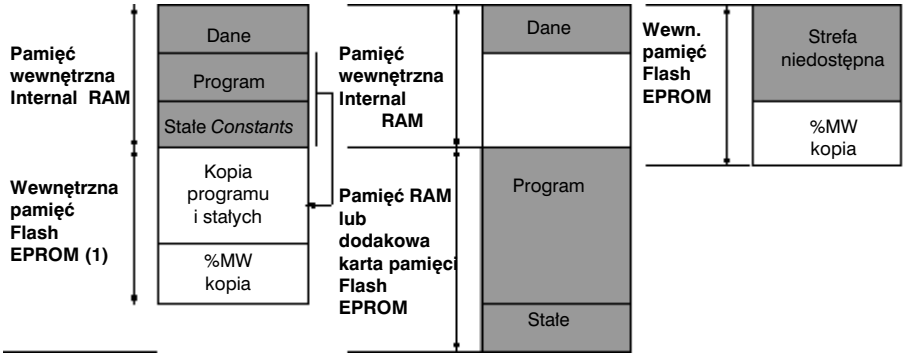
Dostępną dla użytkownika pamięć sterowników **TSX 37**, podzielono na dwie grupy:

- **Pamięć bitową (*bit memory*)**
  Pamięć RAM znajdująca się w module procesora. Pojemność 1280 bitów obiektowych.
- **Pamięć słowną (*word memory*)**
  Słowa 16-bitowe (program, dane i stałe) są przechowywane w pamięci RAM modułu procesora. Pamięć tę można zwiększyć o dodatkową pamięć RAM o pojemności 32 lub 64Ksłów lub przy użyciu karty FLASH EPROM (pamięć dla TSX 37-21/22). Pamięć FLASH EPROM o pojemności 16Ksłów zintegrowana z modułem procesora może być wykorzystywana do jednoczesnego przechowywania programu aplikacyjnego (15 Ksłów) i 1000 słów wewnętrznych (1 Ksłów) (rozdział 1.3-2).

  Karta pamięci FLASH EPROM o pojemności 32Ksłów może być wykorzystywania do wykonywania kopii (*backup*) albo do uaktualniania aplikacji zapisanej w wewnętrznej pamięci RAM procesora. Na karcie takiej zapisywana jest część programu i stałe, ale nie dane.

  Pamięć słowna może być usystematyzowana na dwa różne sposoby (w zależności od tego czy wykorzystuje się kartę PCMCIA, czy też nie):

TSX 37-10 lub
TSX 37-21/22 (bez karty PCMCIA) **TSX 37-21/22 (z kartą PCMCIA)**



Dane : dane dynamiczne systemu i aplikacji.
Program : deskryptory i wykonywalne kody zadań.
Stałe : słowa stałe, wartości początkowe (*initial values*), konfiguracja I/O.

Uwaga

Do podtrzymania zasilania pamięci można używać baterii kadmowo-niklowych zainstalowanych w module procesora dla pamięci bitowej i wewnętrznej (RAM)

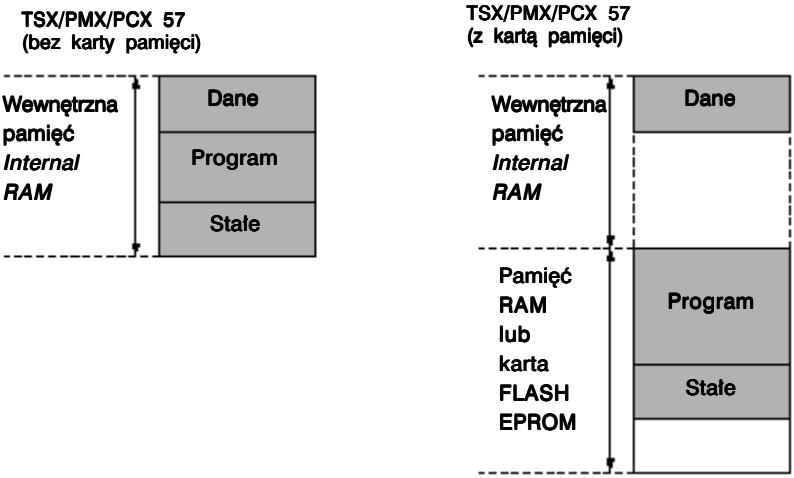
(1) W przypadku utraty aplikacji zapisanej w pamięci RAM (awaria zasilania lub brak baterii) następuje automatyczny transfer aplikacji z pamięci FLASH EPROM do pamięci RAM. Przy pomocy programowalnego terminala można dokonać takiego transferu ręcznie.

Pamięć w sterownikach **TSX/PMX/PCX 57** nie jest podzielona. Pamięć bitowa, występująca oddzielnie w przypadku sterowników TSX 37, wchodzi w skład pamięci słownej (bity zapamiętywane są w strefie danych). Jej pojemność to 4096 bitów.

- **Pamięć słowna:**
  Jest to pamięć RAM znajdująca się w module prcesora, służąca do przechowywania 16-bitowych słów (program, dane i stałe). Można ją zwiększyć poprzez zastosowanie dodatkowej pamięci RAM o pojemności 32, 64, 128 lub 256Kstów lub karty pamięci FLASH EPROM (1).

  Karta FLASH EPROM o pojemności 32Kstów może być wykorzystana do wykonania kopii (*backup*) albo do uaktualnienia aplikacji zapisanej w wewnętrznej pamięci RAM procesora. Na karcie zapisywana jest część programu i stałe (nie dane).

  Pamięć słowna może być usystematyzowana na dwa różne sposoby (w zależności od tego czy wykorzystuje się kartę PCMCIA, czy też nie):



Data : dynamiczne dane aplikacji i systemu (system rezerwuje sobie obszar o pojemności minimum 5Kstów: patrz część B, rozdział 8).
Program : deskryptory i wykonywalne kody zadań.
Stałe : słowa stałe, wartości początkowe (*initial values*), konfiguracja I/O.

Nie ma możliwości przepełnienia danych na karcie pamięci jak i jednoczesnej obecności programu w wewnętrznej pamięci RAM i na karcie.

Uwaga

Do podtrzymania zasilania pamięci RAM mogą służyć baterie niklowo-kadmowe.

(1) Karty pamięci o pojemności 256Kstów są stronicowane. Na jednej stronie 128Kstów zapisywane są kody wykonywalne, a na drugiej - informacje graficzne.
Więcej informacji należy szukać w rozdziale 8 części B.

1.3-2 Zapamiętywanie / odczytywanie słów wewnęrznych %MWi

Zapamiętywanie słów wewnętrznych %MWi
Sterowniki TSX 37 mają możliwość kopiowania do 1000 słów wewnętrznych (%MW) do wewnętrznej pamięci Flash EPROM. Dzięki temu w przypadku awarii zasilania, gdy bateria podtrzymująca jest niesprawna (lub jej w ogóle nie ma), następuje automatyczne przekopiowanie danych regulacyjnych (*adjustment data*) do pamięci Flash EPROM. Aby możliwe było zapisanie słów wewnętrznych do pamięci Flash EPROM, **aplikacja musi być zatrzymana** (*stopped*). Dokonuje się tego, w zależności od konfiguracji, w następujący sposób:

- wystawienie na wejściu dyskretnym %I1.9 wartości 1, lub
- z poziomu panela regulacyjnego (*adjustment panel*), poprzez nadanie bitowi 0 słowa %SW96 wartości 1.

W słowie %SW97 zdefiniowana jest liczba słów %MWi do zapisania (maks. 1000).

Po zakończeniu operacji kopiowania (*backup*) na wyświetlaczu pojawia się komunikat OK lub NOK (w zależności od wyniku operacji).

Słowa wewnętrzne %MWi są zawsze zapisywane podczas zapisywania programu.

Jeżeli słowo systemowe %SW97 ma wartość 0, to do pamięci Flash EPROM kopiowany jest tylko program rezydujący w wewnętrznej pamięci RAM (jest to równoważne z kopiowaniem programu).
Ostrzeżenie: Wszystkie zapisane dotychczas słowa %MWi zostaną skasowane.

Odzyskiwanie słów wewnętrznych %MWi
W przypadku **zimnego startu** (*cold start*) zapamiętane słowa %MWi przepisywane są z wewnętrznej pamięci Flash EPROM do pamięci RAM. Taki start może być spowodowany:

- utratą zawartości wewnętrznej pamięci RAM. W takim przypadku, jeśli dostępna jest kopia programu aplikacyjnego, to jest ona również przepisywana do wewnętrznej pamięci RAM (dla TSX 37-10 i TSX 37-20 nie wyposażonych w kartę PCMCIA),
- naciśnięciem przycisku RESET na płycie czołowej sterownika,
- nadaniem bitowi %S0 wartości 1, podczas pracy w trybie regulacji (*adjusting*),
- kliknięciem przycisku "Cold start" w oknie testowania (*debug*) procesora,
- transferem programu do sterownika (za pomocą portu terminala, sieci FIPWAY, itd),
- włożeniem karty PCMCIA.

Aby przy "zimnym starcie" nastąpiło odzyskanie zapisanych w pamięci wewnętrznej słów %MW, należy wyłączyć opcję "Reset %MWi on cold start" (co oznacza: "przy zimnym starcie przywróć domyślne wartości słowom %MWi") w oknie konfiguracji procesora.
Więcej szczegółowych informacji należy szukać w części A instrukcji instalacji sterowników TSX Micro.

1.3-3 Pamięć bitowa

Charakterystyka

W przypadku sterowników TSX 37 pamięć ta może pomieścić do 1280 bitów obiektowych. Natomiast dla sterowników TSX/PMX/PCX 57 pamięć bitowa nie jest wydzielona, a jej zawartość znajduje się w pamięci słownej w strefie danych aplikacji.

		TSX 37-10	TSX 37-21/22	TSX/PMX PCX 57-1•	TSX/PMX 57-2•	TSX/PMX PCX 57-3•
Bity systemowe	**%SI**	128	128	128	128	128
Bity I/O	**%I/Qx**	(1)	(1)	(1)	(1)	(1)
Bity wewnętrzne	**%Mi**	256	256	3962 (2)	8056 (2)	12152 (2)
Bity kroków	**%Xi** (3)	96	128	1024	1024	1024

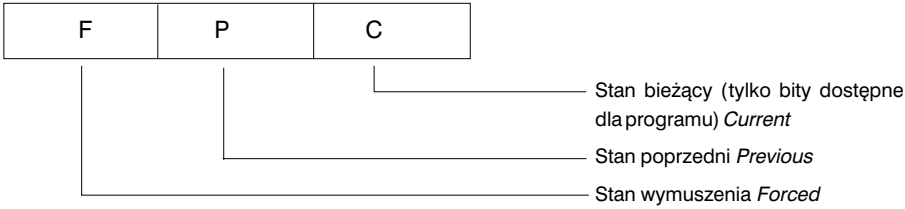
(1) Zależy od zadeklarowanej konfiguracji sprzętowej (typ modułu I/O, rodzaj urządzeń na szynie AS-i i na szynie FIPIO)

(2) W konfiguracji można nadawać wartości wybranym bitom wewnętrznym. Domyślna wartość bitu (256 do 2048) zmienia się w zależności od zastosowanego procesora oraz od tego, czy jest stosowana karta pamięci. Pozostała część pamięci jest do dyspozycji funkcji aplikacji.

(3) Dla sterowników TSX/PMX/PCX 57 jest to całkowita liczba bitów kroku i bitów skoku.

Struktura

Każdy bit obiektowy jest zapisywany w pamięci bitowej za pośrednictwem trzech przypisanych mu bitów:



Podczas uaktualniania pamięci bitowej system wykonuje następujące operacje:

- zapisanie stanu bieżącego jako stanu poprzedniego,
- uaktualnienie bieżącego stanu bitu przez program, system lub terminal (gdy następuje wymuszenie wartości bitu).

Zbocze opadające lub narastające sygnału

Taka struktura pamięci bitowej jest wykorzystywana do wykrywania zbocza opadającego lub narastającego:

- bitów I/O,
- bitów wewnętrznych.

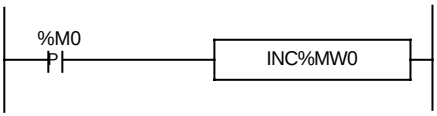
Zalecenia dotyczące wykorzystania zbocza sygnału

Przełączanie styku na zboczach sygnału działa poprawnie przy zachowaniu następujących zasad:

- w odniesieniu do pojedynczych obiektów - we wszystkich przypadkach:
  - bit wejściowy: reakcja na zbocze w zadaniu, do którego jest przypisany,
  - bit wyjściowy lub wewnętrzny: odczyt i zapis w obrębie **tego samego zadania**.

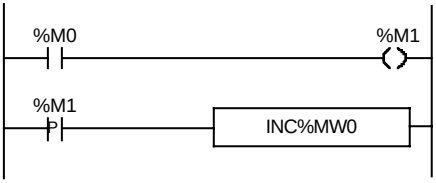
- Cewki stosować **tylko wtedy**, kiedy styk reagujący na zbocze jest wykorzystany w programie.
- Podczas kontroli zbocza sygnału nie wolno wydawać poleceń SET (ustaw) oraz RESET (skasuj do stanu początkowego), bo nawet w przypadku, gdy wynikiem równania warunkującego wykonanie polecenia SET/RESET jest 0, to choć operacja nie zostanie wykonana, nastąpi uaktualniene rejestru operacji obiektu (*object log*) czego efektem będzie utrata (zignorowanie) zbocza.
- Nie wolno testować wejść i wyjść (I/O) wykorzystanych w zadaniu wyzwalanym zdarzeniami, w zadaniu głównym (MASTER) i szybkim (FAST).
- W przypadku bitów wewnętrznych: testowanie zbocza jest niezależne w stosunku do "przebiegu" zadania (*task scan*). Zbocze bitu wewnętrznego %Mi jest wykrywane kiedy sygnał zmienia stan pomiędzy 2 operacjami odczytu. **Wynik testowania zbocza pozostaje tak długo, jak długo ten wewnętrzny bit nie zostanie poddany ponownemu przetworzeniu.**

W przykładzie obok, jeśli bit %M0 ma wymuszony stan 1 (w tablicy animacji), to układ zachowuje się jakby zbocze sygnału było permanentnie otwarte.



Aby zbocze było wykrywane jednorazowo należy zastosować dodatkowy, pośredni bit wewnętrzny.

W takim przypadku uaktualniany jest bit %M1 stąd też zbocze zostanie wykryte tylko raz.



Wymuszanie stanu bitu

Jeśli z terminala przychodzi żądanie (*request*) wymuszenia wartości bitu, to:

- informacja o wymuszeniu bitu *F* przyjmuje wartość 1
- bieżący stan bitu *C* przyjmuje następującą wartość:
  - 1 w przypadku wymuszenia stanu 1,
  - 0 w przypadku wymuszenia stanu 0.

Stany te pozostają nie zmienione dopóty, dopóki:

- wymuszenie zostanie zdjęte, a stan bitu uaktualniony,
- nastąpi zmiana wymuszonej wartości, w którym to przypadku nastąpi tylko zmiana wartości bieżącej bitu *C*.

1.3-4 Pamięć słowna

Jest to pamięć do przechowywania słów 16-bitowych, podzielona na 3 obszary:

- Dane
- Program
- Stałe

Dane aplikacji
Program aplikacji
Stałe aplikacji

Wielkość poszczególnych obszarów zdefiniowana jest w konfiguracji.

Pamięć danych aplikacji

Pamięć danych obejmuje następujące strefy:

- **Słowa systemowe**: stała liczba.
- **Bloki funkcyjne**: rozmiar zależy od liczby słów oraz wejść i wyjść tych bloków (wartości bieżące, parametry regulacyjne, itp.).
  Liczba bloków funkcyjnych danego typu jest stała i ustalona w konfiguracji.
- **Słowa wewnętrzne**: rozmiar zależy od liczby słów zadeklarowanej w konfiguracji.
- **Liczba I/O**: odpowiada liczbie słów przypisanych każdemu modułowi.
- **Słowa wspólne sieci**: 4 słowa wspólne (*common words*) na 1 stację (pamięć dostępna tylko gdy zamontowano moduł komunikacyjny i został on skonfigurowany na wymianę słów wspólnych).

W przypadku TSX/PMX/PCX 57 pamięć danych zawiera bity danych opisanych w poprzednim paragrafie.

Pamięć programu aplikacyjnego

Ta strefa pamięci zawiera kody wykonywalne zamieszczone w programie, dane graficzne (labelki języka *Ladder*) i komentarze do programu.

Pamięć stałych

W tym obszarze przechowywane są parametry bloków funkcyjnych oraz modułów I/O zdefiniowanych w konfiguracji i słowa stałe %KW.

Uwaga:
Symbole i komentarze powiązane z obiektami nie są rejestrowane w pamięci sterownika lecz są zapisywane w lokalnej aplikacji (na twardym dysku terminala).

1.3-5 Sterowniki TSX 37-10/21/22

Wielkość pamięci bitowej

Procesor		**37 10**	**37 21/22**
Pamięć dostępna w procesorze		1280	1280
Rodzaj obiektu	bity systemowe %Si	128	128
	bity I/O %I/Qx.i	(1)	(1)
	bity wewnętrzne %Mi	256	256
	bity kroków %Xi	96	128

(1) zależy od zdefiniowanej konfiguracji sprzętowej (moduły I/O, urządzenia w magistrali AS-i)

Wielkość pamięci słownej

Procesor	**3710**	**3721**			**37 22**		
Karta pamięci	-	-	32Kstów	64Kstów	-	64Kstów	64Kstów
Rozmiar całk.	14Kstów	20Kstów	52Kstów	84Kstów	20Kstów	52Kstów	84Kstów
Dane (%MWi)	0.5Kstów(1)	0.5Kstów(1)	17.5Kstów	17.5Kstów	0.5Kstów(1)	17.5Kstów	17.5Kstów
Program 100% logiczny							
• Język LD	3.8 Kinstr.	6.6 Kinstr.	13.7 Kinstr.	28.5 Kinstr.	6.3 Kinstr.	13.6 Kinstr.	28.4 Kinstr.
• Język IL	4.9 Kinstr.	8.4 Kinstr.	17.5 Kinstr.	36.3 Kinstr.	8.1 Kinstr.	17.3 Kinstr.	36.1 Kinstr.
• Język ST	3.3 Kinstr.	5.6 Kinstr.	11.7 Kinstr.	24.2 Kinstr.	5.4 Kinstr.	11.5 Kinstr.	24.1 Kinstr.
Program 90% logiczny							
• Język LD	3.1 Kinstr.	5.4 Kinstr.	11.8 Kinstr.	24.7 Kinstr.	5.2 Kinstr.	11.6 Kinstr.	24.5 Kinstr.
• Język IL	3.8 Kinstr.	6.6 Kinstr.	14.3 Kinstr.	30.0 Kinstr.	6.3 Kinstr.	14.2 Kinstr.	29.8 Kinstr.
• Język ST	2.9 Kinstr.	5.1 Kinstr.	11.1 Kinstr.	23.3 Kinstr.	4.9 Kinstr.	11.0 Kinstr.	23.2 Kinstr.
Program 65% logiczny							
• Język LD	2.2 Kinstr.	4.0Kinstr.	9.1 Kinstr.	18.9 Kinstr	3.9 Kinstr.	8.9 Kinstr.	18.8 Kinstr.
• Język IL	2.5 Kinstr.	4.6Kinstr.	10.3 Kinstr.	21.3 Kinstr.	4.4 Kinstr.	10.1 Kinstr.	21.2 Kinstr.
• Język ST	2.5 Kinstr.	4.6Kinstr.	10.3 Kinstr.	21.3 Kinstr.	4.4 Kinstr.	10.1 Kinstr.	21.2 Kinstr.
Stałe (1)	128 słów	128 słów	256 słów	512 słów	128 słów	256 słów	512 słów

(1) Domyślny rozmiar można powiększać, jednak trzeba pamiętać o wpływie jaki będzie to miało na rozmiar programu aplikacyjnego.

Uwaga: W języku PL7 do mapowania pamięci (czyli zarządzania pamięcią) aplikacji służy polecenie **PLC**/**Memory Usage**.

1.3-6 Sterowniki TSX/PCX/PMX 57-10/20/25

Poniższe tabele dotyczą sterowników TSX 57-10, PCX 57-10, PMX 57-10, TSX 57-20, PMX 57-20 i TSX 57-25.

Wielkość pamięci bitowej

Procesor		57 10	57 20/25
Rodzaj	bity systemowe %Si	128	128
obiektu	bity I/O %I/Qx.i	(1)	(1)
	bity wewnętrzne %Mi	3962	8056
	bity kroków %Xi	1024	1024

(1) zależy od zdefiniowanej konfiguracji sprzętowej (moduły I/O, urządzenia na szynie AS-i oraz na szynie FIPIO)

Wielkość pamięci słownej (w ilości Kstów)

Procesor	TSX-PCX57 10 / PMX 57 10			TSX-PMX 5720 / TSX 57 25			
Karta pamięci	-	32K	64K	-	32K	64K	128K
Pamięć wewn.	32K/48K	32K/48K	32K/48K	48K/64K	48K/64K	48K/64K	48K/64K
Dane (%MWi)	1K(1)	26K	26K	1K(1)	30.5K	30.5K	30.5K
Program 100% logiczny							
• Język LD	8.8/16.1K	12.4/12.4K	27.2/27.2K	16.1/23.5K	12.4/12.4K	27.2/27.0K	56.8/56.8K
• Język IL	11.2/20.6K	15.8/15.8K	34.7 /34.7K	20.5/29.9K	15.8/15.8K	34.7/34.4K	72.4/72.4K
• Język ST	13.7 /13.7K	10.6 /10.6K	23.1/23.1K	13.7/19.9K	10.5/10.5K	23.1/23.0K	48.2/48.2K
Program 90% logiczny							
• Język LD	5.3/11.6K	8.8/8.8K	21.6/21.6K	11.6/17.6K	8.7/8.7K	21.6/21.6K	47.7/47.5K
• Język IL	6.3/14.2K	10.5/10.7K	25.9/26.3K	13.9/21.1K	10.4/10.4K	25.8/25.8K	56.6/56.7K
• Język ST	5.0/11.0K	8.4/8.3K	20.7/20.4K	11.1/16.9K	8.4/8.4K	20.7/20.7K	45.4/45.4K
Program 65% logiczny							
• Język LD	3.7/8.5K	6.8/6.6K	16.9/16.5K	8.7/13.7K	6.8/6.8K	16.8/16.8K	37.0/37.0K
• Język IL	3.8/9.7K	6.9/7.5K	17.3/18.7K	8.9/14.1K	6.9/6.9K	17.2/17.2K	37.9/37.9K
• Język ST	4.4/9.7K	8.0/7.5K	20.0/18.7K	10.3/16.3K	8.0/8.0K	19.9/19.9K	43.8/43.8K
Stałe (1)	128słów	128słów	256słów	512słów	128słów	256słów	512słów

(1) Domyślny rozmiar można zmieniać, lecz ma to wpływ na wielkość programu aplikacyjnego.

Uwaga:

- Tam, gdzie w tabeli podano 2 wartości rozdzielone "/" oznacza, że dla 2 rodzajów procesora z nagłówka tabeli są one różne.
- W języku PL7 do mapowania pamięci aplikacji (czyli zarządzania pamięcią) służy polecenie **PLC/Memory Usage**.

1.3-7 Sterowniki TSX/PCX 57-30/35

Poniższe tabele dotyczą sterowników TSX 57-30, TSX 57-35 i PCX 57-35.

Wielkość pamięci bitowej

Procesor		5730/35
Rodzaj obiektu	bity systemowe %Si	128
	bity I/O %I/Qx.i	(1)
	bity wewnętrzne %Mi	12152
	bity kroków %Xi	1024

(1) zależy od zdefiniowanej konfiguracji sprzętowej (moduły I/O, urządzenia na szynie AS-i oraz na szynie FIPIO)

Wielkość pamięci słownej (w ilości Ksłów)

Procesor	TSX/PMX 5730 / TSX 57 35				
Karta pamięci	-	32K	64K	128K	256K
Pamięć wewn.	64K / 80 K	64K / 80 K	64K / 80K	64K / 80K	64K / 80K
Dane (%MWi)	1K(1)	30.5 K	30.5K	30.5K	30.5K
Program 100% logiczny					
• Język LD	23.5/30.8 K	12.4 K/12.4 K	27.2 K/27.2 K	56.8 K/56.8 K	90.5 K/90.5 K
• Język IL	29.9/39.3 K	15.8 K/15.8 K	34.7K/34.7K	72.4 K/72.4 K	83.6 K/83.6 K
• Język ST	19.9/26.2 K	10.5 K/10.5 K	23.1 K/23.1 K	48.2 K/48.2 K	74.6 K/74.6 K
Program 90% logiczny					
• Język LD	18.0/24.4 K	8.7 K/8.7 K	21.6 K/21.6 K	47.5 K/47.5 K	76.8 K/76.8 K
• Język IL	21.5/29.2 K	10.4 K/10.4 K	25.8 K/25.8 K	56.7 K/56.7 K	73.9 K/73.9 K
• Język ST	17.2/23.4 K	8.4 K/8.4 K	20.7 K/20.7 K	45.49K/45.49K	76.0 K/76.0 K
Program 65% logiczny					
• Język LD	13.7/18.8 K	6.8 K/6.8 K	16.8 K/16.8 K	37.0 K/37.0 K	63.1 K/63.1 K
• Język IL	14.1/19.2 K	6.9 K/6.9 K	17.2 K/17.2 K	37.9 K/37.9 K	61.4 K/61.4 K
• język ST	16.3/22.2 K	8.0 K/8.0 K	19.9 K/19.9 K	43.8 K/43.8 K	63.8 K/63.8 K
Stałe (1)	256 słów	256 słów	256 słów	1024 słów	1024 słów

(1) Domyślny rozmiar można zmieniać, lecz ma to wpływ na wielkość programu aplikacyjnego.

Uwaga:

- Tam, gdzie w tabeli podano 2 wartości rozdzielone "/" oznacza, że dla 2 rodzajów procesora z nagłówka tabeli są one różne.
- W języku PL7 do mapowania pamięci (czyli zarządzania pamięcią) aplikacji służy polecenie **PLC/ Memory Usage**.

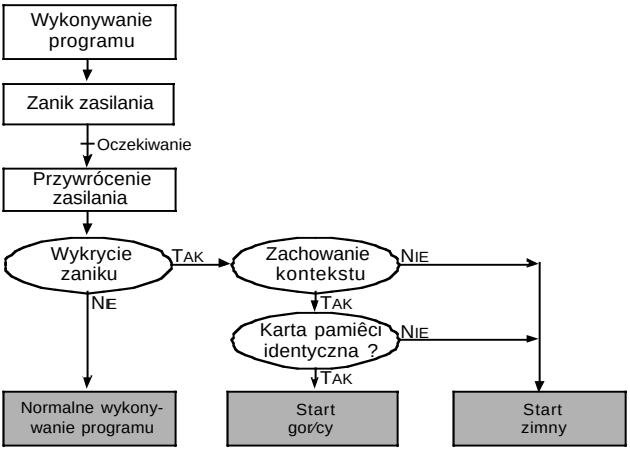
1.4 Tryby pracy

1.4-1 Reakcja sterownika na zanik zasilania i jego przywrócenie

W przypadku zaniku zasilania system zachowuje kontekst aplikacji oraz rejestruje czas wystąpienia zaniku i wystawia na wszystkich wyjściach wartości trybu zastępczego (czyli taki stan na wyjściach jaki został zdefiniowany w konfiguracji).
Po przywróceniu zasilania zachowany kontekst aplikacji porównywany jest z bieżącym i w zależności od wyniku realizowana jest odpowiednia procedura startu systemu:

- jeśli kontekst się zmienił (utrata kontekstu lub załadowanie nowej aplikacji), to sterownik inicjuje aplikację, co oznacza zimny start,
- jeżeli porównywane konteksty są identyczne, to sterownik restartuje system bez inicjacji danych, co oznacza start gorący.

Jeżeli czas zaniku zasilania jest krótszy niż czas rozładowania filtra wejściowego (czyli około 10ms dla zasilacza prądu przemiennego oraz około 1ms dla zasilacza prądu stałego), to nie jest on wychwytywany przez program (jest on normalnie wykonywany).



Uwaga:

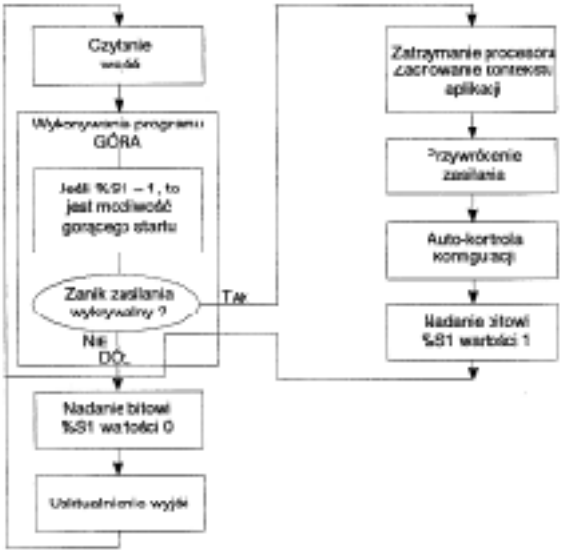
Zimny start realizowany jest w następujących przypadkach:

- Przywrócenie zasilania, gdy system zgubi kontekst (np. bateria podtrzymania jest niesprawna).
- Podczas pierwszego uruchomienia aplikacji.
- Po naciśnięciu przycisku RESET na module procesora.
- Po nadaniu bitowi systemowemu %S0 wartości 1.
- Przy inicjacji z poziomu terminala.
- Po włożeniu karty pamięci PCMCIA lub po wykonaniu manipulacji przy slocie karty (z wyjątkiem sterownika PCX 57, gdzie nie wolno wkładać karty w trakcie pracy systemu).

Gorący start realizowany jest w następujących przypadkach:

- Przywrócenie zasilania bez utraty kontekstu aplikacji.
- Po nadaniu bitowi systemowemu %S1 wartości 1.
- Programowo, z poziomu terminala.

1.4-2 Operacje wykonywane podczas gorącego startu systemu

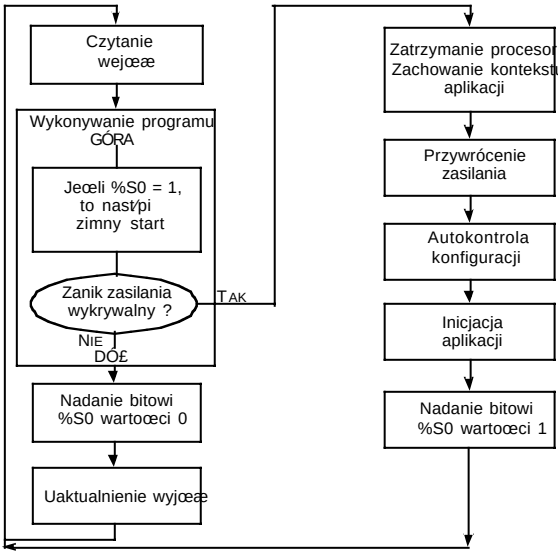


Przywrócenie wykonywania programu
Restart programu odbywa się od miejsca, w którym nastąpił zanik zasilania, przy czym wyjścia nie są aktualizowane. System ponownie rozpoczyna cykl, w którym od nowa uwzględniane są stany wszystkich wejść, ponownie uaktywnia zadanie główne MASTER nadając bitowi %S1 wartość 1 podczas jednego "przejścia" zadania (*scan of the task*) i aktualizuje stany wyjść.
System wyklucza, na czas pierwszego "przejścia" zadania głównego MASTER, zadanie szybkie oraz zadania wyzwalane zdarzeniami.

Przetwarzanie informacji o gorącym starcie
Jeżeli użytkownik chce, by w przypadku gorącego startu wykonywane były jakieś specjalne operacje w odniesieniu do aplikacji, to musi napisać odpowiedni program realizujący te operacje i testujący, na początku programu zadania MASTER, stan bitu %S1 (na wypadek, gdy przyjmie on stan 1).

Zmiana wartości na wyjściach
- Przez cały czas trwania zaniku zasilania wyjścia przyjmują stany trybu zastępczego, tzn. albo zachowują wartości bieżące, albo przyjmują wartości zastępcze (reakcja zależy od wyboru dokonanego w konfiguracji).
- Po przywróceniu zasilania wyjście utrzymuje stan 0 dopóty, dopóki nie zostanie ono zaktualizowane przez zadanie.

1.4-3 Operacje wykonywane podczas zimnego startu



Inicjacja systemu i jego danych, co odpowiada:

- Skasowaniu bitów, obrazu wejść i wyjść oraz słów wewnętrznych (jeżeli w oknie konfiguracji sterownika zaznaczona jest opcja kasowania słów wewnętrznych w przypadku zimnego startu - patrz rozdział 1.3, część D). Jeżeli nie zaznaczono opcji kasowania słów %MW, a są one zapisane w pamięci, to są przywracane.
- Inicjacji bitów i słów systemowych.
- Inicjacji bloków funkcyjnych bazujących na danych konfiguracyjnych.
- Wykluczeniu z realizacji do końca pierwszego "przebiegu" zadania głównego MASTER, wszystkich innych zadań
- Przejście na diagramie programu *Grafcet* do kroków inicjujących (*initial steps*).

Przetwarzanie informacji o zimnym starcie

W celu przypisania wykonania określonych operacji w przypadku wystąpienia zimnego startu, należy testować stan bitu %S10:X0 (gdy %S10:X0 = 1 to znaczy, że realizowany jest zimny start). Przy zimnym starcie, sterownik **kontynuuje wykonywanie operacji, lub nie**, w zależności od wyboru dokonanego w konfiguracji (parametr RUN AUTO).

Zmiana stanów wyjść

- Przy zaniku zasilania wyjścia przyjmują stany trybu zastępczego, tzn. albo zachowują wartości bieżące, albo przyjmują wartości zastępcze (zależnie od konfiguracji).
- Po przywróceniu zasilania wyjścia utrzymują stan 0 dopóty, dopóki nie zostaną zaktualizowane przez zadanie.

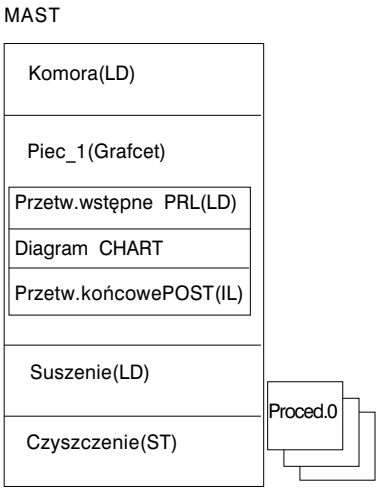
1.5 Struktura jednozadaniowa

1.5-1 Zadanie główne MASTER

W przypadku jednozadaniowej struktury aplikację tworzy jedno zadanie - zadanie główne MAST.
Program zadania głównego tworzy trzon programu obejmujący główne operacje wykonywane w kilku sekcjach i procedury.

- Program główny (MAIN) podzielony jest na sekcje. Każda sekcja programowana jest oddzielnie, w języku dla niej najbardziej odpowiednim (LD, IL, ST lub Grafcet).
- Procedury SRi (i=0 do 253)
  Procedury są również programowane niezależnie w języku LD, IL lub ST, przy czym mogą być one wywoływane zarówno z programu głównego, jak i z innych procedur (można zagnieździć do 8 procedur).

W trakcie konfiguracji określa się sposób wykonywania zadania głównego (cykliczne lub okresowe).



Sekcje

Sekcje charakteryzują:

- nazwa o maksymalnie 24 znakach,
- język, w którym jest programowana,
- zadanie, do którego należy,
- warunek wykonania (opcjonalnie). Sekcja jest aktywna gdy warunek = 1 i nie jest aktywna, gdy warunek = 0.

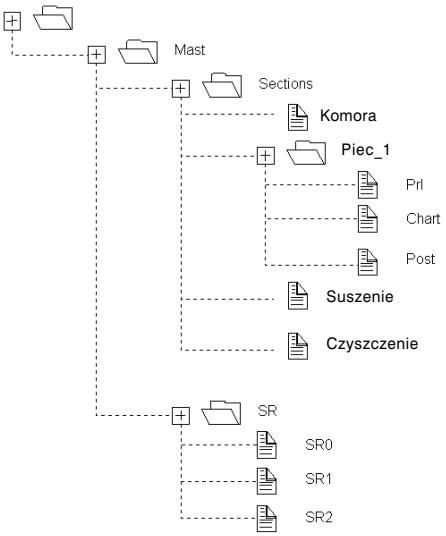
  Jako warunki można wykorzystywać następujące obiekty:
  Bity %M,%S,%X, bity indeksowane, bity wydzielone ze słów, bity %I , %Q. Wartości wszystkich bitów (z wyjątkiem %S, bitów indeksowanych, bitów wydzielonych ze słów oraz %I xy.i.ERR i %I xy.MOD.ERR.) można wymuszać za pośrednictwem terminala.
- komentarz o maksymalnej długości 250 znaków.

Uwaga: przy starcie zimnym warunki wykonania przyjmują wartości 0, co powoduje, że wszystkie sekcje, których wykonanie jest warunkowe, są pomijane.

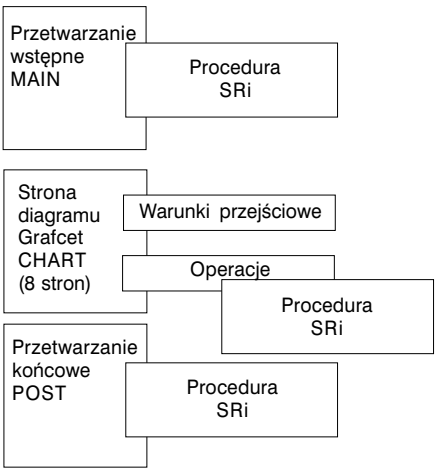
Sekcja jest elementem autonomicznym programu. Etykiety identyfikacyjne linii, instrukcji oraz labelek działają w obrębie sekcji (nie ma możliwości programowego skoku do innej sekcji).

Sekcje są wykonywane w kolejności, w jakiej zostały zaprogramowane (w takiej kolejności są wyświetlane w oknie podglądu).

Zadanie główne w przykładzie obok składa się z:
- sekcji w języku *Ladder* LD,
- sekcji w języku *Grafcet*,
- sekcji w języku tekstowym ST,
- sekcji w języku *List*.

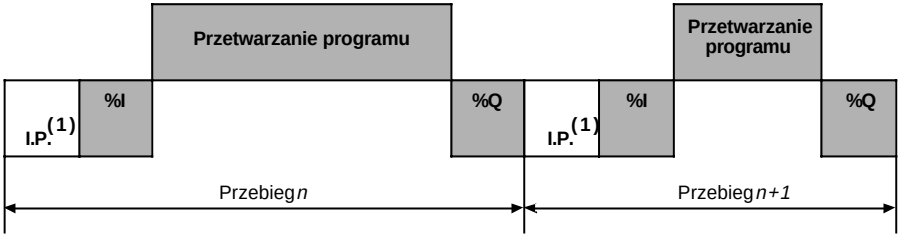


Sekcja języka *Grafcet*:
Sekcja ta składa się z:
- przetwarzania wstępnego (PRL) zaprogramowanego w LD, ST lub IL. Jest wykonywane przed diagramem,
- diagramu *Grafcet* (CHART): na stronach diagramu *Grafcet* programowane są warunki przejściowe związane z operacjami i przejściami kroków oraz skoków,
- przetwarzania końcowego (POST) programowanego języku LD, ST lub IL. Wykonanie operacji następuje po diagramie,
- Procedury SRi (i = 0 do 253). Procedury są programowane oddzielnie, przy czym mogą być wywoływane poczas operacji zdeklarowanych w krokach zarówno podczas przetwarzania wstępnego, jak i końcowego oraz z innych procedur (można zagnieździć maksymalnie 8 procedur).



1.5-2 Cykliczne wykonywanie zadania

Cykliczne wykonywanie zadania jest normalnym trybem pracy sterownika (ustawienie domyślne). "Przebieg" (*scan*) składa się z kilku powiązanych ze sobą cykli zadania głównego MAST. Po zaktualizowaniu stanów wyjść system wykonuje zaprogramowane operacje i rozpoczyna nowy "przebieg" zadania.



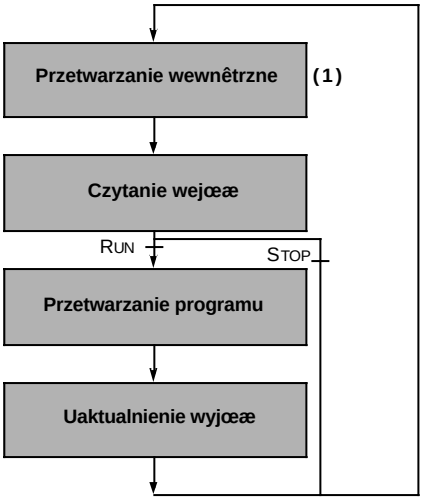
I.P. **Przetwarzanie wewnętrzne**: system w sposób niejawny monitoruje pracę sterownika (zarządza bitami i słowami systemowymi, aktualizuje wartość bieżącą zegara czasu rzeczywistego, aktualizuje stan kontrolek, wykrywa zmiany trybu pracy RUN/STOP, etc) oraz przetwarza żądania pochodzące z terminala,

%I **Czytanie wejść**: zapisanie do pamięci informacji o stanach wejść przypisanych do tego zadania modułów dyskretnych i modułów użytych w aplikacji,

Przetwarzanie: wykonywanie napisanego przez użytkownika programu.

%Q **Aktualizacja wyjść**: zapisanie (zgodnie ze statusem określonym przez program aplikacji) przypisanych do zadania bitów i słów wyjściowych związanych z modułami wyjść dyskretnych i modułami zastosowanymi w aplikacji.

Przebieg operacji i monitorowanie



Sterownik pracuje (RUN) - procesor wykonuje w kolejności następujące operacje: przetwarzanie wewnętrzne, czytanie wejść, wykonanie programu oraz uaktualnienie wyjść.

Sterownik zatrzymany (STOP) - procesor wykonuje następujące operacje:

- przetwarzanie wewętrzne,
- czytanie wejść,
- zależnie od konfiguracji:
  - **tryb zastępczy (*fallback*):** wyjścia przyjmują wartości zastępcze,
  - **zachownie wartości (*maintain*):** wyjścia zachowują bieżące wartości.

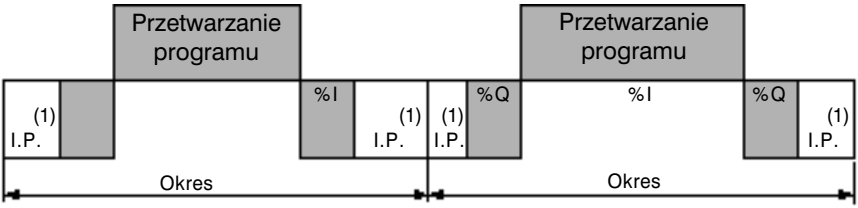
Monitorowanie czasu "przebiegu": przebieg jest kontrolowany - patrz rozdz.1.5-4.

(1) W przypadku sterowników TSX/PMX/PCX 57, przetwarzanie wewnętrzne odbywa się równolegle z przetwarzaniem I/O.

1.5-3 Okresowe wykonywanie zadania

W tym trybie pracy czytanie wejść, wykonywanie programu i uaktualnianie wyjść wykonywane jest okresowo, przy czym okres ten jest definiowany w konfiguracji (od 1 do 255ms).

Na początku "przebiegu" licznik, którego wartość początkowa jest równa okresowi zdefiniowanemu w konfiguracji, rozpoczyna odliczanie. Sterownik musi zakończyć operacje przed upływem nastawionego czasu, bo po tym następuje wyzwolenie nowego "przebiegu".



I.P. **Przetwarzanie wewnętrzne**: system w sposób niejawny monitoruje pracę sterownika (zarządza bitami i słowami systemowymi, aktualizuje wartość bieżącą zegara czasu rzeczywistego, aktualizuje stan kontrolek, wykrywa zmiany trybu pracy RUN/STOP, etc) oraz przetwarza żądania pochodzące z terminala,

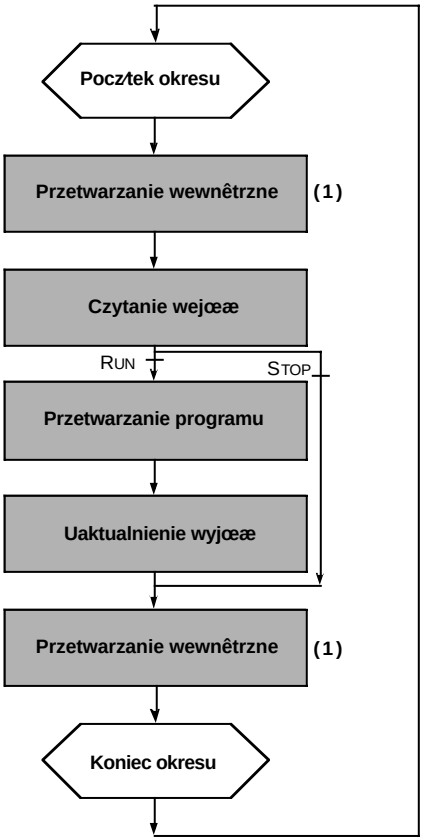
%I **Czytanie wejść**: zapisanie do pamięci informacji o stanach wejść przypisanych do tego zadania modułów dyskretnych i modułów użytych w aplikacji,

Przetwarzanie: wykonywanie napisanego przez użytkownika programu.

%Q **Aktualizacja wyjść**: zapisanie (zgodnie ze statusem określonym przez program aplikacji) przypisanych do zadania bitów i słów wyjściowych związanych z modułami wyjść dyskretnych i modułami zastosowanymi w aplikacji.

(1) W przypadku sterowników TSX/PMX/PCX 57, przetwarzanie wewnętrzne odbywa się równolegle z przetwarzaniem I/O.

Przebieg operacji i monitorowanie



Sterownik pracuje (RUN) - procesor wykonuje w kolejności następujące operacje: przetwarzanie wewnętrzne, czytanie wejść, wykonanie programu oraz uaktualnienie wyjść.
Jeżeli czas okresu nie upłynął, to sterownik wykonuje czynności końcowe do czasu zakończenia okresu (przetwarzanie wewnętrzne).

Jeżeli czas trwania opracji przekracza czas jednego okresu, to sterownik sygnalizuje przekroczenie czasu poprzez nadanie wartości 1 bitowi systemowemu zadania %S19. Przetwarzanie trwa do czasu jego zakończenia (czas realizacji jednego "przebiegu" nie może przekroczyć czasu zdefiniowanego w układzie monitorującym *watchdog*). Następny "przebieg" jest inicjowany po niejawnym zapisaniu stanów wyjść w "przebiegu" bieżącym.

Sterownik zatrzymany (STOP) - procesor wykonuje następujące operacje:
- przetwarzanie wewętrzne,
- czytanie wejść,
- zależnie od konfiguracji:
  - tryb zastępczy (*fallback*): wyjścia przyjmują wartości zastępcze,
  - zachownie wartości (*maintain*): wyjścia zachowują bieżące wartości.

Monitorowanie przebiegu:
Kontrolę sprawuje się dwutorowo:
- przekroczenie czasu okresu (*period overrun*),
- przy wykorzystaniu układu monitorującego typu *watchdog* (patrz rozdział 1.5-4).

(1) W przypadku sterowników TSX/PMX/PCX 57, przetwarzanie wewnętrzne odbywa się równolegle z przetwarzaniem I/O.

1.5-4 Monitorowanie czasu "przebiegu" zadania

Programowy układ śledzący *watchdog* (zadania cykliczne i okresowe)

Czas wykonywania zadania głównego MAST (zarówno dla pracy cyklicznej, jak i okresowej) jest kontrolowany przez sterownik (układ śledzący *watchdog*) pod kątem przekroczenia maksymalnego czasu *Tmax* przewidzianego w konfiguracji na wykonanie zadania (domyślnie - 250 ms, maksymalnie - 500 ms).

W przypadku przekroczenia tego czasu aplikacja traktowana jest jako uszkodzona, w efekcie czego następuje zatrzymanie sterownika (wyjście alarmowe %Q2.0 sterownika TSX 37 przyjmuje wartość 0, o ile zostało ono skonfigurowane, natomiast w przypadku sterowników TSX/PMX/PCX 57 przekaźnik alarmowy modułu zasilania przełącza się na pozycję 0).

Do monitorowania wykonania zadania wykorzystuje się bit %S11.

Sygnalizuje on przekroczenie zadeklarowanego czasu (przyjmuje wtedy wartość 1).

W przypadku sterowników TSX/PMX/PCX 57 zadeklarowany czas maksymalny dla układu monitorującego musi być dłuższy niż czas trwania okresu.

W przypadku okresowego wykonywania zadania, dla wykrycia przekroczenia czasu wykorzystuje się dodatkowe parametry:

- %S19 : sygnalizuje przekroczenie czasu trwania okresu. W przypadku, gdy czas "przebiegu" zadania jest dłuższy niż jego okres, system nadaje mu wartość 1.
- %SW0 : w słowie tym zapisana jest wartość czasu trwania okresu (w ms). Wartość z jaką jest inicjowane to słowo podczas zimnego startu jest definiowana w konfiguracji i może być modyfikowana przez użytkownika.

Czas wykonywania zadania MAST

Wymienione poniżej słowa systemowe zawierają dane dotyczące czasu "przebiegu" (*scan*) zadania:

- %SW30 zawiera czas trwania ostatniego przebiegu zadania.
- %SW31 zawiera czas trwania najdłuższego przebiegu zadania.
- %SW32 zawiera czas trwania najkrótszego przebiegu zadania.

Uwaga:
Dane te są dostępne również z poziomu edytora konfiguracyjnego.

1.6 Struktura wielozadaniowa

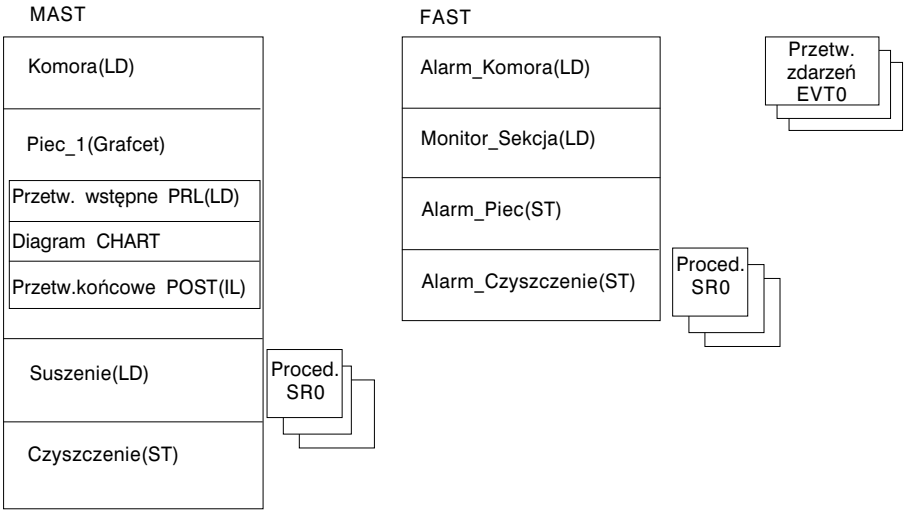
1.6-1 Wprowadzenie

Struktura wielozdaniowa jest następująca:

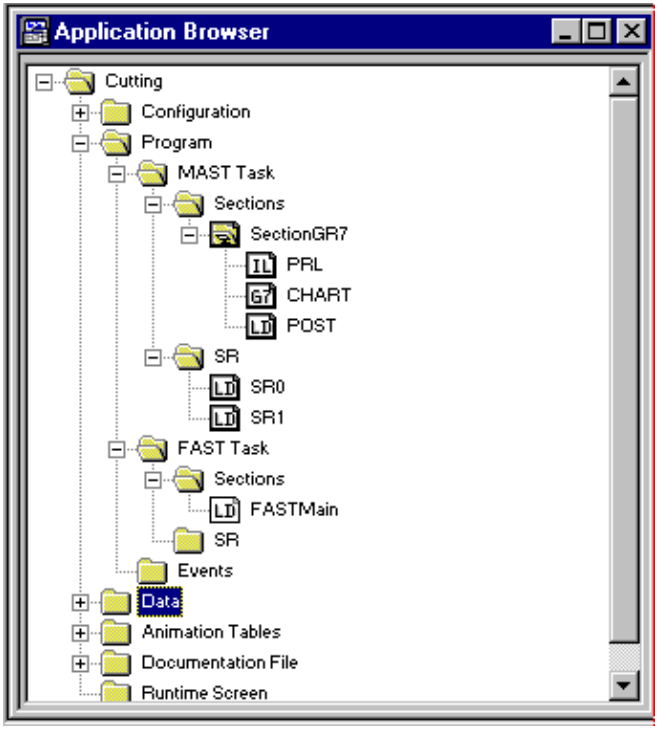
- Zadanie główne MAST jest zawsze obecne w strukturze, przy czym może być wykonywane cyklicznie albo okresowo.
- Zadanie szybkie FAST (opcjonalne), przy czym zawsze jest okresowe.
- Zadania wyzwalane zdarzeniami EVTi wywoływane przez system po wystąpieniu określonych warunków na module I/O. Ten rodzaj przetwarzania ma charakter opcjonalny i jest wykorzystywane w aplikacjach wymagających odpowiedzi w bardzo krótkim czasie aby wykonać określone operacje na wejściach i wyjściach.

Zadania główne MAST i szybkie FAST są podzielone na sekcje (patrz opis sekcji w rozdziale 1.5).

Tylko zadanie główne MASTER może zawierać seksję języka *Grafcet*.
Zadania wyzwalane zdarzeniami składają się tylko z jednej sekcji, której nazwy nie można modyfikować



Przykład struktury wielozadaniowej



Zarządzanie zadaniami:

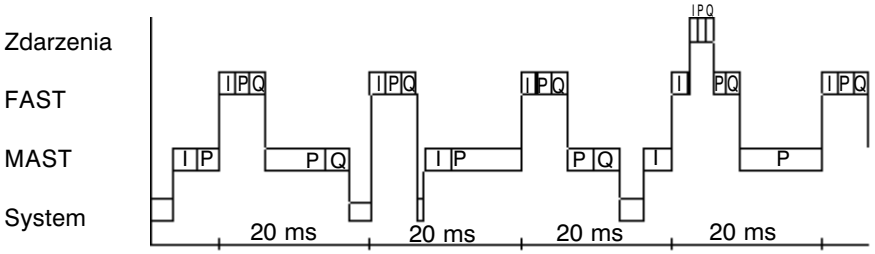
Zadanie nadrzędne MAST uaktywnia się samoistnie tak samo jak i zadanie szybkie, o ile oczywiście zostało wcześniej zaprogramowane. Przetwarzanie zdarzeń (*event-triggered task*) uaktywnia się pod wpływem wystąpienia zdefiniowanego warunku.

W przypadku wystąpienia takiego zdarzenia oraz na czas wykonywania zadania szybkiego FAST następuje zawieszenie wykonywania zadań o niższym priorytecie.

Wykonywanie zadania szybkiego i zadań wyzwalanych zdarzeniami można kontrolować za pomocą programu wykorzystując następujące bity systemowe:

- %S30 - uaktywnia i dezaktywuje zadanie główne MAST.
- %S31 - uaktywnia i dezaktywuje zadanie szybkie FAST.
- %S38 - uaktywnia i dezaktywuje przetwarzanie zdarzeń EVTi.

Przykład przetwarzania wielozadaniowego:
- zadanie główne wykonywane cyklicznie,
- zadanie szybkie o zadanym okresie 20 ms,
- zadania wyzwalane zdarzeniami.



1.6-2 Zadanie główne MAST

Jest to zadanie o najniższym priorytecie, które stanowi zdecydowaną większość programu aplikacji.

Jest ono zorganizowane według modelu opisanego w poprzednim rozdziale: niejawne odczytanie stanów wejść, wykonanie programu aplikacyjnego i niejawne zaktualizowanie stanów wyjść.

Bez względu na to, czy zadanie jest wykonywane cyklicznie, czy też okresowo, jest ono monitorowane przez układ sprawdzający (*watchdog*), czy aplikacja nie jest wykonywana zbyt długo (co jest sytuacją nienormalną). W przypadku przekroczenia zdefiniowanego czasu trwania zadania bit %S11 przyjmuje wartość 1 (błąd aplikacji), co powoduje zatrzymanie pracy sterownika.

Bit systemowy %S30 umożliwia zablokowanie i odblokowanie zadania głównego.

1.6-3 Zadanie szybkie FAST

To zadanie ma wyższy priorytet od zadania głównego i zawsze jest okresowe dzięki czemu możliwe jest wykonania zadania o niższym priorytecie (żeby wystarczyło czasu na jego wykonanie).

Dodatkowo, operacje przetwarzania sprzężone z tym zadaniem powinny być bardzo szybkie tak, by nie oddziaływały negatywnie na zadanie główne.

Okres zadania szybkiego definiuje się podczas konfiguracji, przy czym musi się on zawierać w przedziale od 1 do 255 ms. Może być on porównywalny z okresem zadania MAST tak, by umożliwić jego dopasowanie do okresowych operacji przetwarzania, które są stosunkowo wolne ale mają wyższy priorytet.

Zadanie jest monitorowane przez układ sprawdzający (*watchdog*), czy aplikacja nie jest wykonywana zbyt długo (co jest sytuacją nienormalną). W przypadku przekroczenia zdefiniowanego czasu trwania zadania bit %S11 przyjmuje wartość 1 (błąd aplikacji), co powoduje zatrzymanie pracy sterownika.

Sterowanie zadaniem szybkim FAST

W słowie systemowym %SW1 zapisany jest zdeklarowany czas trwania okresu. Podczas zimnego startu do tego słowa zapisywana jest wartość określona w konfiguracji, przy czym może ona być modyfikowana przez użytkownika za pośrednictwem programu lub terminala.

Do monitorowania wykonywania tego zadania służą następujące słowa systemowe:

- %S19 : sygnalizuje przekroczenie czasu przydzielonego na wykonanie zadania. Przyjmuje wartość 1 gdy czas "przebiegu" zadania jest dłuższy od okresu zadania.
- %S31 : umożliwia zablokowanie i odblokowanie zadania. W przypadku zimnego startu aplikacji system nadaje mu, po pierwszym "przejściu" (*scan*) zadania głównego MAST, wartość 0. Zmiana wartości na 1 powoduje odblokowanie, a na 0 - zablokowanie zadania szybkiego FAST.

Wyświetlenie czasu trwania zadania szybkiego

Informacje o czasie trwania "przebiegu" zadania szybkiego zawierają następujące słowa systemowe:

- %SW33 zawiera czas trwania ostatniego "przebiegu".
- %SW34 zawiera czas trwania najdłuższego "przebiegu".
- %SW35 zawiera czas trwania najkrótszego "przebiegu".

1.6-4 Przypisywanie kanałów I/O do zadania MAST i FAST

Poza realizacją programu aplikacji zadania FAST i MAST mogą inicjować funkcje systemowe zarządzające niejawną wymianą danych za pomocą, przypisanych do tych zadań, kanałów I/O.

Przypisania kanału, czy też grupy kanałów, do zadania dokonuje się w oknie konfiguracyjnym odpowiedniego modułu.

Ponieważ dyskretne moduły I/O są podzielone na moduły po 8 kolejnych kanałów (kanały 0 do 7, kanały 8 do 15, itd.) można je przypisywać do zadań MAST i FAST grupami po 8 kanałów:

- Wejścia 0 do 7 przypisuje się do zadania głównego MAST.
- Wejścia 8 do 15 przypisuje się do zadania szybkiego FAST.
- Wyjścia 0 do 7 przypisuje się do zadania głównego MAST.
- Wyjścia 8 do 15 przypisuje się do zadania szybkiego FAST.

Każdy kanał modułu licznika może być przypisany do zadania MAST lub FAST. Dla przykładu - kanały modułu licznika dwukanałowego można przypisać następująco:

- kanał 0 do zadania głównego MAST,
- kanał 1 do zadania szybkiego FAST.

Kanały analogowego modułu wejściowego TSX 37 mogą być przypisywane tylko do zadania głównego MAST. Jednakże istnieje możliwość przypisania analogowych kanałów wyjściowych do zadania MAST lub FAST, w modułach po 2 kanały:

- kanały 0 i 1 do zadania głównego MAST,
- kanały 2 i 3 do zadania szybkiego FAST.

Kanały analogowych modułów I/O TSX 57 mogą być przypisywane do zadania MAST lub FAST. Każdy kanał izolowanego modułu analogowego I/O (4 izolowane kanały) przypisuje się indywidualnie. W przypadku innych modułów stosuje się przypisywanie w grupach 4-kanałowych.

1.6-5 Zadania wyzwalane zdarzeniami

Przetwarzanie zdarzeń (*event-triggereg processing*) stosuje się w celu zmniejszenia czasu odpowiedzi programu na zdarzenia typu komenda.

Zdarzenia typu komenda

Są to **zdarzenia zewnętrzne** (*external events*) powiązane z aplikacją. Wystąpienie zdarzenia tego typu powoduje skierowanie programu na wykonanie operacji przypisanej do kanału I/O, na którym pojawiło się to zdarzenie. Wejścia (%I, %IW, %ID) przypisane do kanału I/O, w którym nastąpiło zdarzenie podlegają uaktualnieniu przez system jeszcze przed uruchomieniem przetwarzania. Możliwe są konfiguracje:

- 8 zdarzeń w przypadku sterownika TSX 37-10 (EVT1 do EVT8),
- 16 zdarzeń w przypadku sterownika TSX 37-21/22 (EVT0 do EVT15),
- 32 zdarzenia w przypadku sterowników TSX/PMX/PCX 57-10 (EVT0 do EVT31),
- 64 zdarzenia w przypadku sterowników TSX/PMX/PCX 57-20 (EVT0 do EVT63).

Przypisania kanału i numeru zdarzenia dokonuje się w oknie konfiguracyjnym kanału.

W przypadku TSX 37 przetwarzanie zdarzeń może być wyzwalane przez:

- wejścia 0 do 3 modułu z pozycji 1(zbocze opadające lub rosnące sygnału),
- kanał (lub kanały) modułu licznika,
- kanały licznika modułu 1 (o ile jest skonfigurowany jako licznik),
- odebranie telegramu w przypadku TSX 37-21/22 wyposażonego w moduł TSX FPP20.

W przypadku sterowników TSX/PMX/PCX 57 przetwarzanie można wyzwalać:

- wejściami modułów DEY 16 FK i DMY 28 FK,
- kanałami modułów liczników,
- kanałami modułów CAY,
- kanałami modułów CFY sterujących silnikami krokowymi,
- kanałami komunikacyjnymi "FPP20".

Zarządzanie przetwarzaniem zdarzeń

Przetwarzanie zdarzeń można globalnie zablokować i odblokować przy pomocy programu aplikacyjnego, za pośrednictwem bitu systemowego %S38. Jeżeli w czasie, gdy przetwarzanie zdarzeń jest zablokowane pojawią się jakieś zdarzenia, to zostaną one "zgubione" (przyporządkowane im operacje nie zostaną wykonane).

Dwie instrukcje języka PL7 - MASKEVT() i UNMASKEVT() umożliwiają nakładanie i zdejmowanie maski w odniesieniu do przetwarzania zdarzeń. Jeżeli w czasie, gdy na przetwarzanie zdarzeń będzie nałożona maska pojawią się jakieś zdarzenia, to zostaną one zachowane przez system i przyporządkowane im operacje zostaną wykonane po zdjęciu maski.

Dla TSX 37-10 wszystkie komendy (jest ich 8) mają ten sam priorytet. Stąd też jedna operacja przetwarzania zdarzenia nie może być przerwana przez drugą.
Dla TSX 37-21/22 i TSX/PMX/PCX 57 , są 2 poziomy priorytetu dla przetwarzania komend: zdarzenie 0 (EVT0) ma wyższy priorytet niż pozostałe.

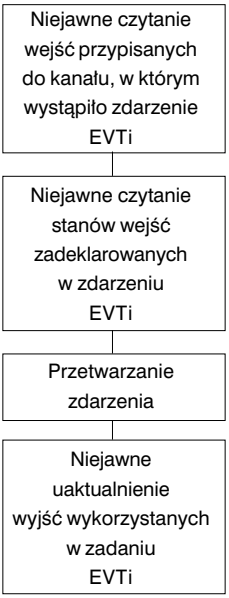
Wykorzystanie kanałów I/O przy przetwarzaniu zdarzeń

W zadaniach wyzwalanych zdarzeniami można wykorzystywać inne kanały I/O niż te, które są przyporządkowane poszczególnym zdarzeniom. Wymiana informacji przeprowadzana jest przez system w sposób niejawny przed (%I) i po (%Q) przewtorzeniu zdarzenia. Ta wymiana może odnosić się do pojedynczego kanału (np. w module licznika) lub do grupy kanałów (moduł dyskretny). W drugim przypadku, jeśli w trakcie przetwarzania następuje zmodyfikowanie na przykład wyjść 2 i 3 modułu dyskretnego, to do modułu zostaną przeniesione obrazy wyjść od 0 do 7.

Podsumowanie operacji przetwarzania i wymiany

IT — Przetwarzanie zdarzeń (zadanie EVTi)

Pojawienie się zdarzenia kieruje program aplikacyjny ku operacji przetwarzania zdarzeń przyporządkowanej kanałowi I/O, w którym zdarzenie wystąpiło:



- Wszystkie wejścia, powiązane z kanałem wyzwalającym przetwarzanie są automatycznie czytane.
- Odczytanie wszystkich wejść, których odczytanie użytkownik zadeklarował na przypadek wystąpienia zdarzenia EVTi.
- Przetwarzanie - powinno trwać tak krótko, jak to tylko możliwe.
- Uaktualnienie wszystkich wyjść wykorzystanych przez użytkownika w zdarzeniu EVTi. Aktualizowane mogą być również wyjścia powiązane z kanałem, który wyzwolił przetwarzanie.

Uwaga:
W przypadku modułów dyskretnych TSX DEY16FK i TSX DMY28FK, wejście, na którym wystąpiło zdarzenie nie powinno być poddawane testowaniu podczas przetwarzania zdarzenia (nie należy aktualizować wartości). Sprawdzenie, na którym zboczu sygnału nastąpiło wyzwolenie zdarzenia powinno być przeprowadzone przy użyciu słowa statusu:
%IWxy.i:X0 = 1 --> zbocze narastające,
%IWxy.i:X1 = 1 --> zbocze opadające.

Uwagi

Podczas przetwarzania zdarzeń w analogowych modułach wejściowych TSX 37, które mogą być wykorzystane tylko w zadaniu MAST, nie należy dokonywać zamiany wejść.

Zamiany We/Wy powiązanej z zadaniem EVTi wykorzystywanych przez program dokonuje się kanał po kanale (moduły liczników) lub w grupach kanałów (moduły dyskretne). Stąd też, jeśli w podczas przetwarzania modyfikuje się np. wyjścia 2 i 3 modułu dyskretnego, do modułu zostanie przeniesiony obraz wyjść 0 - 7.

Dla TSX 37, w każdej operacji przetwarzania zdarzenia można zadeklarować maksymalnie wymianę 2 modułów wejściowych (przed rozpoczęciem przetwarzania) i 2 modułów wyjściowych (po zakończeniu przetwarzania).

Podlegające wymianie wejścia (i grupy powiązanych z nimi kanałów) są, podczas przetwarzania, aktualizowane (utrata rejestru operacji *LOG* i w konsekwencji, informacji o zboczu). Dlatego też, w zadaniu MAST oraz FAST, nie powinno się testować zbocz sygnałów na tych wejściach.

W przypadku TSX/PMX/PCX 57 liczba dokonanych wymian, w zależności od rodzaju procesora, jest ograniczona do:

Liczba wymian zastosowanych w przetwarzaniu zdarzeń EVTi	**P57-1• (32 zdarz.)**	**P57-2•/3• (64 zdarz.)**
Max l-ba wymian dyskretnych	32 wymiany	128 wymian
Max l-ba wymian analogowych	8 wymian	16 wymian
Max l-ba innych rodzajów wymian	4 wymiany	16 wymian

Dla dyskretnych I/O w wymianie biorą udział grupy po 8 kanałów. Wymiana jest generowana, w przypadku użycia wejść z grup 8-kanałowych (inne niż te, na których wystąpiło zdarzenie) oraz gdy następuje zapisanie stanu wyjścia kanałów z grupy.

Dla analogowych oraz pozostałych modułów I/O wymiana jest generowana w przypadku użycia wejść kanału (innego niż ten, w którym wystąpiło zdarzenie) oraz przy zapisywaniu wyjść kanału.

Wyświetlanie liczby przetworzonych zdarzeń

Liczbę przetworzonych zdarzeń zawiera słowo %SW48. Podczas zimnego startu jest ono inicjowane z wartością 0, która jest następnie zwiększana przez system każdorazowo, gdy wystąpi zdarzenie. Może być ono modyfikowane przez użytkownika.

Bit %S39 informuje o utracie zdarzenia.

Uwaga: Podsumowanie operacji jakie można zaprogramować podczas przetwarzania zdarzenia zamieszczono w części A, w rozdziale 5.2 instrukcji opisującej funkcje związane z aplikacjami.

2.1 Prezentacja języka Ladder

2.1-1 Wprowadzenie

Programy napisane w języku *Ladder* składają się z szeregu labelek wykonywanych sekwencyjnie przez sterownik.

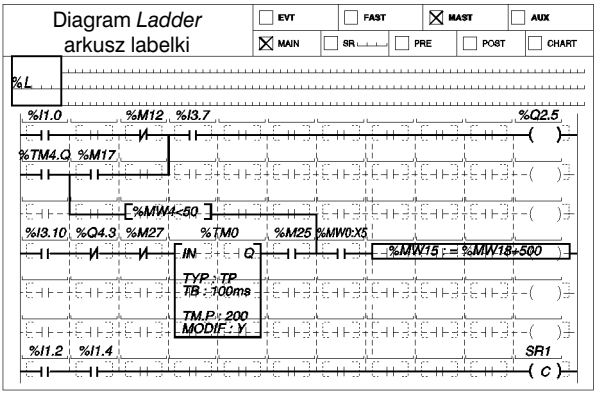
Labelkę tworzą elementy graficzne ograniczone z dwóch stron liniami bazowymi (w notacji elektrycznej odpowiadają one zasilaniu). Odwzorowują one:

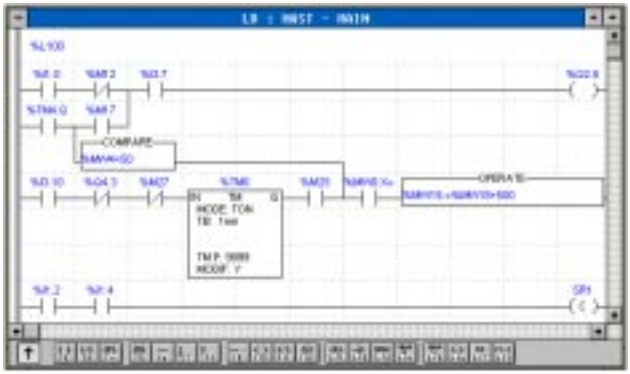
- wejścia i wyjścia (I/O) sterownika (przyciski, czujniki, przekaźniki, kontrolki, itp.),
- standardowe funkcje systemu sterującego (zegary, liczniki, itp.),
- operacje arytmetyczne, logiczne i specjalne,
- zmienne wewnętrzne sterownika.

Elementy graficzne łączone są ze sobą za pomocą poziomych i pionowych linii.

Każda labelka może składać się maksymalnie z 16 linii i 11 kolumn (dla sterowników TSX/PMX/PCX 57) lub 7 linii i 11 kolumn (dla TSX 37) podzielonych na dwie strefy:

- strefę warunków (*test zone*), która definiuje warunki niezbędne dla wykonania określonej akcji,
- strefę operacji (*action zone*), w której zdefiniowane są operacje, które zostaną wykonane po przetestowaniu warunków w strefie warunków.





2.1-2 Elementy graficzne

Elementy podstawowe

Są to elementy zajmujące 1 komórkę (wysokość - 1 linia, szerokość - 1 kolumna).

Grupa elementów		Symbol	Funkcja
Elementy warunków	• Styk normalnie otwarty		Styk się zamyka, gdy kontrolujący go bit obiektowy ma wartość 1.
	• Styk normalnie zamknięty		Styk jest zamknięty, gdy kontrolujący go bit obiektowy ma wartość 0.
	• Styki reagujące na zmianę sygnału		Zbocze narastające: styk zamknięty, gdy bit obiektowy zmienia stan z 0 na 1.
			Zbocze opadające: styk zamknięty, gdy bit obiektowy zmienia stan z 1 na 0.
Elementy łączące	• Łącznik poziomy		Służy do szeregowego łączenia elementów graficznych pomiędzy liniami bazowymi.
	• Łącznik pionowy		Służy do łączenia równoległego elementów graficznych.
Elementy strefy akcji	• Sprzężenie proste		Nadaje bitowi sprzężonemu z obiektem, wartość wynikającą ze strefy warunków.
	• Sprzężenie odwrotne		Nadaje bitowi odwrotną wartość w stosunku do wartości ze strefy warunków.
	• Przełącznik prosty		Nadaje sprzężonemu bitowi wartość 1, gdy wynikiem testowania warunków jest 1.
	• Przełącznik odwrotny		Kasuje do 0 bit sprzężony z obiektem, gdy wynikiem testowania warunków jest 1.
	• Skok (*JUMP*) warunkowy do innej labelki	**->> %Li**	Umożliwia połączenie z labelką o znanej etykiecie, zarówno do przodu, jak i do tyłu. Skoki dopuszczalne są tylko w obrębie jednolitej części programu (program główny, procedura, itd.). **Wykonanie skoku powoduje**: • przerwanie realizacji aktualnej labelki, • wykonanie wywołanej labelki, • część programu między labelką ze skokiem a labelką docelową skoku nie jest wykonywana.
	• Sprzężenie warunkowe		Z języka Grafcet, używane gdy określa się warunki związane z przejściem bramki (*transition*). Umożliwia przejście do kolejnego kroku.

Grupa elementów		Symbol	Funkcja
Elementy strefy akcji (c.d.)	• Wywołanie procedury (*CALL*)	—(C)—	Powoduje przejście do wykonania procedury, gdy wynikiem testowania warunków akcji jest wartość 1. **Wywołanie procedury powoduje:** • przerwanie wykonywania bieżącej labelki, • wykonanie procedury, • powrót do wykonywania labelki.
	• Powrót do modułu	**<RETURN>**	Zarezerwowany dla procedur (SR). Umożliwia powrót do modułu, z którego procedura została wywołana, gdy wynikiem testu jest 1.
	• Zatrzymanie programu	**<HALT>**	Zatrzymuje wykonywanie programu, gdy warunek przyjmuje wartość 1.

Standardowe bloki funkcyjne

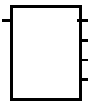
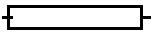
Grupy elementów		Symbol	Function
Elementy strefy warunków	• Bloki: Zegar Licznik Przerzutnik Rejestr Bęben		Każdy ze standardowych bloków posiada wejście lub wyjście, umożliwiające połączenie go z innymi elementami graficznymi. Funkcje poszczególnych bloków są opisane w części B. Wielkość: patrz rozdział 2.2-5.

Blok DFB (blok funkcjonalny tworzony przez użytkownika)

Grupa elementów		Symbol	Function
Elementy strefy warunków	• Bloki programowalne		Każdy z bloków typu DFB posiada wejścia i wyjścia, dzięki którym można je łączyć z innymi elementami graficznymi (tak jest dla obiektów bitowych) lub przypisywać je do obiektów numerycznych lub tablic. Funkcje bloków typu DFB opisano w sekcji 6. Wymiary: patrz rozdział 2.2-5

Sterowniki serii TSX 37 nie mają bloków typu DFB.

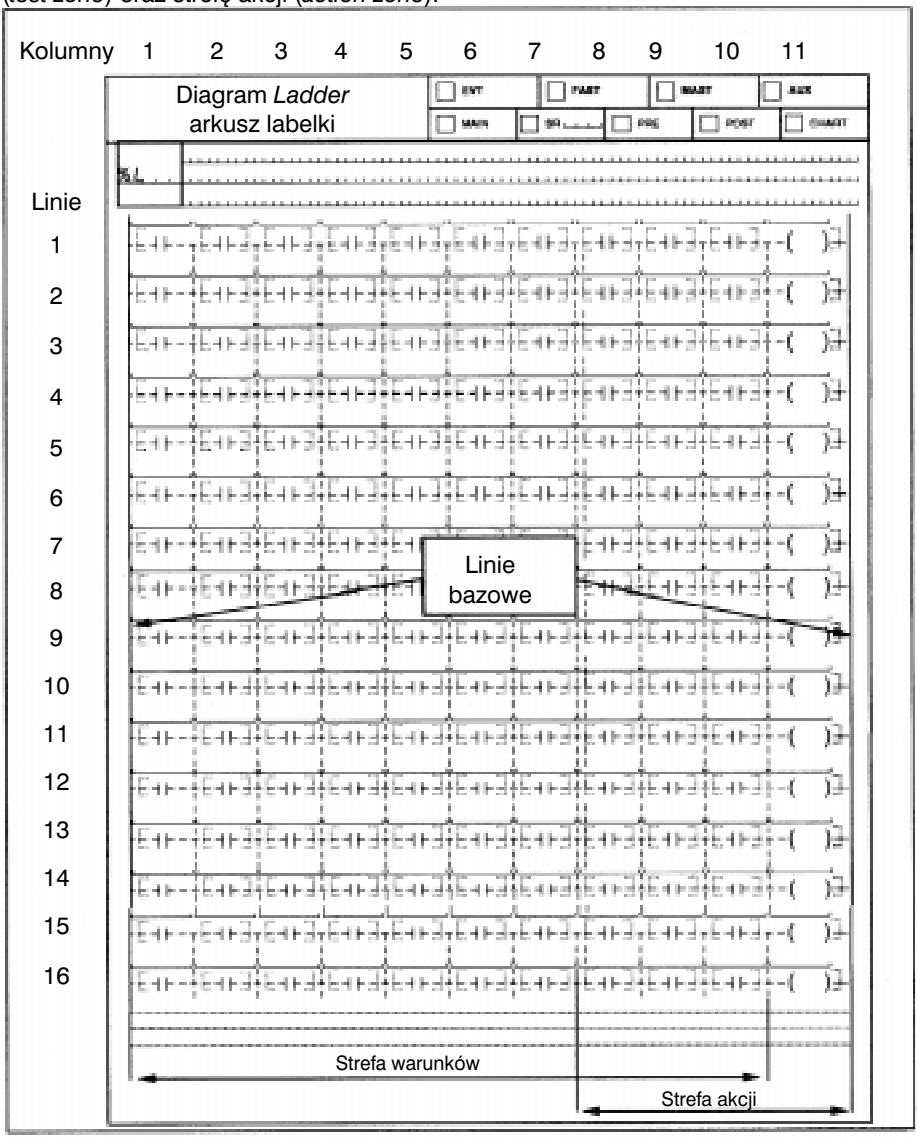
Bloki operacyjne

Grupa elementów		Symbol	Funkcja
Elementy strefy warunków	• Porównywanie złożone *Comparison*		Umożliwia porówywnanie 2 argumentów. W zależności od wyniku odpowiednie wyjście bloku zmienia stan na 1. Wielkość: 2 kolumny / 4 linie.
	• Porównywanie proste *Comparison*		Umożliwia porównywanie 2 argumentów. Wyjście zmienia stan na 1, kiedy wynik porównania jest zadawalający. Blok może zawierać max 4096 znaków. Wielkość: 2 kolumny / 1 linia.
Elementy strefy akcji	• Blok operacyjny		Pozwala na wykonanie operacji arytmetycznych , logicznych, itp., zdefiniowanych przy użyciu strukturalnego języka tekstowego *ST*. Blok może zawierać max 4096 znaków. Wielkość: 4 kolumny / 1 linia.

2.2 Struktura labelki

2.2-1 Wiadomości ogólne

Labelka składa się grupy elementów graficznych, połączonych między sobą liniami poziomymi i pionowymi rozmieszczonymi pomiędzy liniami bazowymi.
Labelka może składać się maksymalnie z 16 linii i 11 kolumn (sterowniki TSX/PMX/PCX 57) lub z 7 linii i 11 kolumn (dla TSX 37) podzielonych na dwie strefy: strefę warunków (*test zone*) oraz strefę akcji (*action zone*).

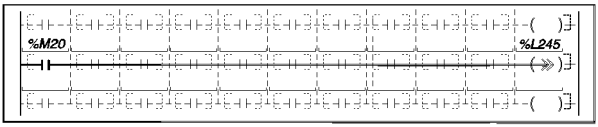


2.2-2 Etykiety

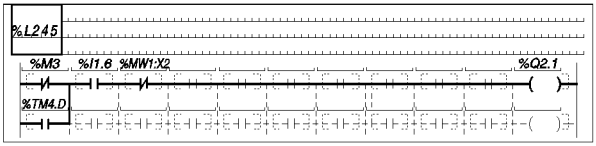
Etykiety umożliwiają identyfikację labelki w obrębie jednolitej części programu (program główny, procedura, itd.). Definiowanie etykiet nie jest obowiązkowe.

Etykiety definiuje się przy użyciu następującej składni: %Li (*i* od 0 do 999). Są one wyświetlane w lewym, górnym rogu labelki.

Dana etykieta może być nadana labelce w obrębie modułu programowego tylko jeden raz.



Jeżeli w programie mają być wykonywane skoki pomiędzy labelkami, to muszą one mieć nadane etykiety.



System wykonuje labelki w takiej kolejności, w jakiej zostały zapisane, a nie w kolejności wynikającej z numerów etyiet.

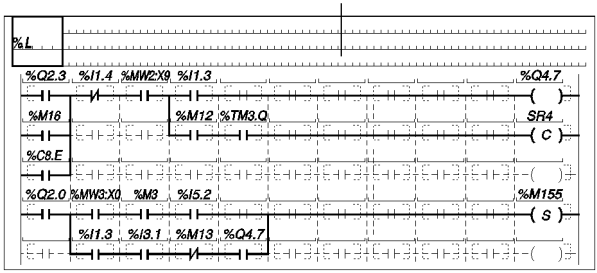
2.2-3 Komentarze

Komentarz jest przypisany do labelki i może składać się maksymalnie z 222 znaków alfanumerycznych ograniczonych na początku i na końcu znakami **(*** i ***)**. Komentarz ułatwia rozpoznanie funkcji danej labelki, jej przeznaczenia i wykonywanych w niej operacji. Stosowanie komentarzy nie jest obowiązkiowe.

Strefa komentarza

Komentarz jest wyświetlany w górnej części labelki, w strefie zarezerwowanej.

Usunięcie labelki z programu powoduje usunięcie również jej komentarza.



Komentarze są zapisywane w pamięci programowej (*program memory*) sterownika tak, że użytkownik ma do nich dostęp w dowolnym momencie.

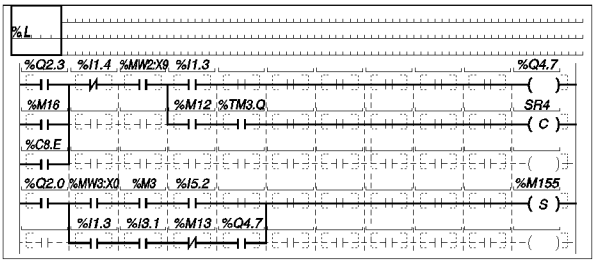
2.2-4 Labelki

Wygląd labelki jest podobny do schematów funkcjonalnych przekaźników.

Proste elementy graficzne służące do określania warunków i wykonywania prostych operacji zajmują w labelce jedną komórkę labelki (przecięcie 1 kolumny z 1 linią).

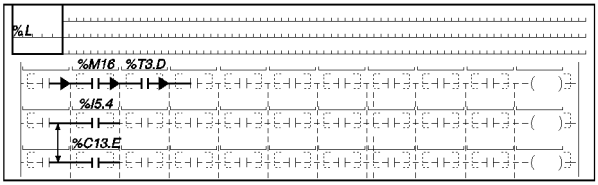
Wszystkie linie łączące styki biorą początek z lewej linii bazowej i kończą się na prawej linii bazowej.

Warunki zapisywane są w kolumnach od 1 do 10. Operacje są zawsze umieszczane w kolumnie 11.



Kolejność realizacji labelki jest następująca:

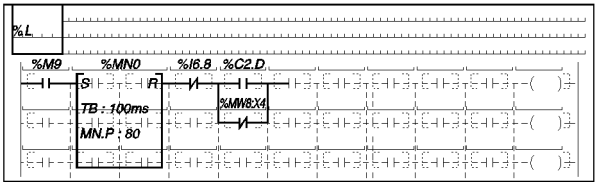
- linie poziome - **od lewej strony do prawej**,
- linie pionowe - w obydwu kierunkach.



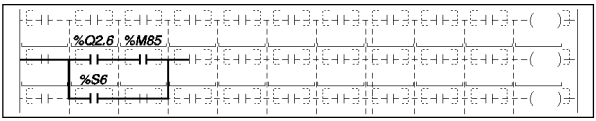
Strefa warunków

Ta strefa zawiera:

- styki (*contact*), którym można przypisać wszystkie, zdefiniowane powyżej, bity obiektowe,
- bloki funkcyjne,
- bloki porównywania.



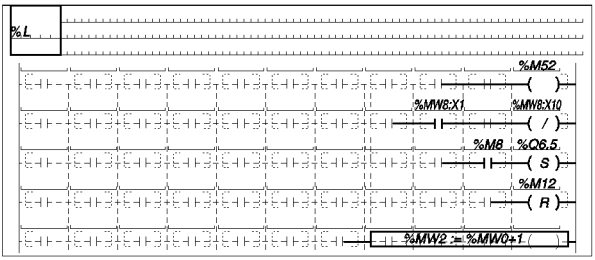
Reakcja na zbocza sygnałów może być przypisywana jedynie bitom obiektowym I/O oraz bitom wewnętrznym.



Strefa akcji

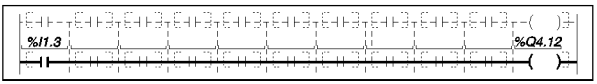
Strefa ta zawiera:

- sprzężenia (cewki) proste, odwrotne, przełączniki proste i odwrotne, przypisywane dowolnym bitom obiektowym, zapisywanym przez użytkownika,
- bloki operacyjne,
- inne: wywołania, skoki, zatrzymanie, powrót.

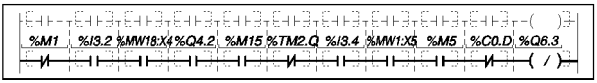


Labelki proste

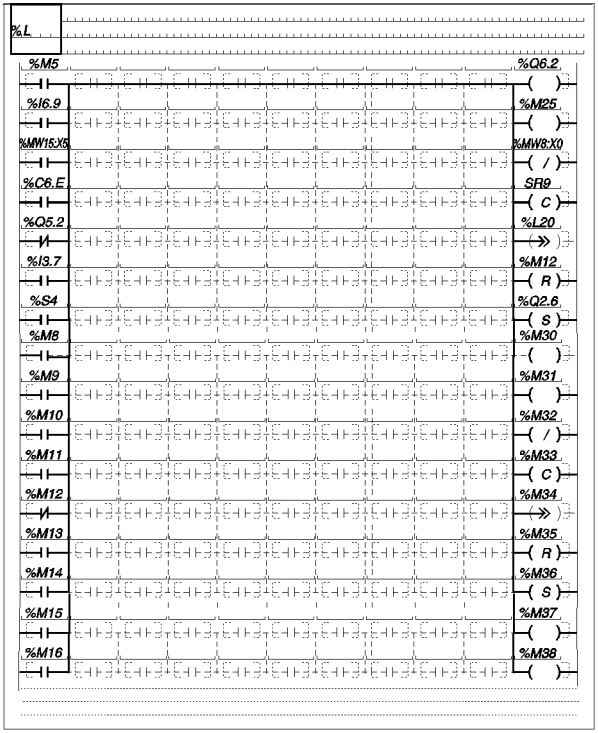
Sterowanie sprzężeniem (cewką) uzależnione od położenia styku.



Zastosowanie 10 styków połączonych szeregowo.



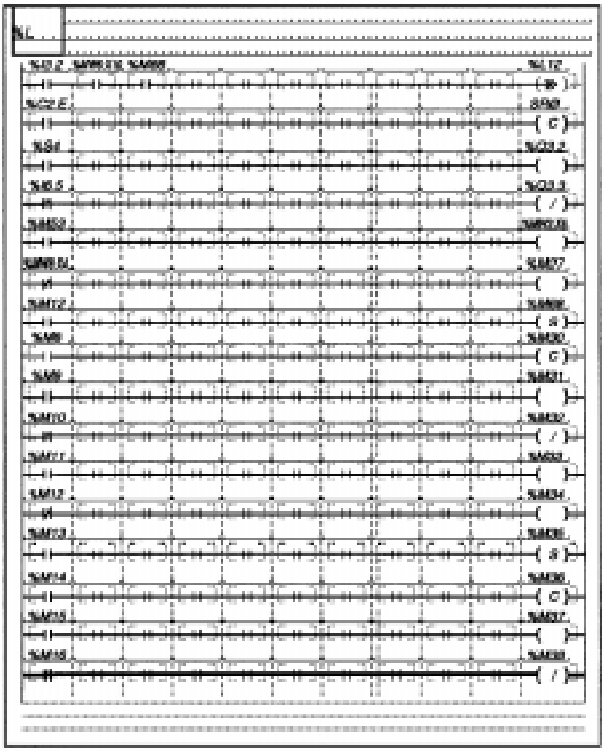
Zastosowanie 16 styków (7 styków dla TSX 37) połączonych równolegle (ustawione w jednej kolumnie) do kontrolowania 16 cewek (7 dla TSX 37) również połączonych równolegle.



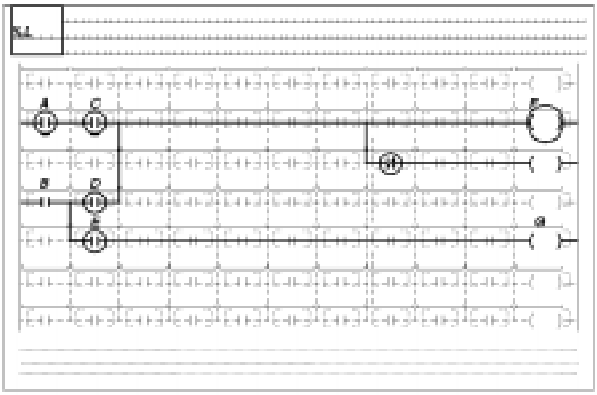
Labelki złożone

Labelkę można podzielić na kilka niezależnych linii, w których styki niezależnie sterują sprzężeniami.

Maksymalnie można zestawić 16 niezależnych linii (7 linii dla TSX 37).



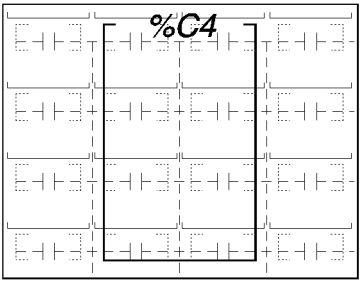
Labelki złożone, w których zastosowano różne rozwiązania opisane powyżej. Symbole o stanach logi-



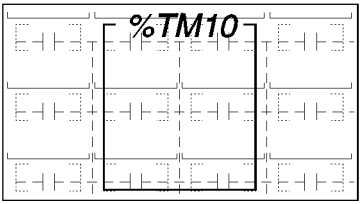
2.2-5 Labelki zawierające bloki operacyjne i funkcyjne

- Bloki funkcyjne umieszcza się w strefie warunków (*test zone*).

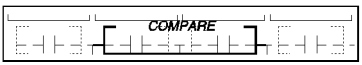
Licznik dwukierunkowy — 2 kolumny
Blok porównywania złożony — 4 linie



Zegary — 2 kolumny
Przerzutnik monostabilny — 3 linie
Rejestr
Bęben



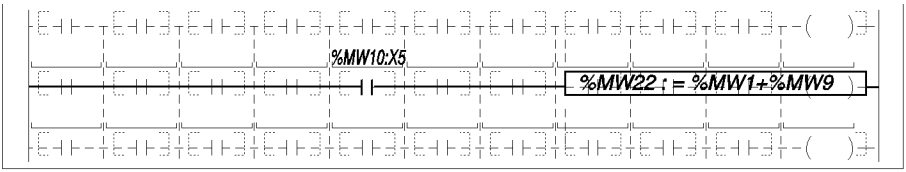
Blok porównywania prosty — 2 kolumny
1 linia



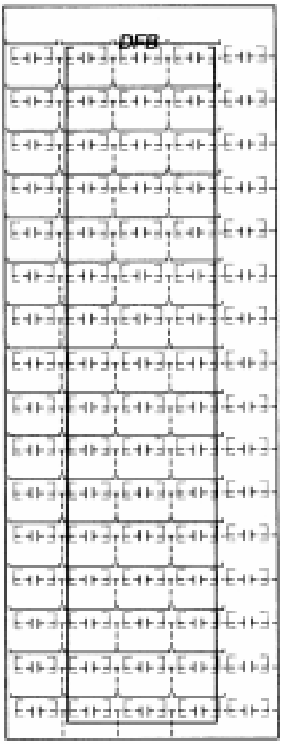
Uwaga:

W standardowych blokach funkcyjnych nie podłączone wejścia mają stan 0.

- Bloki operacyjne zawsze umieszcza się w strefie akcji (*action zone*). Zajmują one obszar 4 kolumn w obrębie jednej linii. Operacje definiuje się przy pomocy języka tekstowego ST. Są one zawsze podłączone bezpośrednio do prawej linii bazowej.



- Bloki DFB (bloki własne użytkownika) umieszcza się w strefie warunków. Wielkość zajmowanego obszaru zależy od liczby wykorzystanych wejść i wyjść:



Szerokość bloku jest stała - zajmuje on 3 kolumny.

Liczba zajmowanych linii jest większa o 1 od jednej z dwóch liczb:

- liczba wejść oraz I/O

  lub

- liczba wyjść oraz I/O.

Przykład:
Liczba wejść i I/O wynosi 4, liczba wyjść i I/O wynosi 3, tak więc wysokość bloku równa się 4 + 1 = 5.

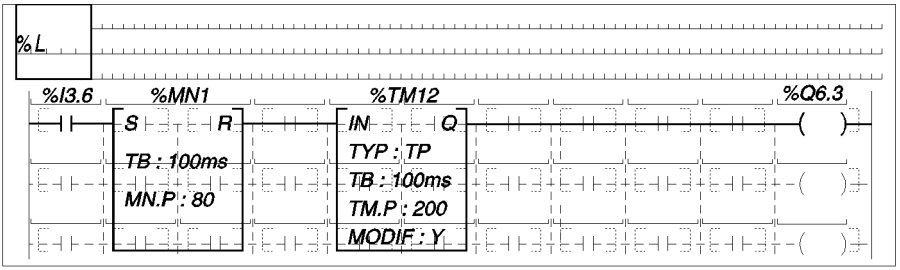
Blok może maksymalnie zajmować 16 linii.

Uwagi:

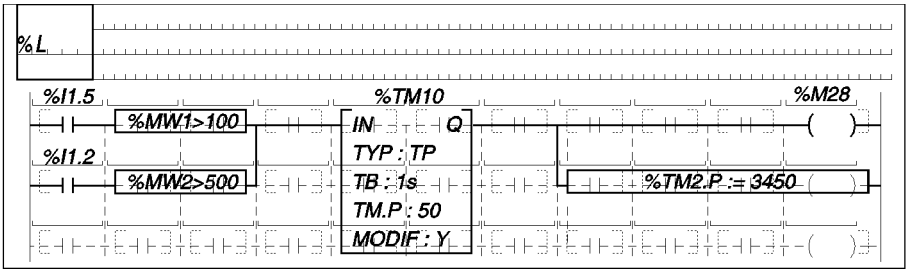
- Blok DFB musi mieć podłączone co najmniej jedno wejście logiczne.
- Cyfrowe wejścia, wyjścia lub I/O bloku nie są podłączane. Obiekty umieszczone na przeciwko danego pinu są z nim powiązane.
- Nie podłączone wejścia bloku DFB zachowują wcześniejsze wartości lub wartości początkowe jeśli blok nigdy nie był wywołany z danym wejściem odblokowanym lub podłączonym.

Bloki funkcyjne można łączyć w kaskady
Tak, jak dla elementów graficznych można tworzyć kombinacje bloków funkcyjnych.

Łączenie szeregowe bloków funkcyjnych:

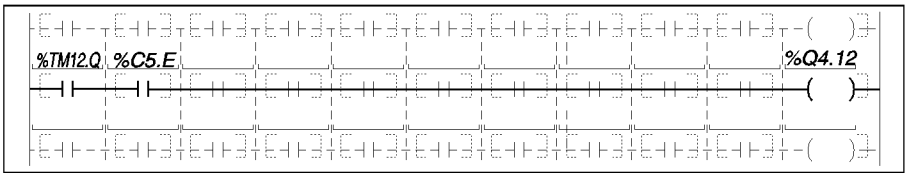


Mieszane łączenie bloków funkcyjnych i operacyjnych



Inne cechy charakterystyczne bloków funkcyjnyh
Niezależnie od zastosowanego bloku funkcyjnego, jego wejście musi być podłączone do lewej linii bazowej albo bezpośrednio, albo za pośrednictwem elementu graficznego.

- **Pozostawienie otwartych wyjść**: nie ma konieczności podłączania wyjść bloków funkcyjnych do innych elementów graficznych.
- **Testowanie stanów wyjść**: wyjścia bloków funkcyjnych są dostępne dla użytkownika w formie bitów obiektowych.



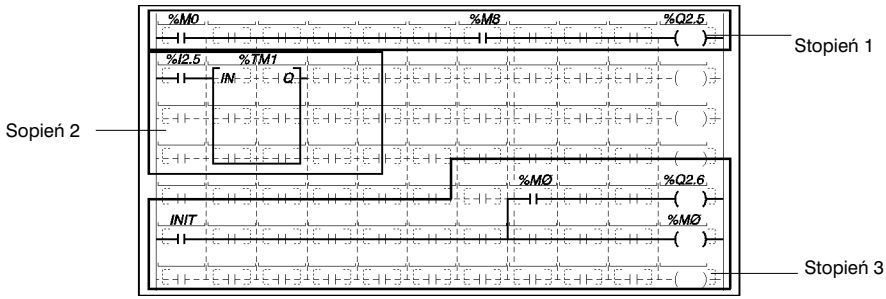
Zmienne wewnętrzne bloków i wyjść mogą być wykorzystywane w innej części programu.

2.3 Zasady rządzące wykonywaniem labelek

2.3-1 Zasady ogólne wykonywania stopni labelki

Stopnie labelki są wykonywane jeden, po drugim, przy czym każdy wykonywany jest z lewa do prawa.

Stopnie labelki zawierają elementy graficzne połączone między sobą przy pomocy linii poziomych i pionowych (poza połączeniem liniami bazowymi) ale są niezależne od innych elementów graficznych labelki (nie ma pionowych połączeń pomiędzy poszczególnymi stopniami labelki).



Jako pierwszy obliczany jest stopień labelki umieszczony w lewym górnym rogu.

Poszczególne stopnie labelki (*rung*) wykonywane są w następującym porządku: linia po linii od góry do dołu, od lewej strony do prawej.
W przypadku zastosowania pętli następuje obliczenie wpierw stopnia pośredniego (z zachowaniem opisanych powyżej zasad), po czym następuje powrót do obliczania stopnia, który zawierał stopień pośredni (*sub-rung*).

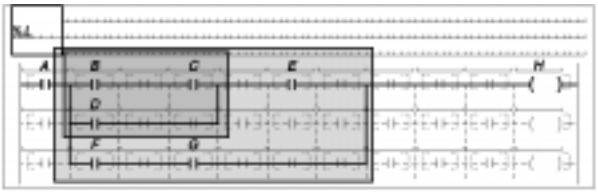
Zgodnie z takim porządkiem, system:

- oszacowuje stan logiczny każdego styku, z uwzględnieniem odczytanych na początku "przebiegu" programu bieżących wartości obiektów wewnętrznych aplikacji lub stanów wejść modułów I/O,
- wykonuje operacje przetwarzania zapisane w funkcjach, blokach funkcyjnych oraz procedurach,
- uaktualnia sprzężone bity obiektowe (wyjścia modułów I/O są aktualizowane na końcu każdego "przebiegu"),
- przechodzi do kolejnej labelki danego modułu programu (albo przeskakuje do innej labelki ->>%Li), powraca do modułu, z którego nastąpiło wywołanie <RETURN> lub zatrzymuje program <HALT>.

Uwaga:
Nie mozna stosować labelek zagnieżdżonych.

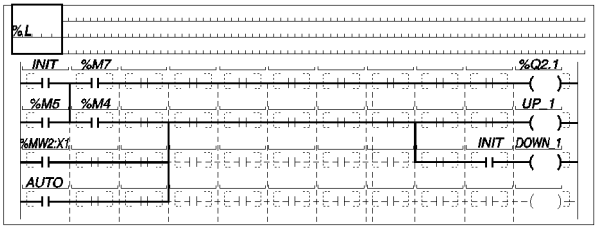
Labelka zawierająca pętlę wykonywana jest w następującym porządku:

- obliczanie labelki do miejsca, w którym zamyka się pierwsza pętla (styki A, B i C),
- obliczenie pierwszej pętli (styk D),
- kontynuacja obliczania labelki do momentu zamknięcia się drugiej pętli (styk E),
- obliczenie drugiej pętli (styki F i G),
- oszacowanie wartości cewki H.



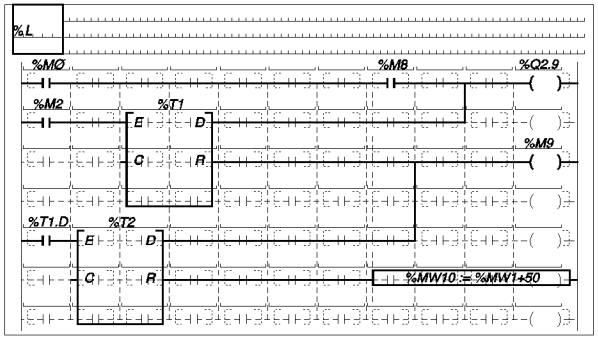
Przykład labelki logicznej
Porządek wykonywania:

- sprzężenie 1: INIT, %M5, %M7, %Q2.1,
- sprzężenie 2: %M4, %MW2:X1, AUTO, UP_1,
- sprzężenie 3: INIT, DOWN_1.



Przykład labelki zawierającej bloki
Porządek wykonywania:

- sprzężenie1: %M0, %M8, %M2, %T1, %Q2.9,
- sprzężenie 2: %T1.R, %T2, %M9,
- blok operacyjny.



3.1 Prezentacja języka

3.1-1 Wprowadzenie

Program napisany w języku LIST składa się z szeregu instrukcji wykonywanych sekwencyjnie przez sterownik.

Przykładowa instrukcja: LD %I1.0

Kod instrukcji Argument

Każda instrukcja składa się z kodu instrukcji i argumentu.

Instrukcje te oddziaływują na:

- obiekty I/O sterownika (przyciski, czujniki, przekaźniki, kontrolki, itd.),
- standardowe funkcje sterujące (zegary, liczniki, itd.),
- operacje arytmetyczne i logiczne oraz operacje związane z transferem danych,
- zmienne wewnętrzne sterownika.

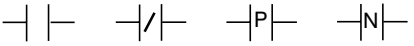
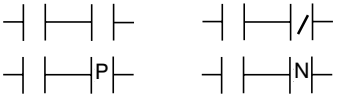
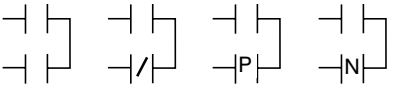
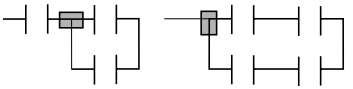
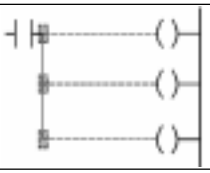
Można wyróżnić dwa rodzaje instrukcji:

- instrukcje definiujące warunki (*test instructions*), których spełnienie warunkuje wykonanie akcji, np. LD, AND, OR, itd.
- instrukcje definiujące akcje jakie ma wykonać sterownik w razie spełnienia zdefiniowanych warunków, np. ST, STN, R, itd.

3.1-2 Instrukcje

Instrukcje podstawowe

Bardziej szczegółowych informacji o instrukcjach należy szukać w części B.

Grupa	Instrukcje	Funkcje równoważne
Instrukcje warunków	• **LD**, **LDN**, **LDR**, **LDF**	
	• **AND**, **ANDN**, **ANDR**, **ANDF**	
	• **OR**, **ORN**, **ORR**, **ORF**	
	• **AND(**, **OR(** (8 poziomów nawiasów)	
	• **XOR**, **XORN**, **XORR**, **XORF** nierównoważność (*exclusive OR*)	
	• **MPS** **MRD** **MPP**	
	• **N**	Negacja
Instrukcje akcji	• **ST**, **STN**, **S**, **R**	
	• **JMP**, **JMPC**, **JMPCN**	Realizują skok (bezwarunkowy, warunkowy, gdy wynikiem jest 0 lub warunkowy, gdy wynikiem jes1) do instrukcji o podanej etykiecie zarówno do przodu, jak i do tyłu programu.
	• **SRn** **RET, RETC, RETCN**	Umożliwia wywołanie wybranej procedury (SR). Powrót z procedury do programu (bezwarunkowy, warunkowy, gdy wynikiem jest 1 i warunkowy, gdy wynikiem jest 0).
	• **END, ENDC, ENDCN** **HALT**	Koniec programu (bezwarunkowy, warunkowy, gdy wynikiem jest 1 i warunkowy, gdy wynikiem jest 0). Zatrzymanie wykonywania programu.

Instrukcje związane z blokami funkcyjnymi (patrz część B, rozdział 1.3)

Grupa	Instrukcje	Funkcje
Elementy warunków	• Bloki: Zegar *Timer* Licznik *Counter* Przerzutnik*Monostable* Rejestr *Register* Bęben *Drum controller*	Są to instrukcje sterujące działaniem wszystkich standardowych bloków funkcyjnych. Wejścia, wyjścia, I/O bloków funkcyjnych są podłączane bezpośrednio w formie ustrukturyzowanej.

Instrukcje numeryczne (patrz część B)

Grupa	Instrukcje	Funkcje
Elementy warunków	• LD[.....] AND[.....] OR[.....]	Służy do porównywania dwóch argumentów (patrz: część B, rozdział 1.4-2). Po dokonaniu porównania i spełnieniu warunku wyjście zmienia stan na 1.
	Przykład: LD[%MW10<1000]	Gdy %MW10<1000, to wyjście zmienia stan na 1.
Elementy akcji	• [.....]	Wykonanie operacji arytmetycznych, logicznych. Składnia strukturalnego języka ST (patrz cz. B).
	Przykład: [%MW10:=%MW0+100]	Wynik operacji %MW0+100 zapisywany jest w słowie wewnętrznym %MW10.

3.2 Struktura programu

3.2-1 Wiadomości ogólne

Podobnie, jak w języku *Ladder* instrukcje zestawiane są w ciągi instrukcji (odpowiadające stopniom labelek) zwane sekwencjami. Każda z sekwencji zawiera co najmniej 1 instrukcję definiowania warunków. Instrukcja ta jest przyporządkowywana (a w zasadzie jej wynik) co najmniej jednej instrukcji definiowania operacji.

Instrukcja zajmuje jedną linię.
Każda sekwencja rozpoczyna się wykrzyknikiem (generowanym automatycznie). Sekwencja może również zawierać komentarz i etykietę umożliwiającą jej identyfikację.

```
!       (*Oczekiwanie na osuszenie*)
        %L2:
        LD      %I0.1
        AND     %M10
        ST      %Q2.5
```

3.2-2 Komentarz

Komentarz umieszcza się na początku sekwencji. Może zajmować maksymalnie 3 linie (tj. 222 znaki alfanumeryczne). Jest on ograniczony z dwóch stron znakami **(*** i ***)**. Komentarz stanowi opis funkcji realizowanej przez daną sekwencję, przy czym jego stosowanie jest nieobowiązkowe.

Komentarze są wyświetlane tylko w pierwszej linii sekwencji.

Usunięcie sekwencji powoduje jednoczesne usunięcie związanego z nią komentarza.

Komentarze zapisywane są w pamięci sterownika (pamięć programu) i są dostępne dla użytkownika w każdym momencie.

3.2-3 Etykiety

Etykiety umożliwiają identyfikację sekwencji w obrębie danej części programu (tzn. program główny, procedura, itd.). Ich stosowanie jest nieobowiązkowe.

Etykiety mają następującą składnię: %Li (gdzie *i* mieści się w przedziale od 0 do 999), przy czym umieszcza się je na początku sekwencji.

Dana etykieta może być przypisana do sekwencji tylko jeden raz, w obrębie danej części programu.

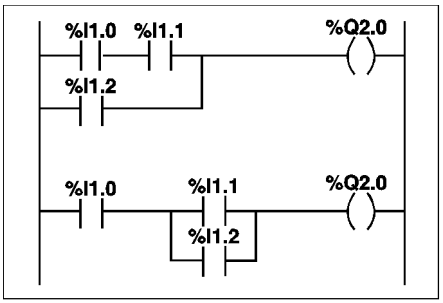
Pomimo tego, że nadawanie etykiet sekwencjom instrukcji jest nieobowiązkowe, należy pamiętać, że aby zrealizować skok do sekwencji, musi ona mieć etykietę.

System realizuje sekwencje w kolejności ich wprowadzania niezależnie od kolejności wynikającej z nadawanych numerów etykiet.

3.2-4 Stosowanie nawiasów

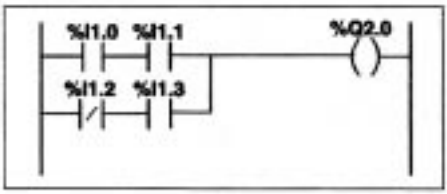
Nawiasy można stosować w przypadku instrukcji AND (koniunkcja) oraz OR (alternatywa). Ich zastosowanie upraszcza tworzenie diagramów języka *Ladder*. Nawias początkowy jest przyporządkowany instrukcji AND lub OR. Nawias końcowy zamyka zakres działania instrukcji. Każdy otwarty nawias musi być zamknięty.

Przykład: AND(




```
LD      %I1.0
AND     %I1.1
OR      %I1.2
ST      %Q2.0

LD      %I1.0
AND(    %I1.1
OR      %I1.2
)
ST      %Q2.0
```

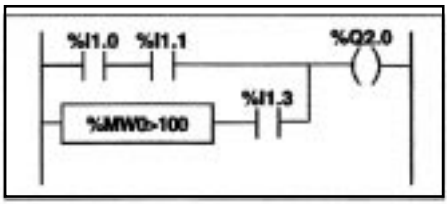
Przykład: OR(




```
LD      %I1.0
AND     %I1.1
OR(N    %I1.2
AND     %I1.3
)
ST      %Q2.0
```

Dopuszczalne są następujące modyfikacje funkcji:

- N negacja, np. AND(N lub OR(N
- F zbocze opadające (**F***alling edge*), np. AND(F lub OR(F
- R zbocze narastające (**R***ising edge*), np. AND (R lub OR (R
- [porównywanie




```
LD      %I1.0
AND     %I1.1
OR(     [%MW0>100]
AND     %I1.3
)
ST      %Q2.0
```

Zagnieżdżanie nawiasów
Dopuszcza się zagnieżdżenie maksymalnie 8 nawiasów.

Przykład

```
LD      %I1.0
AND(    %I1.1
OR(N    %I1.2
AND     %M3
)
)
ST      %Q2.0
```

Przykład

```
LD      %I1.1
AND(    %I1.2
AND     %I1.3
OR(N    %I1.5
AND     %I1.6
)
AND     %I1.4
OR(N    %I1.7
AND     %I1.8
)
)
ST      %Q2.0
```

Uwaga:
- Każdy otwarty nawias **musi** być zamknięty.
- Etykiety %Li: nie wolno ich umieszczać w wyrażeniach umieszczanych w nawiasach. Ten zakaz dotyczy również instrukcji skoku JMP oraz wywołania procedury SRi.
- W nawiasach nie wolno również umieszczać instrukcji przypisania ST, STN, S oraz R.

3.2-5 Instrukcje MPS, MRD i MPP (węzły)

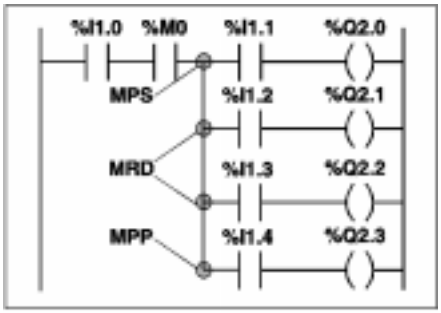
Instrukcje tego typu umożliwiają realizację połączeń ze sprzężeniami (cewkami). Wykorzystują one bufor pamięciowy (zwany stosem) o pojemności umożliwiającej zapamiętanie do 3 bitów danych typu logicznego.

Instrukcja MPS (*Memory PuSh*) powoduje zapamiętanie wyniku ostatniego testowania (ustalenia warunków), ustawia go na szczycie stosu i przesuwa inne wartości w dół stosu.

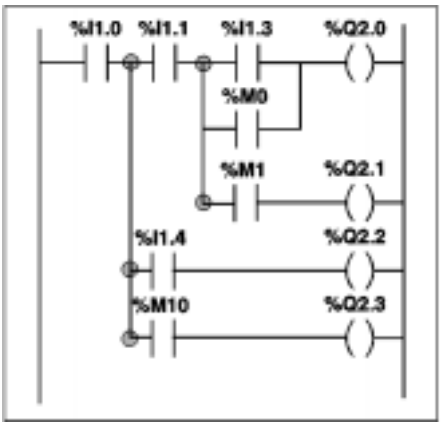
Instrukcja MRD (*Memory ReaD*) powoduje odczytanie informacji ze szczytu stosu.

Instrukcja MPP (*Memory PoP*) odczytuje i przywraca wartość zapisaną na szczycie stosu i przesuwa pozostałe w kierunku szczytu stosu.

Przykłady:




```
LD      %I1.0
AND     %M0
MPS
AND     %I1.1
ST      %Q2.0
MRD
AND     %I1.2
ST      %Q2.1
MRD
AND     %I1.3
ST      %Q2.2
MPP
AND     %I1.4
ST      %Q2.3
```



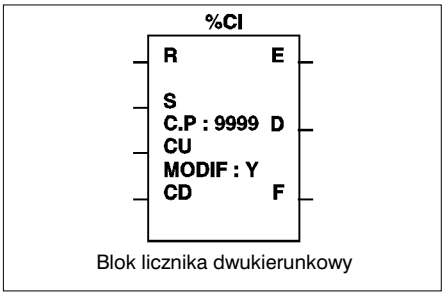

```
LD      %I1.0
MPS
AND     %I1.1
MPS
AND(    %I1.3
OR      %M0
)
ST      %Q2.0
MPP
AND     %M1
ST      %Q2.1
MRD
AND     %I1.4
ST      %Q2.2
MPP
AND     %M10
ST      %Q2.3
```

Uwaga:
Tych instrukcji nie wolno umieszczać w wyrażeniach zapisanych w nawiasach.

3.2-6 Zasady programowania predefiniowanych bloków funkcyjnych

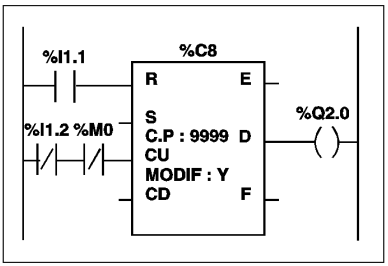
Bloki funkcyjne mogą być programowane na dwa sposoby:

- za pośrednictwem funkcji specjalnych danego bloku (np. CU %Ci). Jest to najprostszy i najbardziej bezpośredni sposób programowania.
- za pośrednictwem instrukcji strukturalnych BLK,OUT_BLK i END_BLK.



Blok licznika dwukierunkowy

Programowanie bezpośrednie



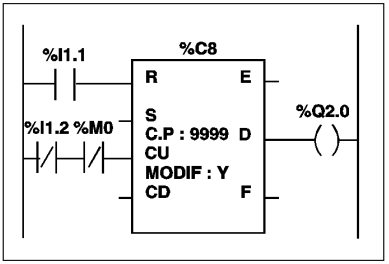

```
LD      %I1.1
R       %C8
LDN     %I1.2
ANDN    %M0
CU      %C8
LD      %C8.D
ST      %Q2.0
```

Instrukcje umożliwiają kontrolowanie stanów wejść bloków (np. CU).
Wyjścia są dostępne za pośrednictwem odpowiednich bitów (np. %C8.D).

Programowanie strukturalne

Przy takim sposobie programowania sekwencje instrukcji są ograniczone przez następujące instrukcje:

- **BLK** oznacza początek bloku,
- **OUT_BLK** umożliwia bezpośrednie połączenie z wyjściami bloku,
- **END_BLK** onacza koniec bloku.




```
BLK     %C8
LD      %I1.1
R
LDN     %I1.2
ANDN    %M0
CU
OUT_BLK
LD      D
ST      %Q2.0
END_BLK
```

Przetwarzanie wejść

Przetwarzanie wyjść

Przy programowaniu strukturalnym istnieje konieczność stosowania dodatkowych instrukcji BLK, OUT_BLK i END_BLK i z tego powodu wymaga ono większej ilości pamięci w stosunku do programowania bezpośredniego.
Programowanie takie stosuje się tam, gdzie trzeba zapewnić zgodność z TSX 07.

3.3 Zasady rządzące wykonywaniem programów w języku LIST

Programy są wykonywane sekwencyjnie, instrukcja po instrukcji.

Pierwszą instrukcją sekwencji instrukcji musi być instrukcja LD albo któraś z instrukcji bezwarunkowych (np. JMP).

Wszystkie instrukcje (z wyjątkiem instrukcji LD oraz instrukcji bezwarunkowych) wykorzystują logiczny wynik instrukcji poprzedzającej.

Przykład:

LD	%I1.1	Wynik logiczny = stan bitu %I1.1.
AND	%M0	Wynik logiczny = AND koniunkcja poprzedzającego wyniku logicznego i stanu bitu %M0.
OR	%M10	Wynik logiczny = OR alternatywa poprzedzającego wyniku logicznego i stanu bitu %M10.
ST	%Q2.0	%Q20 przyjmuje stanu poprzedzającego wyniku logicznego.

Zastosowanie nawiasów umożliwia modyfikowanie porządku, w jakim wyniki logiczne są brane pod uwagę:

Przykład:

LD	%I1.1	Wynik logiczny = stan bitu %I1.1.
AND	%M0	Wynik logiczny = AND koniunkcja poprzedzającego wyniku logicznego i stanu bitu %M0.
OR(	%M10	Wynik logiczny = stan bitu %M10.
AND	%I1.2	Wynik logiczny = AND koniunkcja poprzedzającego wyniku logicznego i stanu bitu %I1.2.
)		Wynik logiczny = OR alternatywa poprzedniego wyniku logicznego oraz wyniku logicznego instrukcji umieszczonej przed instrukcją z nawiasem.
ST	%Q2.0	%Q2.0 przyjmuje stanu poprzedzającego wyniku logicznego.

Sekwencja instrukcji może być modyfikowana poprzez zastosowanie skoku (JMP) oraz instrukcji wywołania procedury.

Przykład:

```
!       LD      %M0
        JMPC    %L10
!       LD      %I1.1
        AND     %M10
        ST      %Q2.0
!       %L10 :
        LD      %I1.3
        AND     %M20
        .....
```

Skok do labelki o etykiecie %L10 jeśli %M0=1

4.1 Prezentacja języka ST

4.1-1 Wprowadzenie

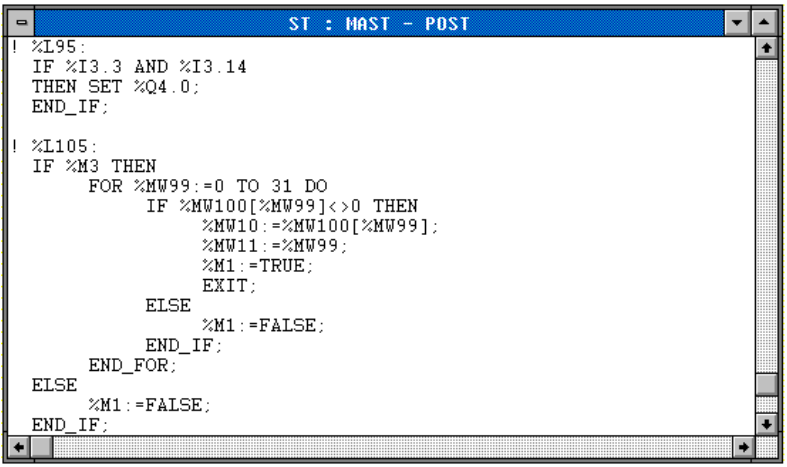
Strukturalny język tekstowy ST służy do tworzenia programów za pomocą znaków alfanumerycznych.
Ten język może być stosowany tylko do programowania sterowników TSX/PMX/PCX 57 wyposażonych w oprogramowanie **PL7 Junior** lub **PL7 Pro**. W wersji PL7 Pro istnieje dodatkowa możliwość tworzenia własnych bloków funkcyjnych typu DFB.

Zdanie (wyrażenie) języka ST składa się z części bazowej i szeregu zdań definujących program.

Podstawowe instrukcję języka ST:

- instrukcje dotyczące bitów,
- instrukcje arytmetyczne i logiczne dotyczące słów i słów podwójnej precyzji,
- instrukcje arytmetyczne dotyczące obiektów zmiennoprzecinkowych,
- numeryczne porównywanie słów, słów podwójnej precyzji i zmiennoprzecinkowych,
- numeryczna konwersja,
- instrukcje tablic bitowych, słów, słów podwójnej precyzji i zmiennoprzecinkowych,
- instrukcje dotyczące łańcuchów znaków,
- alfanumeryczne porównywanie,
- instrukcje dotyczące zarządzaniem czasem,
- instrukcje programowe,
- instrukcje sterujące,
- instrukcje standardowych bloków funkcyjnych,
- instrukcje wymiany jawnej,
- instrukcje specjalne (komunikacja, sterowanie procesem, itp.).

Przykład:



4.1-2 Instrukcje

Instrukcje dotyczące bitów

Oznaczenie	Funkcja
:=	Przypisanie bitu
OR	Alternatywa
AND	Koniunkcja
XOR	Nierównoważność
NOT	Negacja
RE	Zbocze narastające sygnału
FE	Zbocze opadające sygnału
SET	Nadanie bitowi wartości 1
RESET	Skasowanie bitu do 0

Numeryczne porównywanie słów, słów podwójnych i zmiennoprzecinkowych

Oznaczenie	Funkcja
<	Mniejsze od
>	Większe od
<=	Mniejsze, równe
>=	Większe, równe
=	Równe
<>	Różne od

Tablice bitów

Oznaczenie	Funkcja
Tablica := Tablica	Przypisanie dwu tablic
Słowo := Słowo	Przypisanie słowa do tablicy
Słowo := Tablica	Przypisanie tablicy do słowa
Tablica := Słowo podwójne	Przypisanie słowa podwójnej precyzji do tablicy
Słowo podwójne := Tablica	Przypisanie tablicy do słowa podwójnej precyzji
COPY_BIT	Kopiowanie tablicy bitów do tablicy bitów
AND_ARX	Koniunkcja dwóch tablic
OR_ARX	Alternatywa dwóch tablic
XOR_ARX	Nierównoważność między dwiema tablicami
NOT_ARX	Negacja tablicy
BIT_W	Kopiowanie tablicy bitów do tablicy słów
BIT_D	Kopiowanie tablicy bitów do tablicy słów podwójnych
W_BIT	Kopiowanie tablicy słów do tablicy bitów
D_BIT	Kopiowanie tablicy słów podwójnych do tablicy bitów
LENGTH_ARX	Obliczanie długości tablicy (w ilości elementów)

Arytmetyka słów i słów podwójnej precyzji (liczby całkowite)

Oznaczenie	Funkcja
+, -, *, /	Dodawanie, odejmowanie, mnożenie, dzielenie liczb całk.
REM	Reszta po dzieleniu liczb całkowitych
SQRT	Pierwiastek kwadratowy z liczb całkowitych
ABS	Wartość bezwzględna
INC	Zwiększanie
DEC	Zmniejszanie

Arytmetyka na liczbach zmiennoprzecinkowych

Oznaczenie	Funkcja
+, -, *, /	Dodawanie, odejmowanie, mnożenie, dzielenie
SQRT	Pierwiastek kwadratowy
ABS	Wartość bezwzględna
TRUNC	Część całkowita liczby
LOG	Logarytm dziesiętny
LN	Logarytm naturalny
EXP	Exponent naturalny
EXPT	Podnoszenie do potęgi l-b całkowitych (l. rzeczywiste)
COS	Kosinus (w radianach)
SIN	Sinus (w radianach)
TAN	Tangens (w radianach)
ACOS	Arc kosinus (wynik w przedziale od 0 do 2 π)
ASIN	Arc sinus (wynik w przedziale $-\pi/2$ do $+\pi/2$)
ATAN	Arc tangens (wynik w przedziale $-\pi/2$ do $+\pi/2$)
DEG_TO_RAD	Przeliczenie ze stopni na radiany
RAD_TO_DEG	Przeliczenie z radianów na stopnie

Instrukcje logiczne na słowach i słowach podwójnej precyzji

Oznaczenie	Funkcja
AND	Koniunkcja
OR	Alternatywa
XOR	Nierównoważność
NOT	Dopełnienie logiczne
SHL	Logiczne przesunięcie w lewo
SHR	Logiczne przesunięcie w prawo
ROL	Okrężne przesunięcie logiczne w lewo
ROR	Okrężne przesunięcie logiczne w prawo

Instrukcje programowe

Oznaczenie	Funkcja
HALT	Zatrzymanie wykonywania programu
JUMP	Skok do obiektu o podanej etykiecie
SRi	Wywołanie procedury
RETURN	Powrót z procedury do programu
MASKEVT	Maskowanie zdarzeń w sterowniku
UNMASKEVT	Zdjęcie maskowania zdarzeń w sterowniku

Instrukcje numerycznej konwersji

Oznaczenie	Funkcja
BCD_TO_INT	Konwersja BCD ♦ Kod binarny
INT_TO_BCD	Konwersja Kod binarny ♦ BCD
GRAY_TO_INT	Konwersja Kod Gray'a ♦ Kod binarny
INT_TO_REAL	Konwersja l-by całkowitej pojedynczej na zmiennoprzec.
DINT_TO_REAL	Konwersja l-by całkowitej podwójnej na zmiennoprzec.
REAL_TO_INT	Konwersja l-by zmiennoprzec. na całkowitą pojedynczą
REAL_TO_DINT	Konwersja l-by zmiennoprzec. na całkowitą podwójną
DBCD_TO_DINT	Konwersja 32-bit. l-by BCD na 32-bit. l-bę całkowitą
DINT_TO_DBCD	Konwersja 32-bit. l-by całkowitej na 32-bit. kod BCD
DBCD_TO_INT	Konwersja 32-bit. l-by BCD na 16-bit. l-bę całkowitą
INT_TO_DBCD	Konwersja 16-bit. l-by całkowitej na 32-bit. liczbę BCD
LW	Wydzielenie słowa mniej znaczącego ze słowa podw.
HW	Wydzielenie słowa bardziej znacznego ze słowa podw.
CONCATW	Sklejenie dwu pojedynczych słów

Instrukcje dotyczące tablic słów i słów podwójnych

Oznaczenie	Funkcja
Tablica := Tablica	Przypisanie dwu tablic
Tablica := Słowo	Inicjacja tablicy
+, -, *, /, REM	Operacje arytmetyczne na tablicach
+, -, *, /, REM	Operacje arytmetyczne między wyrażeniami i tablicami
SUM	Sumowanie elementów tablicy
EQUAL	Porównywanie dwu tablic
NOT	Logiczne dopełnienie tablicy
AND, OR, XOR	Operacje logiczne pomiędzy dwiema tablicami
AND, OR, XOR	Operacje logiczne między wyrażeniami a tablicami
FIND_EQW, FIND_EQD	Odszukanie pierwszego elementu równego wartości
FIND_GTW, FIND_GTD	Odszukanie pierwszego elementu > od danej wartości
FIND_LTW, FIND_LTD	Odszukanie pierwszego elementu < od danej wartości
MAX_ARW, MAX_ARD	Odszukanie wartości największej w tablicy
MIN_ARW, MIN_ARD	Odszukanie wartości najmniejszej w tablicy
OCCUR_ARW, OCCUR_ARD	Liczba wystąpień danej wartości w tablicy
SORT_ARW, SORT_ARD	Sortowanie tablicy w szyku rosnącym lub malejącym
ROL_ARW, ROL_ARD	Okrężne przesunięcie w lewo w tablicy
ROR_ARW, ROR_ARD	Okrężne przesunięcie w prawo w tablicy
FIND_EQWP,FIND_EQDP	Znalezienie 1-go elementu = wartości w danym wierszu
LENGTH_ARW, LENGTH_ARD	Obliczanie długości tablicy

Instrukcja dotyczące tablic wartości zmiennoprzecinkowych

Oznaczenie	Funkcja
Tablica := Tablica	Przypisanie dwu tablic
Tablica := Wart. zmiennoprzec.	Inicjacja tablicy
SUM_ARR	Sumowanie elementów w tablicy
EQUAL_ARR	Porównywanie dwu tablic
FIND_EQR	Znalezienie 1-go elementu równego danej wartości
FIND_GTR	Znalezienie 1-go elementu > od danej wartości
FIND_LTR	Znalezienie 1-go elementu < od danej wartości
MAX_ARR	Odszukanie największej wartości w tablicy
MIN_ARR	Odszukanie najmniejszej wartości w tablicy
OCCUR_ARR	Liczba wystąpień danej wartości w tablicy
SORT_ARR	Sortowanie tablicy w szyku malejącym lub rosnącym
ROL_ARR	Okrężne przesunięcie w lewo w tablicy
ROR_ARR	Okrężne przesunięcie w prawo w tablicy
LENGTH_ARR	Obliczenie długości tablicy

Instrukcje dotyczące łańcuchów znaków

Oznaczenie	Funkcja
STRING_TO_INT	Konwersja ASCII ♦ Kod binarny (słowo pojedyncze)
STRING_TO_DINT	Konwersja ASCII ♦ Kod binarny (słowo podwójne)
INT_TO_STRING	Konwersja Kod binarny (słowo pojedyncze) ♦ ASCII
DINT_TO_STRING	Konwersja Kod binarny (słowo podwójne) ♦ ASCII
STRING_TO_REAL	Konwersja ASCII ♦ Liczby zmiennoprzecinkowe
REAL_TO_STRING	Konwersja Liczby zmiennoprzecinkowe ♦ ASCII
<, >, <=, >=, =, <>	Alfanumeryczne porównywanie
FIND	Odszukanie zadanego ciągu znaków w łańcuchu
EQUAL_STR	Pozycja pierwszego znaku różnego od wzorca
LEN	Długość łańcucha znaków
MID	Wydzielenie zadanego ciągu z łańcucha znaków
INSERT	Wstawienie zadanego ciągu z łańcucha znaków
DELETE	Usunięcie zadanego ciągu z łańcucha znaków
CONCAT	Sklejenie dwóch łańcuchów znaków
REPLACE	Podmiana łańcucha znaków
LEFT	Początek łańcucha znaków
RIGHT	Koniec łańcucha znaków

Instrukcje związane z zarządzaniem czasem Time management

Oznaczenie	Funkcja
SCHEDULE	Zegar czasu rzeczywistego
RRTC	Odczytanie daty systemowej
WRTC	Uaktualnienie daty systemowej
PTC	Odczytanie daty i kod zatrzymania
ADD_TOD	Dodanie okresu czasu do godziny
ADD_DT	Dodanie okresu czasu do daty i godziny
DELTA_TOD	Pomiar różnicy pomiędzy godzinami
DELTA_D	Pomiar różnicy pomiędzy datami (bez godzin)
DELTA_DT	Pomiar różnicy pomiędzy datami (z godzinami)
SUB_TOD	Odjęcie okresu czasu od godziny
SUB_DT	Odjęcie okresu czasu od daty i godziny
DAY_OF_WEEK	Odczytanie bieżącego dnia tygodnia
TRANS_TIME	Konwersja czasu trwania na datę
DATE_TO_STRING	Konwersja daty na łańcuch znaków
TOD_TO_STRING	Konwersja godziny dnia na łańcuch znaków
DT_TO_STRING	Konwersja pełnej daty na łańcuch znaków
TIME_TO_STRING	Konwersja czasu trwania na łańcuch znaków

Inne instrukcje

Oznaczenie	Funkcja
WSHL_RBIT, DSHL_RBIT	Przesunięcie słowa w lewo z odzyskaniem bitów
WSHR_RBIT, DSHR_RBIT	Przesunięcie słowa w prawo, ze znakiem, z odzyskaniem bitów
WSHRZ_C, DSHRZ_C	Przesunięcie słowa w prawo, wstawienie 0, z odzyskaniem bitów
SCOUNT	Liczenie góra/dół z sygnalizacją osiągnięcia granicy
ROLW,ROLD	Okrężna zmiana w lewo
RORW,RORD	Okrężna zmiana w prawo

Instrukcje dotyczące opóźnienia czasowego

Oznaczenie	Funkcja
FTON	Uaktywnienie opóźnienia
FTOF	Zablokowanie opóźnienia
FTP	Generowanie impulsu
FPULSOR	Generator przebiegu prostokątnego

Wszystkie funkcje i instrukcje są szczegółowo opisane w części B niniejszej instrukcji. Uwaga ta dotyczy również standardowych bloków funkcyjnych.

Instrukcje i funkcje związane z jawną wymianą danych oraz pewne aplikacje są opisane w instrukcji "Funkcje specjalne - instrukcja użytkownika" (*Application-specific functions installation manual*).

Struktury sterujące funkcji i instrukcji opisano w rozdziale 4.2.-5 części A.

4.2 Struktura programu

4.2-1 Wprowadzenie

Program napisany w języku ST jest zorganizowany w zdania (wyrażenia). Każde wyrażenie składa się z następujących elementów:

- etykieta,
- komentarz,
- instrukcje.

Każdy z tych elementów ma charakter opcjonalny, w wyniku czego można mieć wyrażenie puste, wyrażenie składające się jedynie z komentarza, czy etykiety. Każde wyrażenie rozpoczyna wykrzyknik *!* (generowany automatycznie).

Przykład:

```
!     %L2 :    (* To jest wyrazenie zawierajace etykiete i komentarz *)
               SET %M0; %MW4 := %MW2 + %MW9;
               (* i kilka instrukcji *)
               %MF12 := SQRT (%MF14);
```

4.2-2 Komentarz

Komentarz jest z dwóch stron ograniczony znakami (* i *). Można go wstawiać w dowolnym miejscu wyrażenia, przy czym nie ma obostrzenia co do ilości komentarzy w wyrażeniu. Rolą komentarza jest ułatwienie interpretacji treści i przeznaczenia wyrażenia. Stosowanie komentarza jest nieobowiązkowe.

- W komentarzach można używać dowolnych znaków.
- Jeden komentarz może składać się maksymalnie z 256 znaków.
- Niedozwolone jest zagnieżdżanie komentarzy.
- Komentarz może zajmować kilka linii.

Komentarze są zapamiętywane w sterowniku tak, że użytkownik ma do nich dostęp w dowolnym momencie. Z tego względu komentarze **zajmują pamięć programu**.

4.2-3 Etykieta

Etykieta umożliwia identyfikację wyrażenia w obrębie autonomicznej części programu (program główny, procedura, itp.). Stosowanie etykiet jest nieobowiązkowe.

Etykietę definiuje się przy pomocy następującej składni: %Li, gdzie *i* mieści się w przedziale od 0 do 999. Umieszcza się ją na początku wyrażenia. Daną etykietę można przypisać tylko raz w obrębie danej części programu (procedura, program główny, moduł programowy).

Choć stosowanie etykiet nie jest obowiązkowe, to aby możliwe było zrealizowanie skoku do określonego wyrażenia musi ono posiadać etykietę.

Etykiety mogą być przyporządkowywane w dowolnej kolejności - system podczas realizacji wyrażeń wykonuje je w takiej kolejności, w jakiej zostały wprowadzone.

4.2-4 Instrukcje

Program tworzy się przy użyciu instrukcji. Wyrażenie języka ST może zawierać kilka instrukcji z tym, że każda instrukcja winna kończyć się znakiem ';'.

4.2-5 Struktury sterujące

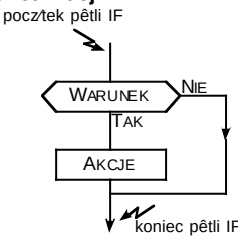
Można wyróżnić cztery struktury, za pośrenictwem których realizuje sie sterowanie:

- pętle warunkowe IF,
- pętle iteracyjne WHILE oraz REPEAT,
- operacje wykonywane w cyklu FOR.

Każda struktura sterująca jest zamknięta pomiędzy dwoma słowami kluczowymi, przy czym dana struktura musi mieć koniec w tym samym wyrażeniu, w którym ma początek. Istnieje możliwość zagnieżdżania struktur (jedna w drugiej) bez względu na ich typ. Przed i po strukturach sterujących można umieszczać inne instrukcje.

Pętla warunkowa IF ... END_IF;

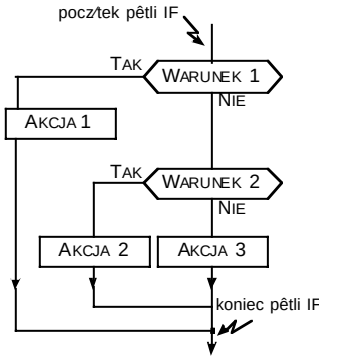
Forma podstawowa (pętla jest wykonywana jeśli warunek jest prawdą).

Składnia	**Sposób realizacji**
IF *warunek* **THEN** *akcja* ; **END_IF;**	

Przykład:

Format ogólny

Składnia	Sposób realizacji

IF *warunek 1* **THEN**
 	
akcja 1;
 	
ELSIF *warunek 2* **THEN**
 	
akcja 2;
 	
ELSE
 	
akcja 3;
 	
END_IF;
	

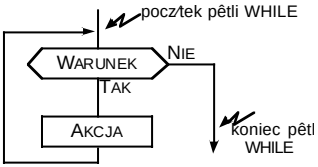
Przykład:

```
ST : MAST - POST
IF %M0 AND %M5 THEN
      RESET %M0;
      INC %MW87;
ELSIF %M0 OR %M7 THEN
      %MW12:=%MW120+1;
      SET %M9;
ELSE
      %MW12:=0;
      %MW87:=0;
      RESET %M9;
END_IF;
```

- Warunki mogą być wielorakie.
- Akcja reprezentowana jest przez listę instrukcji.
- Można zagnieżdżać pętle warunkowe "IF".
- Nie ma ograniczenia odnośnie ilości funkcji ELSIF.
- Może być tylko jedna część ELSE.

Warunkowa pętla iteracyjna WHILE ... END_WHILE;

W pętli iteracyjnej akcja jest powtarzana tak długo, jak długo spełniony jest warunek.

Składnia	**Sposób realizacji**
WHILE *warunek* **DO** *akcja*; **END_WHILE;**	

Przykład:

- Warunki mogą być wielorakie.
- Akcja reprezentowana jest przez listę instrukcji.
- Przed wykonaniem akcji następuje sprawdzenie warunku. W momencie, w którym warunek zostanie spełniony (wynik sprawdzenia - fałsz) nastąpi przerwanie wykonywania akcji.
- Można zagnieżdżać kilka pętli iteracyjnych WHILE.

Warunkowa pętla iteracyjna REPEAT ... END_REPEAT;

Akcja jest powtarzana dopóki nie zostanie spełniony warunek.

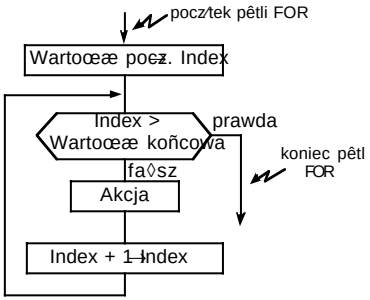
Składnia	**Sposób realizacji**
REPEAT *akcja*; **UNTIL** *warunek* **END_REPEAT;**	

Przykład:

- Warunki mogą być wielorakie.
- Akcja reprezentowana jest przez listę instrukcji.
- Sprawdzenie warunku następuje po wykonaniu akcji. Po spełnieniu warunku (wynik sprawdzenia - fałsz) nastąpi ostatnie wykonanie akcji.
- Można zagnieżdżać kilka pętli iteracyjnych REPEAT.

Pętla cykliczna FOR ... END_FOR;

Ta instrukcja powoduje wykonanie operacji określoną ilość razy (po każdym wykonaniu pętli następuje zwiększenie indeksu o 1).

Składnia	**Sposób realizacji**
FOR *indeks* := *wart. początk.* **TO** *wart.końc.* **DO** *akcja*; **END_FOR;**	

Przykład:

- Po przekroczeniu przez indeks wartości końcowej (indeks > wartość końcowa) wykonywanie pętli jest realizowane do instrukcji END_FOR.
- Indeks jest zwiększany automatycznie tak, że użytkownik nie musi go uaktualniać.
- Akcja reprezentowana jest przez listę instrukcji.
- Wartość początkowa i końcowa muszą być wyrażeniami numerycznymi typu słowo.
- Indeks musi być obiektem typu słowo, dostępnym do odczytu.
- Można zagnieździć kilka pętli FOR.

Instrukcja EXIT

- Słowo kluczowe EXIT umożliwia zatrzymanie wykonywania pętli i przeskoczenie do instrukcji następującej po słowie kluczowym zamykającym pętlę.
- Instrukcja ta może być użyta tylko w przypadku pętli typu WHILE, REPEAT i FOR.
- Instrukcja wyskoczenia z pętli (EXIT) dotyczy najbliższego zamknięcia pętli, t.j. nie powoduje ona zakończenia wszystkich pętli wewnątrz, których się ona znalazła.

Przykład:

W tym przykładzie słowo kluczowe EXIT powoduje zatrzymanie pętli REPEAT, lecz nie przerywa wykonywania pętli WHILE.

4.3 Zasady rządzące wykonywaniem programu w jęzku ST

Program napisany w języku strukturalnym ST jest wykonywany sekwencyjnie, instrukcja po instrukcji, z uwzględnieniem struktur sterujących (pętli).

Dla wyrażeń arytmetycznych lub logicznych zawierających kilka operatorów zdefiniowano zasady określania priorytetu.

Priorytety poszczególnych operatorów

W tabeli poniżej zawarto zasady określania priorytetu wyrażeń.

Operator	Symbol	Priorytet
Nawias	Wyrażenie	Najwyższy
Dopełnienie logiczne Inwersja - przy argumencie + przy argumencie	NOT NOT - +	
Mnożenie Dzielenie Modulo (reszta)	* / REM	
Dodawanie Odejmowanie	+ -	
Porównywanie	<, >, <=, >=	
Przyrównywanie Nierówność	= <>	
Koniunkcja logiczna Koniunkcja boolowska	AND AND	
Nierównoważność logiczna Nierównoważność boolowska	XOR XOR	
Alternatywa logiczna Alternatywa boolowska	OR OR	Najniższy

Przykład:

```
NOT %MW3 * 25 AND %MW10 + %MW12
```

W tym przykładzie, wpierw wykonywana jest negacja NOT na słowie %MW3, potem jej wynik jest mnożony przez 25. Obliczana jest suma %MW10 i %MW12, potem określana jest logiczna koniunkcja AND pomiędzy wynikiem mnożenia (lewa strona), a wynikiem dodawania (prawa strona).

Jeżeli występuje konflikt pomiędzy operatorami o takim samym priorytecie, to następuje wykonanie tej operacji, która jest pierwsza (licząc od lewej strony).

Przykład:

%MW34 * 2 REM 6

W tym przykładzie %MW34 jest wpierw mnożone przez 2, a wynik tego działania jest wykorzystywany do obliczenia modulo (reszty).

Zastosowanie nawiasów

Nawiasy służą do modyfikowania kolejności wykonywania działań, czyli np. w celu nadania wyższego priorytetu dodawaniu w stosunku do mnożenia.

Przykład:

(%MW10 + %MW11) * %MW12

W tym przypadku dodawanie zostane wykonane przed mnożeniem.

Nawiasy można zagnieżdżać, przy czym nie ma ograniczenia co do ich liczby.

Dzięki zastosowaniu nawiasów można zapobiec niewłaściwej interpretacji programu.

Przykład:

NOT %MW2 <> %MW4 + %MW6

Przy zastosowaniu określonych wcześniej priorytetów interpretacja jest następująca:

((NOT %MW2) <> (%MW4 + %MW6))

Natomiast użytkownik mógłby chcieć przeprowadzić następującą operację:

NOT (%MW2 <> (%MW4 + %MW6))

Powyższy przykład obrazuje w jaki sposób można zastosować nawiasy w celu uczynienia programu bardziej przejrzystym.

Konwersje niejawne (implicit conversion)

Konwersje niejawne (czyli niezależne od woli użytkownika) dotyczą słów i słów podwójnej precyzji. Taka konwersja dokonuje się w przypadku operatorów używanych w wyrażeniach arytmetycznych, w przypadku operacji porównywania i przypisywania.

Dla instrukcji zapisywanych w formie : <argument 1> <operator> <argument 2>, możliwe są następujące rodzaje konwersji:

Argument 1 typu:	Argument 2 typu:	Konwersja argumentu 1	Konwersja argumentu 2	Operacja typu:
Słowo	Słowo	Nie	Nie	Słowo
Słowo	Słowo podw.	Słowo podw.	Nie	Słowo podw.
Słowo podw.	Słowo	Nie	Słowo podw.	Słowo podw.
Słowo podw.	Słowo podw.	Nie	Nie	Słowo podw.

Dla operacji przypisania <lewy argument> := <prawy argument>, argument lewy wymusza typ argumentu na jakim ma być wykonana operacja, co oznacza, że prawy musi być poddany konwersji (chyba, że jest tego samego typu) zgodnie z tabelą:

Typ lewego argumentu	Typ prawego argumentu	Konwersja prawego argumentu
Słowo	Słowo	Nie
Słowo	Słowo podwójne	Słowo
Słowo podwójne	Słowo	Słowo podwójne
Słowo podwójne	Słowo podwójne	Nie

Uwaga:
Każda operacja pomiędzy dwiema bezpośrednio przylegającymi do siebie wartościami jest wykonywana z wykorzystaniem podwójnej precyzji.

4.1 Wyszczególnienie

Wartości bezpośrednie *Immediate values*

Obiekty	PL7-2/3	PL7 Micro/Junior
Całkowite, dziesiętne	1234	1234
Całkowite, dwójkowe	L'10011110'	2#10011110
Całkowite, szesnastkowe	H'ABCD'	16#ABCD
Zmiennoprzecinkowe	-1.32e12 (PL7-3)	-1.32e12
Łańcuch znaków	M'aAbBcB'	'aAbBcC'

Etykiety *Labels*

Etykiety	Li i = 0 do 999	%Li i = 0 do 999

Bity

Obiekty	PL7-2/3	PL7 Micro/Junior
Bit wejścia modułu wewn. Indeksowany bit wejścia modułu wew. Bit wejścia zdalnego Indeksowany bit wejścia zdalnego	Ixy,i Ixy,i (Wj) (PL7-3) RIx,y,i (PL7-3) RIx,y,i (Wj) (PL7-3)	%Ixy.i %Ixy.i[%MWj] %I\<ścieżka>\<mod>.<kanał>
Bit wyjścia modułu wewn. Indeksowany bit wyjścia modułu wew. Bit wyjścia zdalnego Indeksowany bit wyjścia zdalnego	Oxy,i Oxy,i (Wj) (PL7-3) ROx,y,i (PL7-3) ROx,y,i (Wj) (PL7-3)	%Qxy.i %Qxy.i[%MWj] %Q\<ścieżka>\<mod>.<kanał>
Bit wewnętrznego błędu I/O • bit błędu modułu • bit błędu kanału	Ixy,S / Oxy,S	%Ixy.MOD.ERR %Ixy.i.ERR
Bit błędu I/O zdalnego • bit błędu modułu • bit błędu kanału • bit kanału wyjściowego • kasowanie bitu kanału wyjściowego	(PL7-3) RDx,y,i / ERRORx,y,i TRIPx,y,i RSTx,y,i	 %I\<ścieżka>\<mod>.MOD.ERR %I\<ścieżka>\<mod>.<kanał>.ERR
Bit wewnętrzny	Bi	%Mi
Indeksowany bit wewnętrzny	Bi(Wj) (PL7-3)	%Mi[%MWj]
Bit systemowy	SYi	%Si
Bit kroku	Xi	%Xi
Bit makrodefinicji	XMj (PL7-3)	%XMj
Bit kroku *i* makrodefinicji *j*	Xj,i (PL7-3)	%Xj.i
Bit kroku wejściowego makra *j*	Xj,I (PL7-3)	%Xj.IN
Bit kroku wyjściowego makra *j*	Xj,O (PL7-3)	%Xj.OUT
Bit *j* słowa wewnętrznego *i*	Wi,j	%MWi:Xj
Bit *j* indeksowanego słowa wewn. *i*	Wi(Wk),j(PL7-3)	%MWi[%MWk]:Xj

Bit *j* słowa stałego *i*	CWi,j	%KWi:Xj
Bit *j* indeks. słowa stałego *i*	CWi(Wk),j (PL7-3)	%KWi[%MWk]:Xj
Bit *j* rejestru *i*	I/OWxy,i,j	%IW/%QWxy.i:Xj
Bit *k* słowa wspólnego *j* stacji *i*	COMi,j,k COMXi,j,k (X = B, C, D)	%NWi.j:Xk %NXWi.j:Xk
Bit *j* słowa systemowego *i*	SWi,j	%SWi:Xj

Słowa

Obiekty	**PL7-2/3**	**PL7 Micro/Junior**
Słowo wewnętrzne pojedyncze	Wi	%MWi
Pojedyncze, indeks. słowo wewnętrzne	Wi(Wj) (PL7-3)	%MWi[%MWj]
Słowo wewnętrzne podwójne	DWi (PL7-3)	%MDi
Podwójne, indeks. słowo wewnętrzne	DWi(Wj) (PL7-3)	%MDi[%MWj]
Rzeczywiste słowo wewnętrzne		%MFi
Rzeczywiste, indeks. słowo wewnętrzne		%MFi[%MWj]
Słowo stałe pojedyncze	CWi	%KWi
Pojedyncze, indeks. słowo stałe	CWi(Wj)	%KWi[%MWj]
Podwójne słowo stałe	CDWi (PL7-3)	%KDi
Podwójne, indeks. słowo stałe	CDWi(Wj) (PL7-3)	%KDi[%MWj]
Rzeczywiste słowo stałe		%KFi
Rzeczywiste, indeks. słowo stałe		%KFi[%MWj]
Pojedyncze słowo wejściowe rejestru	IWxy,i	%IWxy.i
Podwójne słowo wejściowe rejestru		%IDxy.i
Pojedyncze słowo wyjściowe rejestru	OWxy,i	%QWxy.i
Podwójne słowo wyjściowe rejestru		%QDxy.i
Słowo wejściowe rejestru zdalnego	RIWx,y,i (PL7-3)	%IW\<ścieżka>\<mod>.<kanał>
Słowo wyjściowe rejestru zdalnego	ROWx,y,i (PL7-3)	%QW\<ścieżka>\<mod>.<kanał>
Słowo systemowe	SWi	%SWi
Słowo wspólne *j* stacji *i*	COMi,j COMXi,j (gdzie X = B, C, D)	%NW{i}j %NW{[r.]i}j r = nr sieci
Słowo statusu zdalnego modułu dyskretnego	STATUSAx,y,i (PL7-3) STATUSBx,y,i (PL7-3)	
Słowo statusu zdalnego kanału dyskretnego	STSx,y,i(PL7-3)	%IW\<ścieżka>\<mod>.<kanał>.ERR
Czas aktywności kroków *Grafcet-u*	Xi,V	%Xi.T
Czas aktywności kroku *i* makra *j*	Xj,i,V (PL7-3)	%Xj.i.T

Czas aktywności kroku wejściowego makra *j*	Xj,I,V (PL7-3)	%Xj.IN.T
Czas aktywności kroku wyjściowego makra *j*	Xj,O,V (PL7-3)	%Xj.OUT.T

Bloki funkcyjne

Obiekty	**PL7-2/3**	**PL7 Micro/Junior**
Zegar *Timer*	Ti	%Ti
• wartość nastawiona (słowo)	Ti,P	%Ti.P
• wartość bieżąca (słowo)	Ti,V	%Ti.V
• sygnalizacja pracy zegara (bit)	Ti,R	%Ti.R
• sygnalizacja końca pracy zegara (bit)	Ti,D	%Ti.D
Przerzutnik monostabilny *Monostable*	Mi	%MNi
• wartość nastawiona (słowo)	Mi,P	%MNi.P
• wartość bieżąca (słowo)	Mi,V	%MNi.V
• sygnalizacja pracy bloku (bit)	Mi,R	%MNi.R
Licznik dwukierunkowy *Up/down counter*	Ci	%Ci
• wartość nastawiona (słowo)	Ci,P	%Ci.P
• wartość bieżąca (słowo)	Ci,V	%Ci.V
• przekroczenie limitu górnego (bit)	Ci,E	%Ci.E
• osiągnięcie wartości nastawionej (bit)	Ci,D	%Ci.D
• przekroczenie limitu dolnego (bit)	Ci,F	%Ci.F
Rejestr *Register*	Ri	%Ri
• wejście (słowo)	Ri,I	%Ri.I
• wyjście (słowo)	Ri,O	%Ri.O
• rejest pełny (bit)	Ri,F	%Ri.F
• rejestr pusty (bit)	Ri,E	%Ri.E
Tekst	TXTi	brak bloku tekst.
Bęben *Drum controller*	Di (PL7-2)	%DRi
• numer aktywnego kroku (słowo)	Di,S	%DRi.S
• czas aktywności kroku bieżącego (słowo)	Di,V	%DRi.V
• 16 bitów komend (słowo)	Di,Wj	%DRi.Wj
• przetwarzanie ostatniego kroku (bit)	Di,F	%DRi.F
Szybki licznik *Fast Counter* / *Timer*	FC (PL7-2)	-
• wartość nastawiona (słowo)	FC,P	-
• wartość bieżąca (słowo)	FC,V	-
• zewnętrzne kasowanie (bit)	FC,E	-
• osiągnięcie wartości nastawionej (bit)	FC,D	-
• liczenie w toku (bit)	FC,F	-
Zegar czasu rzeczywistego *Real-time clock*	H (PL7-2)	-
• typ "WEEK" (tydzień) lub "YEAR" (rok) wybór dnia MTWTFSS (słowo)	VD	-
• początek czasu aktywności (słowo)	BGN	-
• koniec czasu aktywności (słowo)	END	-
• wartość bieżąca < limit (bit)	<	-
• wartość bieżąca = limit (bit)	=	-
• wartość bieżąca > limit (bit)	>	-

Tablice bitów i słów

Obiekty	PL7-2/3	PL7 Micro/Junior
Strumienie bitów		
• wewnętrzny łańcuch bitów	Bi[L]	%Mi:L
• łańcuch bitów wejściowych	Ixy,i[L] (PL7-3)	%Ixy.i:L
• łańcuch bitów wyjściowych	Oxy.i[L] (PL7-3)	%Qxy.i:L
• łańcuch bitów kroków *Grafcet*	Xi[L] (PL7-3)	%Xi:L
• łańcuch bitów makrodefinicji	XMi[L] (PL7-3)	
Łańcuchy znaków		%MBi:L (1) (gdzie *i* parzyste)
Tablice słów		
• tablica słów wewnętrznych	Wi[L]	%MWi:L
• tablica słów wewnętrznych indeks.	Wi(Wj)[L]	%MWi[%MWj]:L
• tablica podwójnych słów wewnętrznych	DWi[L] (PL7-3)	%MDi:L
• tablica podwójnych, indeks. słów wew.	DWi(Wj)[L] (PL7-3)	%MDi[%MWj]:L
• tablica słów stałych	CWi[L]	%KWi:L
• tablica słów stałych indeksowanych	CWi(Wj)[L]	%KWi[%MWj]:L
• tablica podwójnych słów stałych	CDWi[L] (PL7-3)	%KDi:L
• tablica podwójnych słów stałych indeks.	CDWi(Wj)[L] (PL7-3)	%KDi[%MWj]:L
• tablica obiektów rzeczywistych		%MFi:L
• tablica obiektów rzeczywistych indeks.		%MFi[%MWj]:L
• tablica rzeczywistych stałych		%KFi:L
• tablica rzeczywistych stałych indeks.		%KFi[%MWj]:L
• tablica zdalnych elementów wejściowych	RIx,y,i[L] (PL7-3)	
• tablica zdalnych elementów wyjściowych	ROx,y,i[L] (PL7-3)	
• tablica zdalnych elementów wej. indeks.	RIx,y,i(Wj)[L] (PL7-3)	
• tablica zdalnych elementów wyj. indeks.	ROx,y,i(Wj)[L] (PL7-3)	

Dodatkowe bloki funkcyjne

Obiekty	PL7-3	PL7 Micro/Junior
Dodatkowy blok funkcyjny	<OFB>i	
Element OFB	<OFB>i, <element>	
Indeksowany element OFB	<OFB>i, <element>(Wj)	
Element tablicy OFB	<OFB>i, <element>[L]	
Indeksowany element tablicy OFB	<OFB>i, <element>(Wj)[L]	

Instrukcje

Obiekty	PL7-2	PL7-3	PL7 Micro/Junior
Operacje na bitach - instukcje			
• Zaprzeczenie logiczne		NOT	NOT
• Koniunkcja AND	AND	•	AND
• Alternatywa OR	OR	+	OR
• Nierównoważność XOR	XOR		XOR
• Zbocze rosnące		RE	RE
• Zbocze opadające		FE	FE
• Nadanie wartości 1		SET	SET
• Skasowanie do 0		RESET	RESET
Operacje na słowach i słowach podwójnych - instrukcje			
• Dodawanie	+	+	+
• Odejmowanie	-	-	-
• Mnożenie	*	*	*
• Dzielenie	/	/	/
• Porównywanie	>, >=, <, <=, =, <>		>, >=, <, <=, =, <>
• Reszta z dzielenia	MOD	REM	REM
• Pierwiastek kwadratowy		SQRT	SQRT
• Wartość bezwzględna			ABS
• Koniunkcja AND	AND	AND	AND
• Alternatywa OR	OR	OR	OR
• Nierównoważność	XOR	XOR	XOR
• Logiczne dopełnienie	CPL	CPL	NOT
• Zwiększanie		INC	INC
• Zmniejszanie		DEC	DEC
• Logiczne przesunięcie w lewo		SHL	SHL
• Logiczne przesunięcie w prawo		SHR	SHR
• Okrężne przesunięcie w lewo	SLC	SLC	ROL
• Okrężne przesunięcie w prawo	SRC	SRC	ROR
Operacje zmiennoprzecinkowe (1)			
• Dodawanie		ADDF	+
• Odejmowanie		SUBF	-
• Mnożenie		MULF	*
• Dzielenie		DIVF	/
• Pierwiastek kwadratowy		SQRTF	SQRT
• Wartość bezwzględna			ABS
• Sprawdzenie równości		EQUF	=
• Porównywanie >		SUPF	>
• Porównywanie <		INFF	<
• Inne			>=, <=, <>

Instrukcje (kontynuacja)

Obiekty	PL7-2	PL7-3	PL7 Micro/Junior
Operacje na łańcuchach bajtów			
• Okrężne przesunięcie		SLCWORD	
Konwersje:			
• BCD - kod binarny	BCD	DTB	BCD_TO_INT
• Kod binarny - BCD	BIN	BTD	INT_TO_BCD
• ASCII - kod binarny	ATB	ATB	STRING_TO_INT lub STRING_TO_DINT
• Kod binarny - ASCII	BTA	BTA	INT_TO_STRING lub DINT_TO_STRING
• Kod Gray'a - kod binarny		GTB	GRAY_TO_INT
• Obiekt zmiennoprzecinkowy - całkowity	FTB	REAL_TO_INT lub	REAL_TO_DINT
• Obiekt całkowity - zmiennoprzecinkowy	FTF	INT_TO_REAL lub	DINT_TO_REAL
• BCD - obiekt zmiennoprzecinkowy		DTF	BCD_TO_REAL
• Obiekt zmiennoprzecinkowy - BCD		FTD	REAL_TO_BCD
• ASCII - obiekt zmiennoprzecinkowy		ATF	STRING_TO_REAL
• Obiekt zmiennoprzecinkowy - ASCII		FTA	REAL_TO_STRING
Operacje na tablicach			
• Operacje arytmetyczne		+, -, *, /, REM	+, -, *, /, REM
• Operacje logiczne		AND,OR,XO	RAND,OR,XOR,NOT
• Dodawanie słów w tablicy		+	SUM
• Wyszukiwanie 1-go słowa różniącego tablice		EQUAL	EQUAL
• Wyszukiwanie 1-go jednakowego słowa		SEARCH	FIND_EQU
Instrukcje programowe			
• Skok		JUMP Li	JUMP %Li
• Wywołanie procedury			CALL SRi SRi
• Powrót z procedury		RET	RETURN
• Zatrzymanie aplikacji		HALT	HALT
• Pętla warunkowa		IF/THEN/ELSE	IF/THEN/ELSE/END_IF
• Pętla iteracyjna		WHILE/DO	WHILE/DO/END_WHILE
Przerwania			
• Testowanie (odczytywanie stanów)		READINT	
• Maskowanie		MASKINT	MASKEVT
• Zdejmowanie maski		DMASKINT	UNMASKEVT
• Potwierdzenie		ACKINT	
• Generowanie znaku IT dla modułu		SETIT	
Operacje wymiany jawnej I/O			
• Odczytywanie stanów wejść dyskretnych		READBIT	
• Zapisywanie stanów wyjść dyskretnych		WRITEBIT	
• Odczytywanie rejestrów		READREG	
• Zapisywanie rejestrów		WRITEREG	
• Odczytywanie słów		READEXT	
• Zapisywanie słów		WRITEEXT	

Instrukcje (kontynuacja)

Obiekty	**PL7-3**	**PL7 Micro/Junior**
Operacje na blokach funkcyjnych		
• Nastawianie	PRESET Ti / Ci	PRESET %Ti / %Ci
• Uruchomienie	START Ti / Mi	START %Ti / %MNi •
Uaktywnienie zadań	START CTRLi	
• Kasowanie	RESET Ci / Ri / TXTi	RESET %Ci / %Ri
• Dezaktywacja zadań	RESET CTRLi	
• Zliczanie	UP Ci	UP %Ci
• Odliczanie	DOWN Ci	DOWN %Ci
• Zapisanie w rejestrze	PUT Ri	PUT %Ri
• Odczytanie z rejestru	GET Ri	GET %Ri
• Odebranie komunikatu	INPUT TXTi	
• Nadawanie komunikatu	OUTPUT TXTi	
• Nadanie/odbiór komunikatu	EXCHG TXTi	
• Wykonanie OFB	EXEC <OFBi>	
• Odczytywanie telegramów	READTLG	

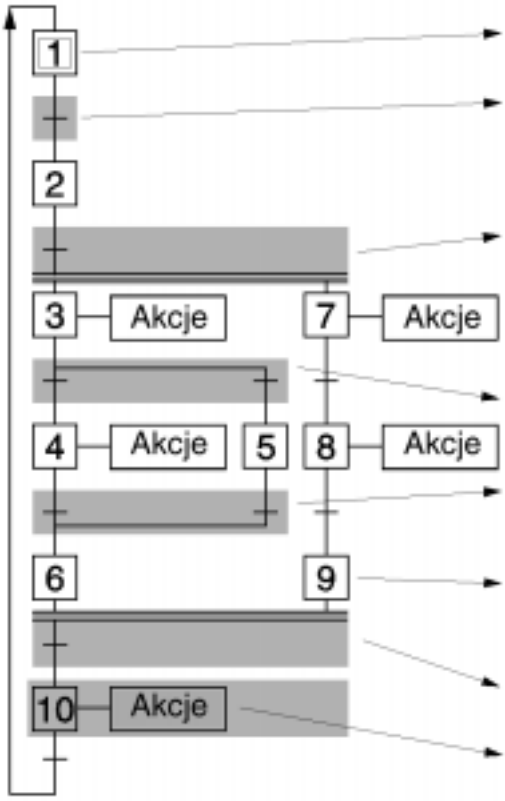
Separatory

Obiekty	**PL7-2/3**	**PL7 Micro/Junior**
Przypisywanie	->	:=
Lewy margines dla indeksowania	(	[
Prawy margines dla indeksowania	)	]
Długość tablicy	[length]	:length

5.1 Prezentacja języka Grafcet

5.1-1 Wprowadzenie

Język *Grafcet* jest zgodny z językiem SFC (*Sequential Function Chart*) opisanym w normie IEC 1131-3.
Język ten stosowany jest do odwzorowywania działania sekwencyjnego systemu sterującego w formie graficznej.
Opis graficzny działania systemu sterującego oraz występujących w trakcie sterowania sytuacji realizuje się za pomocą prostych symboli graficznych:



Krok inicjujący: definiuje warunki początkowe w sterowniku.

Bramka: przypisane **warunki przejścia** określają jakie warunki należy spełnić, by przejść do następnego elementu.

Jednoczesne uaktywnienie kroków 3 i 7 (rozdzielenie koniunkcji). Kroki 3, 4, 5, 6 oraz 7, 8, 9 tworzą dwie sekwencje traktowane jak sekwencje jednoczesne.

Wybór sekwencji (alternatywa - rozdzielenie) przejście od kroku 3 do kroku 4 lub 5.

Koniec wyboru sekwencji (zamknięcie alternatywy) przejście od kroku 4 lub 5 do kroku 6.

Krok końcowy sekwencji: umożliwia zsynchronizowanie sekwencji wykonywanych jednocześnie.

Jednoczesne zakończenie kroków 6 i 9 (zamknięcie koniunkcji).

Krok: przyporządkowane **akcje** wykonywane są tylko gdy krok jest aktywny.

Bramki i łączniki reprezentują w formie symbolicznej dopuszczalne drogi realizacji **aktywnych kroków**.
Akcje przyporządkowane krokom odzwierciedlają, ogólnie rzecz ujmując, "co ma być wykonane" gdy krok się uaktywnia. W szczególności opisują one rozkazy, jakie mają być wysłane do części wykonawczej systemu (wykonującego zautomatyzowany proces) lub innego zautomatyzowanego systemu. Zdefiniowanie, w dowolnym momencie, kroków aktywnych rzutuje na sytuację całego diagramu *Grafcet-u*.

5.1-2 Analiza procesu sterowania: zastosowanie makrodefinicji

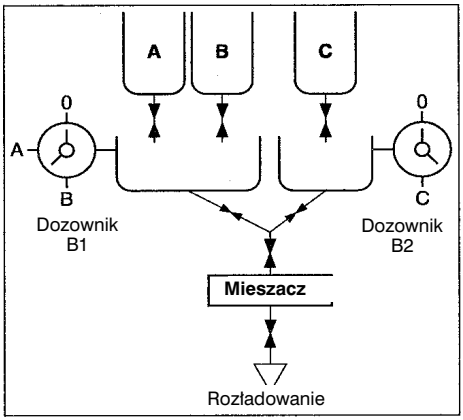
- **Charakterystyka procesu**

System umożliwia mieszanie trzech składników: A, B, C.

Składniki A i B są ważone i mierzone łącznie w dozowniku B1. Trzeci składnik C jest mierzony i ważony w dozowniku B2.

Konsystencję produktu finalnego uzyskuje się poprzez mieszanie tych składników w mieszaczu przez czas ustawiony przez operatora.

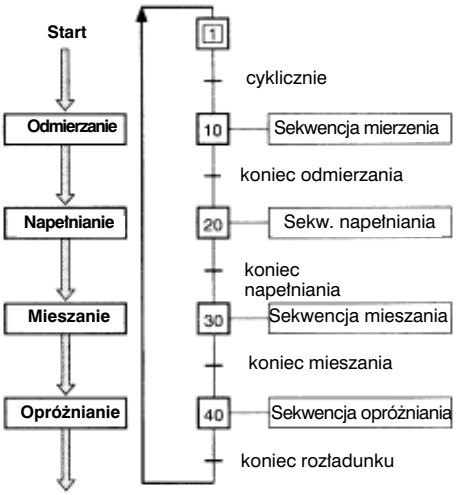
Po odliczeniu nastawionego czasu i uzyskaniu zezwolenia z zewnątrz następuje zrzucenie produktu.



- **Analiza funkcjonalna**

Opisany powyżej proces można podzielić na 4 główne sekwencje:
- odmierzanie 3 składników,
- napełnianie mieszacza,
- mieszanie 3 składników,
- opróżnienie mieszacza.

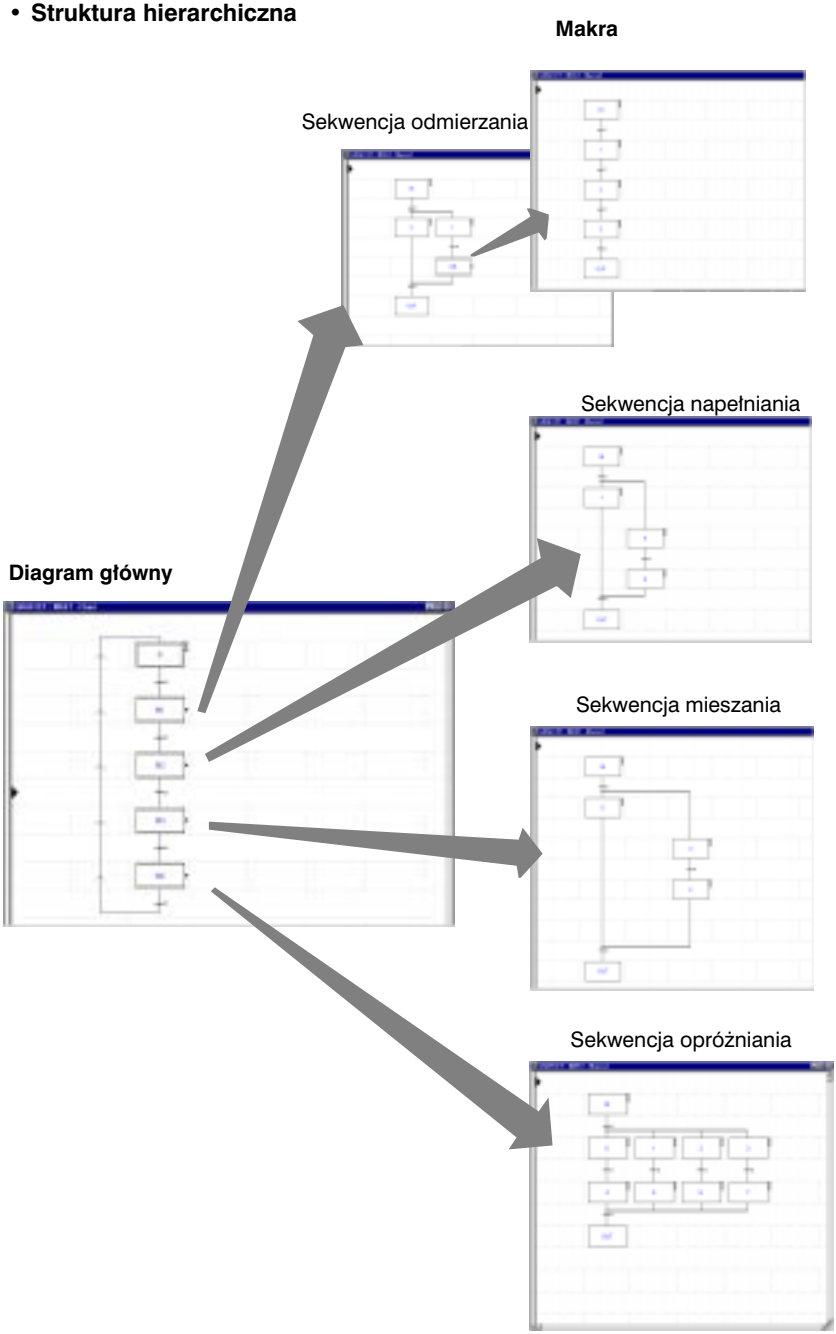
Zamieszczony obok diagram przedstawia kolejność poszczególnych operacji (analiza wstępna - pierwszy stopień).
Szczegółowej analizy każdej sekwencji dokonuje się przy pomocy diagramów drugiego lub trzeciego stopnia, które tworzy się aż do momentu osiągnięcia poziomu elementarnego tak, żeby aplikacja była opisana w całości.



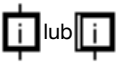
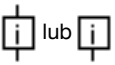
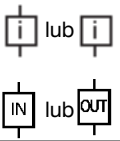
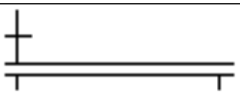
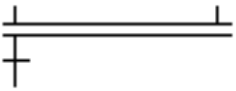
- **Makra**

Diagram pierwszego poziomu przedstawiający ciąg sekwencji pomaga uprościć strukturę części sterującej systemu. Każdej sekwencji przypisany jest specyficzny symbol tzw. makro.
Ideą zastosowania makr jest umożliwienie analizy hierarchicznej systemu. Każdy poziom analizy może być uzupełniany i modyfikowany niezależnie, bez wpływu na pozostałe. Makra można stosować tylko dla sterowników TSX 57.

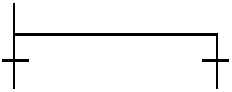
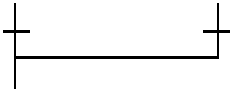
- **Struktura hierarchiczna**



5.2 Symbole graficzne języka Grafcet

Oznaczenie	Symbol	Funkcja
Krok inicjujący		Oznacza kroki aktywne na początku cyklu po inicjacji lub zimnym starcie.
Krok pojedynczy		Oznacza, że system sterujący jest stabilny. Maksymalna liczba kroków jakie można skonfigurować: - 1 do 96 dla TSX 37-10, - 1 do 128 dla TSX 37-20, - 1 do 250 dla TSX 57. Maksymalna liczba jednocześnie aktywnych kroków jest konfigurowana.
Makro		Oznacza makro, czyli zestaw kroków i bramek. Maksymalna liczba makr jakie można skonfigurować wynosi od 0 do 63 (dotyczy tylko TSX 57).
Kroki makra (1)		Oznaczenie kroków makra. Maksymalna liczba kroków makra jest definiowana i wynosi od 0 do 250 (dotyczy tylko sterowników TSX 57). W danym makrze może być tylko jeden krok IN i jeden krok OUT.
Bramka		Bramka realizuje przejście od jednego kroku, do drugiego. Związany z bramką warunek służy do określania warunku logicznego niezbędnego do przejścia do następnego kroku. Maksymalna liczba bramek wynosi 1024. Liczba ta nie podlega konfigurowaniu. Konfiguruje się natomiast maksymalną liczbę bramek jednocześnie otwartych.
Początek koniunkcji		Jest to przejście od jednego kroku do kilku kroków. Umożliwia jednoczesne uaktywnienie maksymalnie 11 kroków.
Koniec koniunkcji		Przejście od kilku jednocześnie otwartych kroków do jedego kroku. Umożliwia jednoczesne zamknięcie maksymalnie 11 kroków.

(1) Maksymalna liczba kroków (kroki diagramu głównego + kroki makr), w sekcji języka *Grafcet*, nie może przekroczyć 1024 (dla TSX 57).

Oznaczenie	**Symbol**	**Funkcja**
Początek alternatywy		Przejście od jednego kroku do kilku innych. Umożliwia wybór maksymalnie 11 sekwencji kroków.
Koniec alternatywy		Przejście od kilku kroków do jednego. Zamyka jedną z maksymalnie 11 sekwencji kroków.
Łącznik źródłowy		'n' jest numerem kroku, od którego przychodzi połączenie (krok źródłowy).
Łącznik docelowy		'n' numerem kroku, do którego ma nastąpić przejście (krok docelowy).
Łączniki bezpośrednie: • do góry • w dół • w lewo lub w prawo		Te łączniki stosowane są do wyboru sekwencji, do przeskoczenia jednego lub kilku kroków, do powtórzenia kroków (sekwencji).

5.3 Obiekty języka Grafcet

Użytkownik ma do dyspozycji bity obiektowe związane z krokami, bity systemowe języka *Grafcet*, słowa obiektowe sygnalizujące aktywność kroków i słów systemowych języka *Grafcet*.

Oznaczenie	Adres	Opis
Bity kroków (1 = krok aktywny)	**%Xi**	Status kroku *i* głównego diagramu. (*i* od 0 do *n*) (*n* zależy od procesora)
	%XMj	Stan makra *j* (*j* od 0 do 63 dla TSX/PMX/PCX 57)
	%Xj.i	Status kroku *i* makra *j*
	%Xj.IN	Status kroku wejściowego makra *j*
	%Xj.OUT	Status kroku wyjściowego makra *j*
Bity systemowe *Grafcet* (1)	**%S21**	Inicjacja diagramu
	%S22	Skasowanie wszystkich diagramów
	%S23	"Zamrożenie" diagramu
	%S24	Skasowanie makr do 0 z uwzględnieniem słów systemowych %SW22 do %SW25
	%S26	Przyjmuje wartość 1 w przypadku: - przepełnienia tablicy (kroki/bramki), - wykonywania niewłaściwego diagramu (połączenie z krokiem nie należącym do diagramu).
Słowa kroków	**%Xi.T**	Czas aktywności kroku *i* diagramu głównego
	%Xj.i.T	Czas aktywności kroku *i* makra *j*
	%Xj.IN.T	Czas aktywności kroku wejścia makra *j*
	%Xj.OUT.T	Czas aktywności kroku wyjścia makra *j*
Słowa systemowe *Grafcet-u*	**%SW20**	Słowo określające, dla bieżącego cyklu, liczbę kroków aktywnych oraz kroków,które mają być uaktywnione lub zakończone.
	%SW21	Słowo określające, dla bieżącego cyklu, liczbę bramek aktywnych oraz bramek, które mają być uaktywnione lub zamknięte
	%SW22 to %SW25	Wartość 4 słów powodujących skasowanie makr do 0, kiedy bit %S24 ma wartość 1.

(1) Szczegółów na temat wykorzystania bitów systemowych należy szukać w rozdziale 6.8-3.

Bity przypisane do kroków %Xi, makr %XMi kroków makr %Xj.I , %Xj.IN, %Xj.OUT

- Przyjmują one wartość 1, gdy krok jest aktywny.
- Bity te mogą być testowane (stanowią warunki wykonania akcji) we wszystkich zadaniach ale mogą być zapisywane jedynie podczas przetwarzania wstępnego zadania głównego MASTER (ustawianie wartości początkowych diagramu). Warunki wykonania akcji oraz same akcje programowane są w języku *Ladder*, języku instrukcji LIST lub języku strukturalnym ST.
- Bity te nie mogą być indeksowane.

Słowa związane z czasem aktywności dla kroków %Xi.T oraz makr %Xj.I , %Xj.IN i %Xj.OUT

- Wartość tych słów jest zwiększana co 100 ms (przyjmują wartości od 0 do 9999).
- Zwiększanie wartości słowa: podczas aktywności przypisanego mu kroku.
- W czasie, gdy krok nie jest aktywny zawartość słowa pozostaje zamrożona.
- Przy ponownym uaktywnieniu kroku, wartość jest kasowana i od nowa zliczana.
- Liczba słów związanych z czasem aktywności nie może być konfigurowana - każdemu krokowi przyporządkowane jest jedno słowo.
- Słowa te nie mogą być indeksowane.

5.4 Możliwości języka Grafcet

Zależą one od procesora, który ma być zaprogramowany. W tabeli poniżej zestawiono poszczególne parametry:

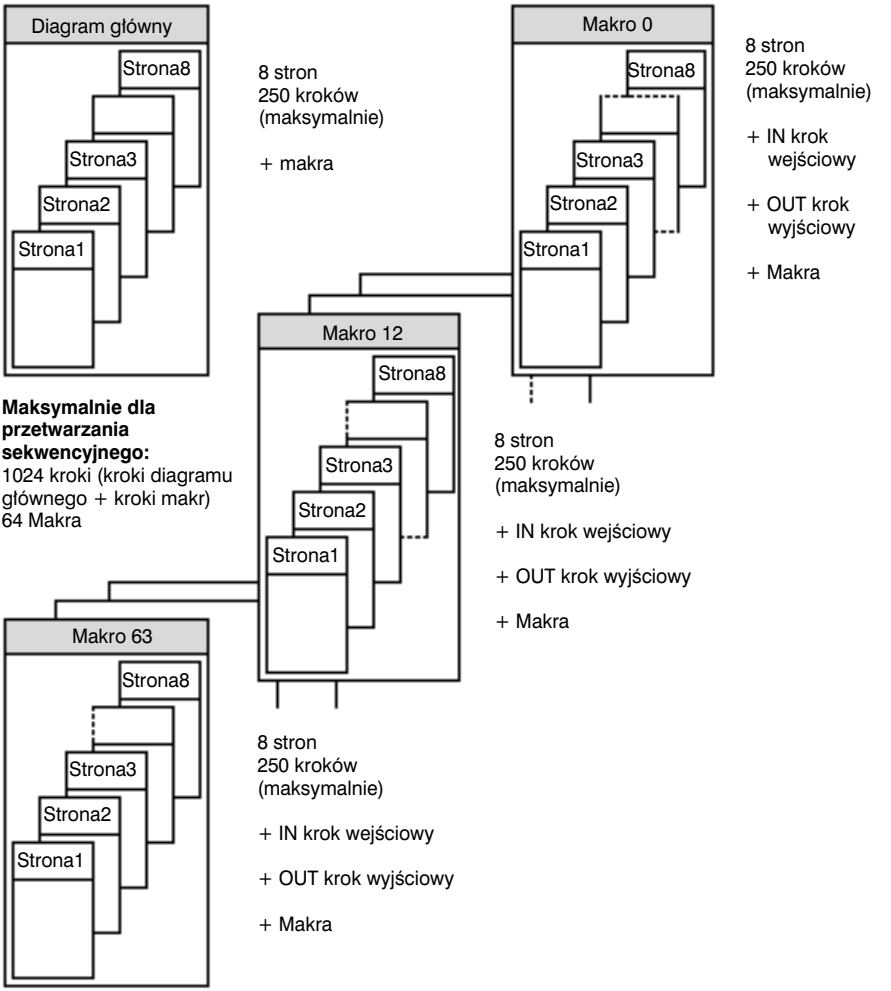
Liczba	TSX 37-10		TSX 37-20		TSX 57	
	Dom.	Max	Dom.	Max	Dom.	Max
Kroki diagramu głównego	96	96	128	128	128	250
Makra	0	0	0	0	8	64
Kroki makr	0	0	0	0	64	250
Całkowita liczba kroków	96	96	128	128	640	1024
Kroki jednocześnie aktywne	16	96	20	128	40	250
Bramki jednocz. dostępne		20	192	24	256	48400

Skrót "Dom." oznacza wartości domyślne. Skrót "Max" oznacza wartości maksymalne.

Przykład dla TSX 57:

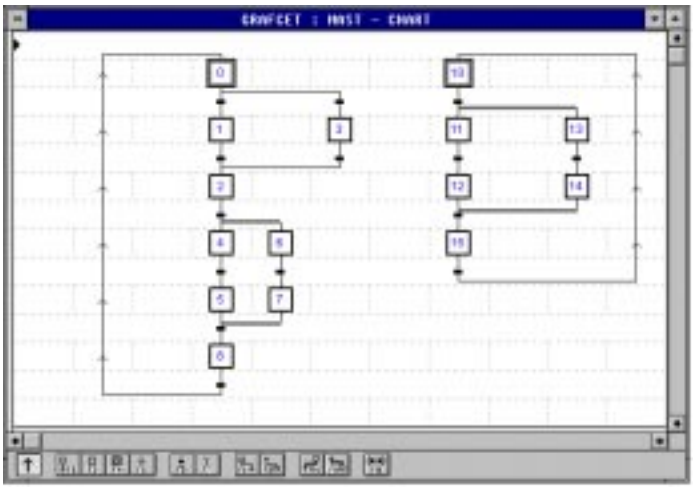
Proces przetwarzania podzielono na:
- 1 podzbiór typu: diagram główny,
- 64 podzbiory typu: makra

Te podzbiory są podzielone na strony (patrz poniżej).



5.5 Wygląd diagramu Grafcet

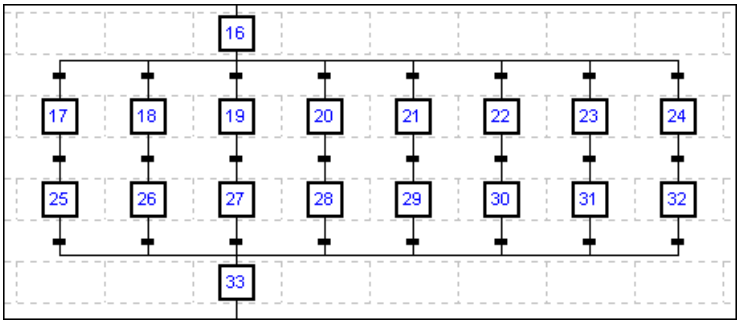
Diagram główny może być zaprogramowany na 8 stronach (strony od 0 do 7). Każda strona zawiera 14 linii i 11 kolumn, co w efekcie daje 154 komórki. W jednej komórce może być umieszczony jeden element graficzny.



Reguły obowiązujące podczas zapisywania diagramu

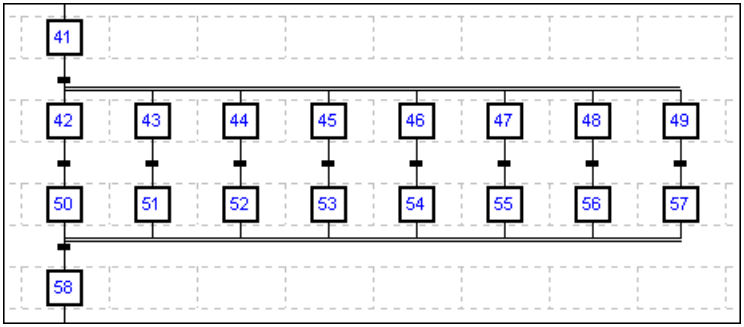
- Pierwsza linia służy do wprowadzania łączników źródłowych.
- Ostatnia linia służy do wyprowadzania łączników docelowych.
- Linie parzyste (od 2 do 12) są liniami kroków (dla kroków i łączników źródłowych).
- Linie nieparzyste (od 3 do 13) są liniami bramek (dla bramek i łączników docelowych).
- Numery kroków (od 0 do 127) definiuje się w dowolnym porządku.
- Na jednej stronie można umieścić kilka diagramów.

Wybór sekwencji i zakończenie wyboru



- Liczba przejść w górę w stosunku do końca wyboru sekwencji (zamknięcie alternatywy) i przejść w dół w stosunku do wyboru sekwencji (początek alternatywy) nie może przekraczać 11.
- Wybór sekwencji rysuje sie przy pomocy łącznika poziomego.
- Otwarta alternatywa musi, ogólnie rzecz biorąc, być zamknięta.
- W celu zabezpieczenia się przed jednoczesnym otwarciem kilku bramek należy tak dobierać warunki, żeby się one wzajemnie wykluczały.

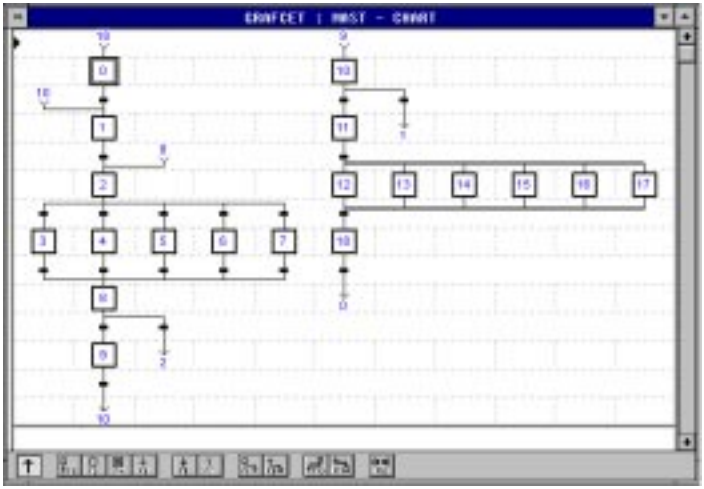
Jednoczesne uaktywnienie i zamknięcie kroków



- Liczba kroków w dół w stosunku do początku koniunkcji (jednoczesne uaktywnienie kroków) i w górę w stosunku do zamknięcia koniunkcji (jednoczesne zamknięcie kroków) nie może przekraczać 11.
- Otwarta koniunkcja musi, ogólnie rzecz ujmując, być zamknięta.
- Jednoczesna aktywacja kroków zawsze odbywa się od lewej do prawej.
- Jednoczesne zamknięcie kroków odbywa się zawsze od prawej do lewej.

Stosowanie łączników

Zastosowanie łączników źródłowych i docelowych pozwala na zapewnienie ciągłości diagramu wtedy, kiedy łącze bezpośrednie nie może być narysowane w obrębie danej strony lub dwu kolejnych stron. Tę ciągłość zapewnia zastosowanie łącznika docelowego, który występuje w parze z łącznikiem źródłowym.

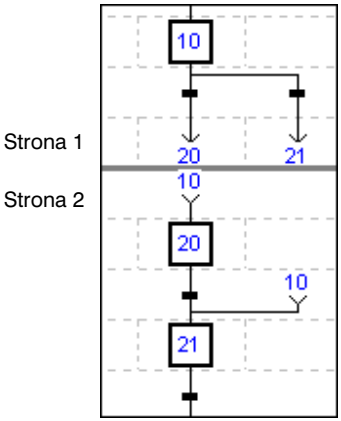


- Przy pomocy tych łączników można zrealizować pętlę (np. zapętlić krok 18 z 0).
- Za pomocą łączników można restartować sekwencję (np. zastosować połączenie kroku 10 z krokiem 1 lub kroku 8 z krokiem 2).
- Zastosowanie łączników jest niezbędne, gdy rozgałęzienie programu wykracza poza stronę (np. połączenie kroku 9 z krokiem 10).

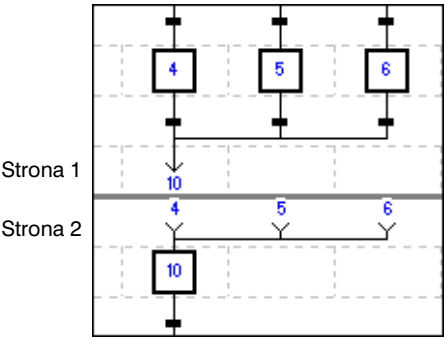
Łączniki wyboru sekwencji i zamknięcia wyboru sekwencji

- W przypadku wyboru sekwencji, bramki i łączniki docelowe muszą być umieszczone na tej samej stronie.

- W przypadku zamknięcia wyboru sekwencji, łącznik źródłowy musi być umieszczony na tej samej stronie co krok docelowy.

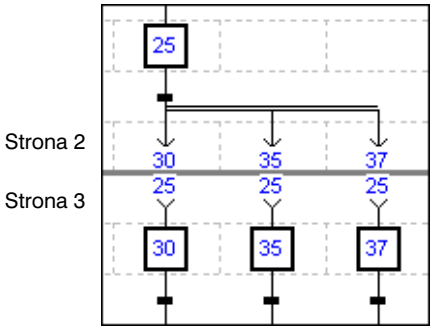


- W przypadku, gdy zamknięcie wyboru sekwencji występuje przed łącznikiem docelowym, to należy zdefiniować tyle samo łączników źródłowych ile jest kroków przed zamknięciem wyboru sekwencji.



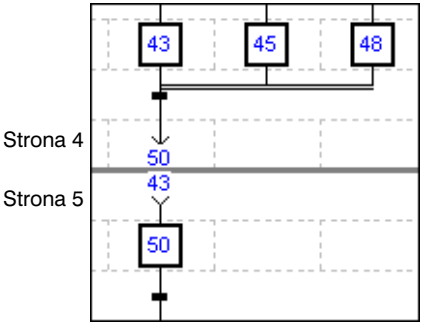
Łączniki jednoczesnego uaktywnienia kroków

- W przypadku jednoczesnego uaktywnienia kroków należy wstawić łączniki docelowe na tej samej stronie, co krok i bramka.



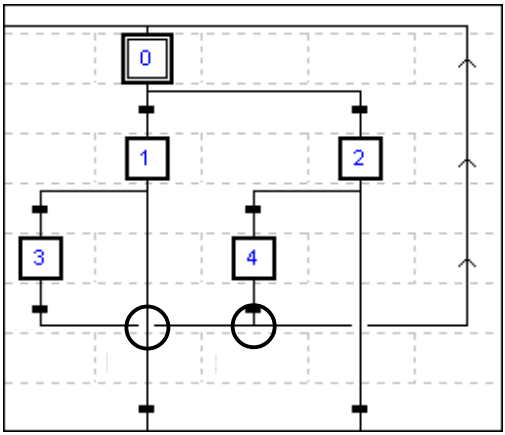
- W przypadku jednoczesnego zamknięcia, kroki i bramka zamknięcia, muszą znajdować się na tej samej stronie co łącznik docelowy.

  Kiedy kilka kroków działa na jedną bramkę, łącznik źródłowy przyjmuje numer pierwszego kroku od lewej strony licząc (patrząc w górę w stosunku do łącznika).



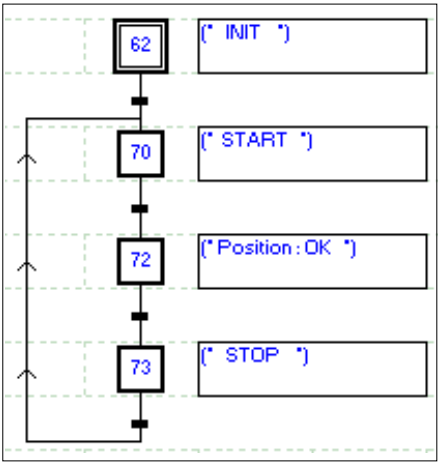
Łączniki bezpośrednie

- Łączniki bezpośrednie łączą krok z bramką albo bramkę z krokiem. Mogą być poziome lub pionowe.
- Łącza bezpośrednie mogą:
  - przecinać łączniki innych rodzajów,
  - łączyć się (tworzyć węzły) z łącznikami tego samego rodzaju.
- Łącznik bezpośredni nie może być przecinany przez jednoczesne uaktywnienie kroków oraz przez zamknięcie jednoczesnego uaktywnienia.



Komentarz

- Do każdej komórki diagramu *Grafcet* można dopisać komentarz. Komentarz zaczyna się znakiem (* i kończy *). Może on mieć maksymalnie 64 znaki.
- Komentarz zajmuje obszar dwóch kolejnych komórek i może mieć szerokość dwóch linii. Jeżeli komentarz się nie mieści w okienku, to następuje jego obcięcie. Poczas drukowania dokumentu komentarz jest drukowany w całości.
- Komentarze umieszczane na stronach diagramu *Grafcet* zapamiętywane są w sterowniku jako dane graficzne.



5.6 Makrodefinicje (makra)

Makro jest unikatowym zbiorem kroków i bramek. Makro charakteryzuje krok wejściowy i krok wyjściowy.

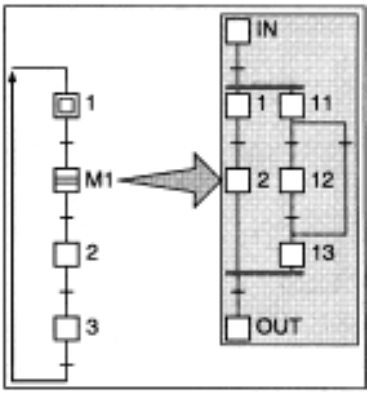
5.6-1 Wprowadzenie

Makro stanowi graficzną reprezentację sekwencji.

Makro odziela się od kroków przy pomocy dwóch linii poziomych.

W stosunku do kroku wejściowego obowiązują te same zasady co do innych kroków.

Do kroku wyjściowego nie można przypisywać żadnych akcji.

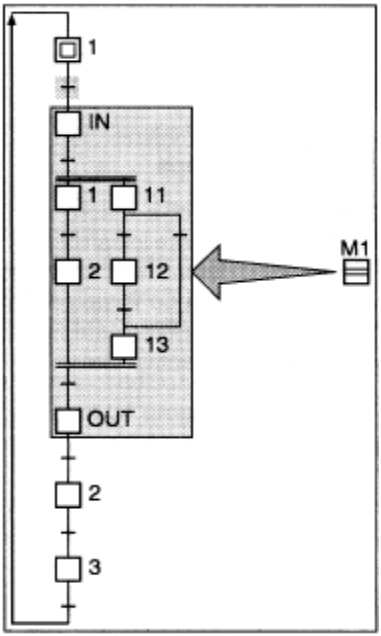


W czasie wykonywania makra obowiązują zasady opisane wcześniej.

Makro M1 uaktywnia się wraz z uaktywnieniem kroku 1, a znajdująca się za nim bramka jest otwarta (warunek jest spełniony - prawda).

Zamknięcie makra następuje, gdy uaktywnia się krok wyjściowy, a warunek otwarcia bramki M1 >2 jest prawdą.

Po zamknięciu makra następuje uaktywnienie kroku 2.



5.6-2 Charakterystyka

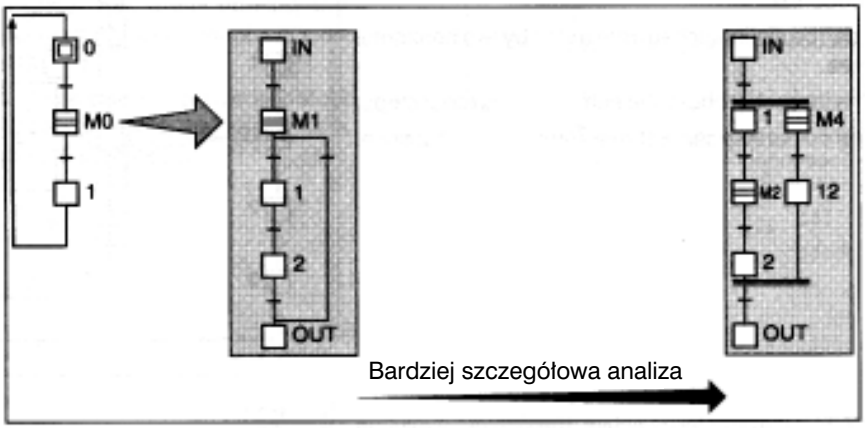
Język *Grafcet* PL7 umożliwia zaprogramowanie 64 makrodefinicji (M0 do M63).

Rozwinięcie (treść) makrodefinicji zawierające jedną lub kilka sekwencji może być programowane na 8 stronach. Może ono składać się maksymalnie z 250 kroków plus krok wejściowy IN i wyjściowy OUT.

Makrodefinicja może zawierać kilka innych makrodefinicji.

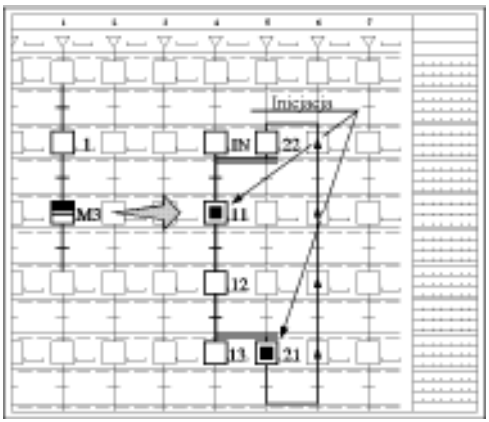
W ten sposób można budować makra do 64 poziomu.

Dzięki temu można zastosować bardziej szczegółową analizę wykonywanych operacji.



5.6-3 Kroki inicjujące

Rozwinięcie makrodefinicji może zawierać kilka kroków inicjujących. Kroki te uaktywniają się po załączeniu zasilania lub programowo - podczas inicjacji. Makro jest wtedy wyświetlane jako aktywne.



5.7 Akcje przyporządkowane krokom

Każdemu krokowi przyporządkowywuje się akcje (operacje) zaprogramowane w języku *Ladder*, języku instrukcji *LIST* lub języku strukturalnym *ST*. Akcje te są wykonywane tylko wtedy, kiedy sprzężony z nimi krok jest aktywny. PL7 posiada trzy rodzaje akcji:

- **akcje na skutek uaktywnienia**: akcje są wykonywane w momencie, gdy następuje uaktywnienie kroku,
- **akcje na skutek dezaktywacji**: akcje są wykonywane w momencie dezaktywacji kroku,
- **akcje ciągłe**: akcje wykonywane w sposób ciągły przez cały czas aktywności sprzężonego z nimi kroku.

Akcje tych trzech rodzajów można stosować dla każdego kroku.

Pojedyncza akcja składa się z kilku elementów programowych (sekwencje, wyrażenia, labelki).

Opisywanie akcji

Akcje te opisuje się w następujący sposób:

MAST - <nazwa sekcji *Grafcet*> - DIAGRAM (lub MAKROk) - STRONA n %Xi x

gdzie: x = P1 Uaktywnienie
= N1 Ciągłe wykonywanie akcji
= P0 Dezaktywacja
n = numer strony
i = numer kroku

Przykład: MAST - Malowanie - DIAGRAM - STRONA 0 %X1 P1
Akcja po uaktywnieniu kroku 1 strony 0 sekcji *Malowanie*

Zasady

- Wszystkie akcje są zapamiętywane, co w konsekwencji powoduje, że:
  - akcja uruchamiana na czas aktywności kroku Xn musi być skasowana (reset) w momencie dezaktywacji kroku Xn lub w momencie uaktywnienia kroku Xn+1,
  - akcja, której efekty wpływają na kilka kroków, przyjmuje wartość 1 przy uaktywnieniu kroku Xn i jest resetowana przy uaktywnieniu kroku Xn+m.
- Wszystkim akcjom można przypisywać warunki logiczne (wykonywanie warunkowe).
- Akcje, które są wykonywane z zastosowaniem zabezpieczeń (*safety interlocks*) umieszcza się w przetwarzaniu końcowym (*post-processing*). Jest to przetwarzanie wykonywane na końcu każdego "przebiegu" programu (patrz rozdział 5.2 "Organizacja zadania głównego MASTER").

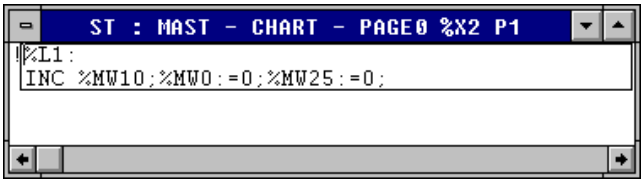
Akcje wykonywane przy uaktywnianiu i dezaktywacji kroku

Te akcje wykonywane są jednorazowo po wywołaniu. Służą do wywoływania procedur, zwiększania wartości licznika, itd.

Przykłady:

- Wywołanie procedury:

- Zwiększenie wartości słowa %MW10 i skasowanie słów %MW0 i %MW25:



Akcje wykonywane w sposób ciągły

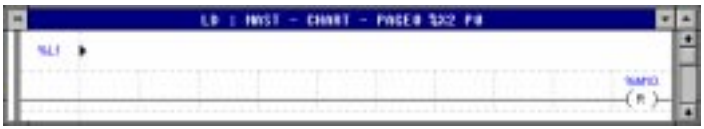
- **Akcje warunkowe**

  Przykład:

  Stan bitu %M10 zależy od wejścia %I2.5 lub bitu wewnętrznego %M9 i wejścia %I1.2.

  Tak długo, jak długo krok 2 jest aktywny, a warunki są spełnione, bit %M10 ma wartość 1. Stan bitu odczytany ostatnio zostaje zapamiętany, ponieważ sprzężone z nim akcje nie są ponownie wykonywane.

  Stąd wynika konieczność skasowania bitu %M10 do 0, na przykład przy okazji wykonania akcji powiązanej z dezaktywacją kroku.



- **Akcje warunkowe - czasowe**

  Jest to szczególny przypadek, gdy rolę warunku logicznego pełni czas. Takie rozwiązanie można zastosować przez testowanie czasu aktywności kroku.

  Przykład:

  Bit %M12 jest kontrolowany tak długo, jak długo czas aktywności kroku 3 jest mniejszy od 10 sekund (podstawa czasu: 100 ms).

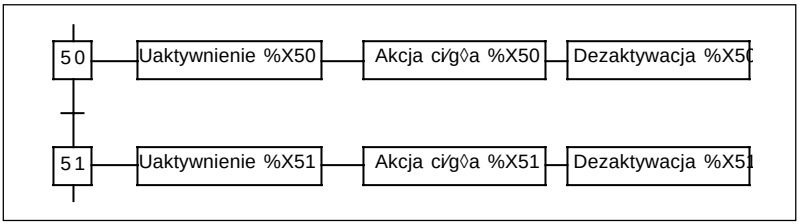
- Te akcje mogą być również wykonywane bezwarunkowo.

Kolejność wykonywania akcji

W przykładzie zamieszczonym poniżej, kolejność wykonywania akcji (w trakcie jednego "przejścia") jest następująca:

Uaktywnienie kroku 51 powoduje uaktywnienie akcji w następującym porządku:

1. akcje związane z dezaktywacją kroku 50,
2. akcje związane z uaktywnieniem kroku 51,
3. ciągłe wykonywanie akcji związanej z krokiem 51.



Dezaktywacja kroku 51 powoduje przerwanie wykonywania akcji ciągłej.

5.8 Warunki przejścia bramki

- Każdej bramce przypisuje się warunek zaprogramowany w języku *Ladder*, języku instrukcji *LIST* lub języku ST.
- Warunek przejścia bramki jest uwzględniany tylko wtedy, kiedy bramka jest aktywna.
- Warunek stanowi labelka, lista instrukcji lub wyrażenie języka ST składające się z szeregu odczytów (testów) stanów bitów i słów.
- **Warunek przejścia, któremu nie przypisano żadnego programu jest zawsze traktowany jako warunek typu fałsz.**

Opisywanie warunków przejścia

Warunki przejścia opisuje się w następujący sposób:

MAST-<nazwa sekcji *Grafcet*>-DIAGRAM (lub MAKROk) - STRONA n %X(i) --> % X(j)

gdzie: n = numer strony
i = numer kroku poprzedzającego
j = numer kroku następnego

Przykład: MAST - Malowanie - DIAGRAM - STRONA 0 %X(0) --> %X(1)
Warunek przejścia powiązany z krokiem 0 i 1 strony 0 sekcji *Malowanie* diagramu.

W przypadku jednoczesnego uaktywniania lub dezaktywacji kroków wpisuje się adres kroku w pierwszej kolumnie od lewej strony.

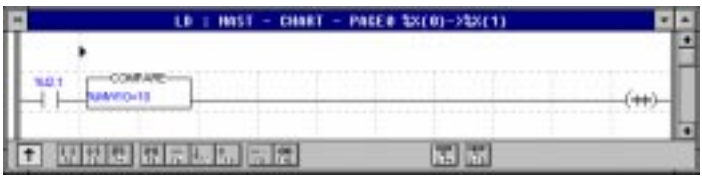
Reguły programowania w języku Ladder

Warunek przejścia przez bramkę programuje się w formie labelki składającej się co najmniej ze strefy definiowania warunków (*test zone*) i strefy akcji (*action zone*).

Struktura labelki jest taka sama jak w przypadku labelki umieszczanej w modułach programowych.

Można stosować tylko następujące elementy:

- elementy graficzne warunków wykonania akcji: styki (%Mi, %I, %Q, %TMi.D, itd.), bloki porównywania,
- elementy graficzne akcji: tylko sprzężenie (cewka) z warunkiem przejścia - pozostałe nie mają w tym przypadku żadnego zastosowania.

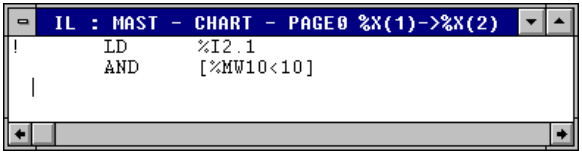


Reguły programowania w języku LIST

Warunek przejścia bramki programuje się w formie listy instrukcji zawierającej tylko instrukcje definiowania warunków (testowania).

Lista instrukcji stosowana do zapisywania warunków przejścia różni się od listy standardowej w następujący sposób:

- struktura ogólna:
  - brak etykiety (%L),
- lista instrukcji:
  - brak instrukcji akcji (bitów obiektowych, słów lub bloków funkcyjnych),
  - brak skoków, wywołań procedur.

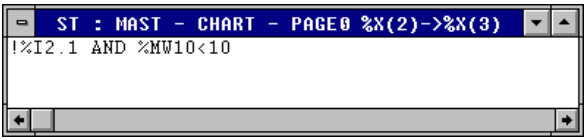


Reguły programowania w języku strukturalnym ST

Warunek przejścia bramki programuje się w formie wyrażeń logicznych, arytmetycznych lub ich kombinacji.

Wyrażenia służące do zapisywania warunków przejścia różnią się od zwykłej linii języka *ST* w następujący sposób:

- struktura ogólna:
  - brak etykiety (%L),
  - brak wyrażeń akcji, wyrażeń warunkowych oraz iteracyjnych,
- lista instrukcji:
  - brak akcji w odniesieniu do bitów obiektowych,
  - brak skoków, wywołań procedur,
  - brak transferu oraz instrukcji akcji w odniesieniu do bloków.

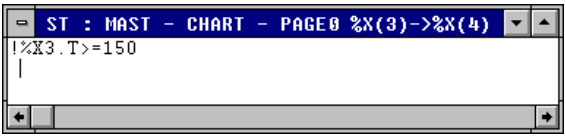


Warunek przejścia wykorzystujący czas aktywności kroku

W pewnych aplikacjach akcje są sterowane bez sprzężenia zwrotnego (koniec trasy, czujnik, itp.). Czas trwania aktywności kroku można uwarunkować: język PL7 umożliwia definiowanie czasu aktywności każdego kroku.

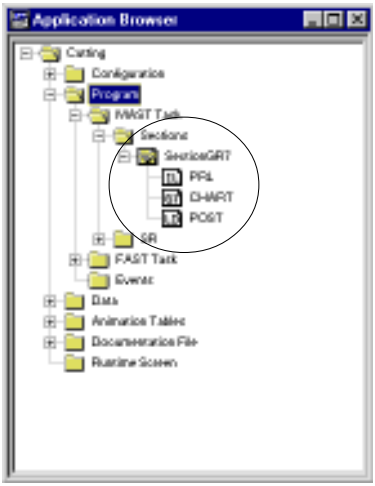
Przykład:

Jeżeli użytkownik pragnie, by krok 3 trwał 15 sekund, to warunek przejścia pomiędzy krokiem 3 i 4 będzie wyglądał następująco (np. w języku strukturalnym ST):



5.9 Organizacja sekcji Grafcet

5.9-1 Wprowadzenie



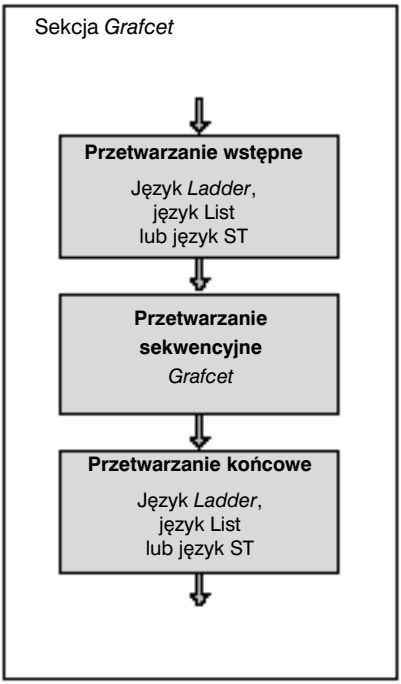
Program zapisany w języku *Grafcet* można podzielić na trzy kolejne sekcje:

- przetwarzanie wstępne *Prl*,
- przetwarzanie sekwencyjne *Chart*,
- przetwarzanie końcowe *Post*.

Makrodefinicje są wykonywane w takim porządku, w jakim są wczytywane w czasie przetwarzania sekwencyjnego.

Sekcja języka *Grafcet* umieszczana jest w zadaniu głównym MAST.

Sekcja jest wczytywana w następującym porządku:



Przetwarzanie wstępne

Umożliwia przetwarzanie:

- inicjacjujące w razie zaniku zasilania,
- przygotowujące diagram *Grafcet*,
- wejść logicznych.

Przetwarzanie sekwencyjne:

Służy do przetwarzania struktury sekwencyjnej aplikacji i umożliwia przetwarzania warunków przejścia oraz akcji powiązanych bezpośrednio z krokami.

Przetwarzanie końcowe:

Umożliwia przetwarzanie:

- logicznych wyjść,
- monitorowanie i zabezpieczanie wyjść.

5.9-2 Przetwarzanie wstępne

Tę część programuje się w języku *Ladder*, języku *LIST* lub *ST* i jest ona wykonywana w całości od góry do dołu.

Przetwarzanie wstępne wykonywane jest przed przetwarzaniem sekwencyjnym i przetwarzaniem końcowym i służy ono do przetwarzania wszystkich zdarzeń mogących mieć wpływ na:

- zarządzaniem systemem w razie zaniku zasilania oraz podczas reinicjacji,
- kasowaniu lub wstępnym zadawaniu parametrów diagramów *Grafcet*.

Stąd też tylko w przypadku przetwarzania wstępnego powyższych zdarzeń wykorzystane zostaną bity sprzężone z krokami (nadanie wartości 0 lub 1 bitom kroków %Xi lub %Xi.j za pośrednictwem instrukcji *Set* i *Reset*).

Przetwarzanie przygotowujące diagram Grafcet

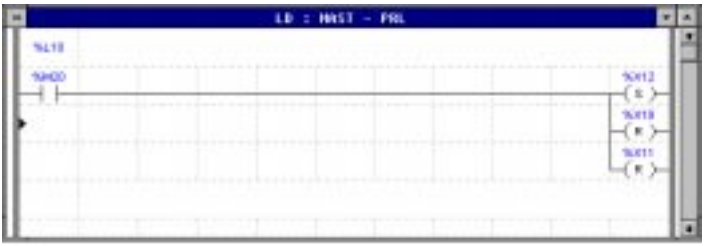
Czasami konieczne jest przygotowanie diagramu *Grafcet*, gdy ma nastąpić zmiana trybu pracy z pracy normalnej na tryb pracy specjalnej lub w razie wystąpienia jakiegoś zdarzenia (przykład: błąd powodujący zakłócenie pracy).

Operacje te powodują zakłócenie normalnej pracy programu, dlatego też powinny być stosowane z dużą ostrożnością. Przetwarzanie przygotowujące może odnosić się do całości lub części przetwarzania sekwencyjnego:

- za pośrednictwem instrukcji SET i RESET,
- poprzez zresetowanie całego systemu (%S22) i nadanie krokom, przy następnym przejściu programu, wartości 1.

Uwaga:

Po skasowaniu kroku do zera, akcje związane z dezaktywacją kroku nie są wykonywane.



5.9-3 Wykorzystanie bitów systemowych w przetwarzaniu wstępnym

Ponieważ bity systemowe związane z diagramem *Grafcet* są ponumerowane w porządku ważności (%S21 do %S24), jednoczesne nadanie im wartości 1 podczas przetwarzania wstępnego powoduje, że są one przetwarzane jeden po drugim w kolejności rosnących numerów (w jednym "przejściu" programu przetwarzany jest tylko jeden bit). Bity te są uwzględniane na początku przetwarzania sekwencyjnego.

Inicjacja diagramu Grafcet: %S21

Bit %S21 ma normalnie wartość 0, nadanie mu wartości 1 powoduje:

- dezaktywację aktywnych kroków,
- uaktywnienie kroków inicjujących.

Nadanie wartości 1	Skasowanie do 0
• Przez nadanie %S0 wartości 1 • Za pomocą programu • Za pomocą terminala (1)	• Przez system, na początku przetwarzania sekwencyjnego • Za pomocą programu • Za pomocą terminala

- **Stosowanie**
  W przypadku nadawania wartości za pomocą programu, %S21 musi mieć nadaną w czasie **przetwarzania wstępnego** wartość 0 lub 1.

Skasowanie diagramu Grafcet do zera: %S22

Bit %S22 ma normalnie wartość 0, nadanie mu wartości 1 powoduje dezaktywację wszystkich aktywnych kroków przetwarzania sekwencyjnego.

Nadanie wartości 1	Skasowanie do 0
• Za pomocą programu • Za pomocą terminala (1)	• Przez system, na końcu przetwarzania końcowego.

(1) Z poziomu ekranu poprawiania *Grafcet-u* (*debug screen*) lub w tablicy animacji.

- **Reguły**
  - temu bitowi należy nadać **w przetwarzaniu wstępnym** wartość 1,
  - skasowanie %S22 do 0 przez system; stąd **nie trzeba go kasować do 0** za pomocą programu, czy terminala.

Aby, w danej sytuacji, możliwe było ponowne wystartowanie przetwarzania sekwencyjnego, aplikacja musi zawierać procedurę inicjacji lub procedurę przygotowawczą diagramu *Grafcet*.

Zamrażanie diagramu Grafcet: %S23

Bit %S23 ma normalnie wartość 0, a zmiana wartości na 1 powoduje zatrzymanie wykonywania diagramu. Niezależnie od wartości warunków przejścia następujących po aktywnych krokach stan diagramu się nie zmienia. Taki stan zamrożenia trwa dopóty, dopóki bit %S23 ma wartość 1.

Nadanie wartości 1	Nadanie wartości 0
• Za pomocą programu • Za pomocą terminala (1)	• Za pomocą programu • Za pomocą terminala (1)

(1) Z poziomu ekranu poprawiania *Grafcet-u* (*debug screen*) lub w tablicy animacji.

- **Reguły**
  - w przypadku wykorzystania programu, bitowi nadaje się wartość 1 lub 0 **przy przetwarzaniu wstępym**,
  - bit %S23 sprzężony z bitami %S21 i %S22 umożliwia zamrożenie przetwarzania początkowego z wartościami inicjującymi lub ze stanem 0. Analogicznie, diagram *Grafcet* może być przy użyciu bitu %S23 wstępnie przygotowany, po czym zamrożony.

Przy uruchamianiu nowej aplikacji lub w przypadku utraty kontekstu, system inicjuje zimny start. System nadaje bitowi %S21 wartość 1 jeszcze przed przed uruchomieniem przetwarzania wstępnego tak, że diagram jest przygotowany do wykonywania z uwzględnieniem kroków inicjujących. Jeżeli użytkownik chce aby aplikacja była po zimnym starcie wykonywana w jakiś szczególny sposób, to może on tego dokonać wykorzystując wartość bitu %S20, który ma stan 1 przez cały czas trwania pierwszego "przejścia" zadania głównego MAST.

W przypadku zaniku zasilania bez utraty kontekstu aplikacji, system inicjuje gorący start rozpoczynając wykonywanie aplikacji od miejsca, w którym wystąpił zanik. Jeżeli użytkownik chce aby aplikacja była po gorącym starcie wykonywana w jakiś szczególny sposób, to może on tego dokonać wykorzystując, przy przetwarzaniu przygotowującym, wartość bitu %S1 i wywołując odpowiedni program.

Kasowanie makrodefinicji do 0: %S24

Bit %S24 ma normalnie stan 0, przestawienie go na 1 powoduje skasowanie do zera makr wskazanych w tablicy mieszczącej 4 słowa systemowe (%SW22 do %SW25).

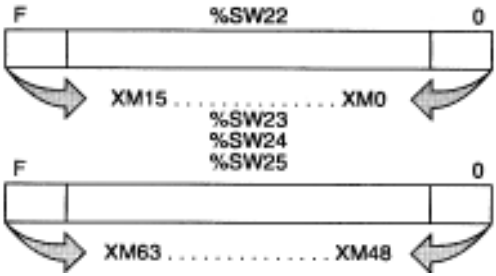
Nadanie wartości 1	Skasowanie do 0
• Za pomocą programu	• Przez system, na początku przetwarzania sekwencyjnego

- **Reguły**
  - wartość tego bitu winna być zmieniana na 1 **tylko podczas przetwarzania wstępnego**,
  - kasowanie %S24 do 0 realizuje system; **stąd nie ma potrzeby kasowania** za pomocą programu lub terminala.

Tablica słów %SW22 do %SW25

Każdemu bitowi z tej tablicy przyporządkowana jest makrodefinicja.
Tablica jest wykorzystywana w następujący sposób:

- wczytywanie tablicy słów %SW22 do %SW25 (wartość 1 nadaje się bitom odpowiadającym makrodefinicjom, które nie mają być kasowane do 0),
- odblokowanie, za pomocą bitu %S24.



Przykład: ! IF %I4.2 AND %T3.D THEN
%SW22:=16#AF8F;
%SW23:=16#F3FF;
%SW24:=16#FFEF;
%SW25:=16#FFFF;
SET %S24

Jeśli %S21 = 1, to te cztery słowa są inicjowane z wartością 16#FFFF.

5.9-4 Przetwarzanie sekwencyjne

Ta część programu stosowana jest do programowania sekwencyjnej struktury programu. Przetwarzanie takie składa się z diagramu głównego zapisanego na 8 stronach.

W diagramie głównym można zamieścić kilka autonomicznych diagramów *Grafcet*, które mogą być programowane i wykonywane jednocześnie.

Przebieg procesu

Diagram *Grafcet* powinien być programowany w następujący sposób:

Faza 1:

1. Oszacowanie warunków aktywnych bramek.
2. Żądanie dezaktywacji, połączonych z bramką, kroków poprzedzających.
3. Żądanie uaktywnienia, połączonych z bramką, kroków następnych.

Faza 2:

Zaktualizowanie stanu diagramu z uwzględnieniem otwartych bramek:

1. Dezaktywacja kroków poprzedzających otwarte bramki.
2. Uaktywnienie kroków następujących po otwartych bramkach.
3. Zablokowanie otwartych bramek.
4. Odblokowanie bramek następujących po nowo uaktywnionych krokach.

System aktualizuje dwie dedykowane tablice zwierające informacje o aktywności kroków i odblokowaniu bramek:

- **tablica aktywności kroków** przechowuje aktualne (dla bieżącego "przejścia" programu) informacje o krokach aktywnych, krokach, które mają być uaktywnione oraz o krokach, które mają być zdezaktywowane,
- **tablica odblokowania bramek** przechowuje aktualne (dla bieżącego "przejścia" programu) informacje o bramkach następujących po krokach uwzględnionych w poprzedniej tablicy,

Faza 3:

Akcje przyporządkowane aktywnym krokom są wykonywane w następujący sposób:

1. Akcje związane z dezaktywacją kroków, które są dezaktywowane.
2. Akcje związane z uaktywnianiem kroków, które są uaktywaniane.
3. Akcje związane z aktywnymi krokami, wykonywane w sposób ciągły.

Przekroczenie wartości dopuszczalnych

Liczba elementów w tablicy aktywności kroków i tablicy odblokowania bramek jest konfigurowalna. Przekroczenie pojemności tych tablic powoduje:

- zatrzymanie pracy sterownika (przerwanie wykonywania aplikacji),
- zmianę stanu bitu %S26 na 1 (przekroczenie pojemności jednej z tablic),
- zapalenie się kontrolki ERR na sterowniku.

System daje do dyspozycji użytkownikowi dwa słowa systemowe:

- **%SW20** : słowo zawierające (dla bieżącego "przejścia") liczbę kroków aktywnych, do uaktywnienia i dezaktywacji,
- **%SW21** : słowo zawierające (dla bieżącego "przejścia") liczbę bramek odblokowanych, do odblokowania i do zablokowania,

W przypadku wystąpienia błędu blokującego pracę sterownika słowa systemowe **%SW125 do %SW127** pozwalają na określenie natury tego błędu.

- %SW125 = DEF7 (hex) Przepełnienie tablicy (kroków/bramek).
- %SW125 = DEFE (hex) Wykonywanie niewłaściwego diagramu. (problem z odwołaniem do nieistniejącego obiektu docelowego).

%SW125	%SW126	%SW127	
DEF7	0	= 0	Przepełnienie tablicy kroków
DEF7	= 0	0	Przepełnienie tablicy bramek
DEFE	Nr kroku	Nr makra (1)	Niewłaściwe wyk. diagramu

(1) lub 64 dla diagramu głównego.

5.9-5 Przetwarzanie końcowe

Przetwarzanie końcowe programuje się w języku *Ladder*, języku instrukcji *LIST* lub języku strukturalnym *ST*. Jest to przetwarzanie wykonywane jako ostatnie, przed zaktualizowaniem stanów wyjść, może być wykorzystane do programowania logiki wyjść.

Akcje związane z diagramem Grafcet

Przetwarzanie końcowe umożliwia wykonanie akcji wygenerowanych w czasie przetwarzania sekwencyjnego poprzez zintegrowanie trybu pracy i zatrzymania oraz uwzględnienie zabezpieczeń (*safety interlocks*) przypisanych do akcji podczas obliczania stanów wyjść. To przetwarzanie może być również wykorzystane do przetworzenia wyjść kilkukrotnie uaktywnianych podczas przetwarzania sekwencyjnego.

Jako regułę należy traktować programowanie, w przetwarzaniu końcowym, akcji mających bezpośredni wpływ na sterowany proces.

Przykład:

- **%I2.4** : zabezpieczenie wyjścia %Q4.1.
- **%M26** : bit wewnętrzny, którego stan jest odbiciem logiki wejść kontrolującej tryby pracy i zatrzymania sterownika.
- **%I1.0** : przycisk.

Wyjście %Q4.1 jest uaktywniane w przetwarzaniu sekwencyjnym przez kroki 5, 8 i 59.

Akcje niezależne w stosunku do diagramu Grafcet

Przetwarzanie końcowe jest również wykorzystywane do programowania wyjść, których stany mają być niezależne od przetwarzania sekwencyjnego.

Monitorowanie wykonywania diagramu Grafcet

W niektórych okolicznościach może zaistnieć konieczność monitorowania działania diagramu. Dokonuje się tego poprzez odczytywanie czasu aktywności pewnych kroków.

Odczytany czas jest porównywany z wartością zdefiniowaną przez użytkownika (z wartością maksymalną lub minimalną). Sposób reakcji na przekroczenie zdanej wartości zależy od użytkownika (sygnalizacja błędu, uruchomienie specjalnej procedury, wygenerowanie komunikatu).

Przykład :
```
! IF (%X2.T > 100 AND %X2) THEN
        SET %Q4.0 ;
END_IF ;
```

6.1 Prezentacja bloków typu DFB

6.1-1 Wiadomości ogólne

Dzięki możliwościom języka PL7 użytkownik może tworzyć własne bloki funkcyjne, dostosowane do wymagań jakie stawia przed nim konkretna aplikacja.

Utworzone przez użytkownika bloki mogą być umieszczane w strukturze aplikacji. Mogą być stosowane wszędzie, gdzie sekwencja programu jest kilkukrotnie powtarzana lub w razie konieczności zamrożenia standardowego programu (np. algorytm sterujący silnikiem uwzględniający działanie zabezpieczeń miejscowych).

Utworzone przez jednego użytkownika bloki mogą być wykorzystywane przez innych użytkowników, mogą być stosowane wielokrotnie w tej samej aplikacji albo w innych aplikacjach (funkcja export/import).

Zastosowanie bloków własnych użytkownika w aplikacji powoduje:
- uproszczenie tworzenia i zapisu programu,
- zwiększenie przejrzystości programu,
- uproszczenie poprawiania programu (wszystkie zmienne "obrabiane" przez blok DFB mogą być łatwo zidentyfikowane),
- redukcję rozmiarów generowanych kodów (kod odpowiadający blokowi DFB jest zapisywany do pamięci tylko jeden raz, bez względu na to ile razy blok DFB jest wywoływany w programie).

W porównaniu z procedurami, bloki własne użytkownika pozwalają na:
- ułatwienie wprowadzania przetwarzanych parametrów,
- wykorzystanie, niezależnych od aplikacji, wewnętrznych zmiennych bloków DFB,
- niezależne testowanie aplikacji.

W języku *Ladder* dla uproszczenia programowania i poprawiania (*debugging*) można wyświetlać bloki typu DFB w formie graficznej.

W dodatku, bloki własne użytkownika mogą wykorzystywać pozostałe dane.

Bloki własne użytkownika (DFB) tworzy się przy użyciu oprogramowania PL7 Pro. Mogą one być stosowane w sterownikach TSX/PCX/PMX 57 pracujących z oprogramowaniem PL7 Pro lub PL7 Junior.

Bloki typu DFB można programować przy pomocy języka ST, a stosować w językach *Ladder* (LD), ST oraz LIST.

6.1-2 Konfigurowanie bloku typu DFB

Bloki DFB konfiguruje się w trzech etapach:

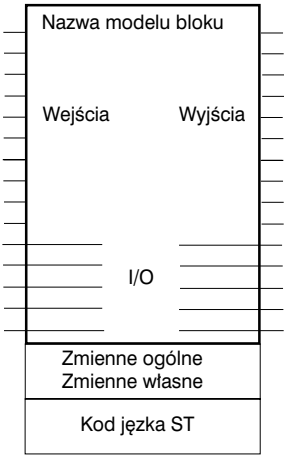
1 tworzenie modelu DFB (typ bloku DFB),

2 tworzenie obrazu tego bloku, zwanego blokiem zastępczym (*instance*), za każdym razem, gdy jest on użyty w aplikacji

3 zastosowanie bloku zastępczego w programie PL7.

Tworzenie modelu bloku DFB
Model bloku składa się z:

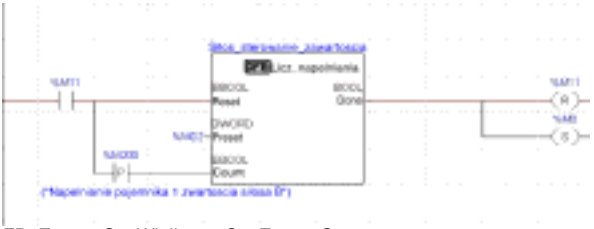
- nazwy
- parametrów:
  - wejść
  - wyjść
  - I/O
- zmiennych:
  - ogólnych (*public variables*)
  - własnych (*private variables*)
- kodu języka ST (*Structured Text*)
- komentarza
- pliku opisowego.



Te informacje muszą być zdefiniowane w fazie tworzenia modelu bloku. Model bloku tworzy się przy pomocy edytora bloków DFB.

Tworzenie bloku zastępczego
Na bazie raz utworzonego modelu bloku użytkownik definiuje blok zastępczy, czy to korzystając z edytora zmiennych, czy też edytora programu (przez wywołanie funkcji).

Zastosowanie bloków w programie
Zdefiniowany blok zastępczy może być potem wykorzystywany tak samo jak standardowe bloki funkcyjne języka *Ladder* lub jako funkcja podstawowa języka *ST* lub języka *LIST*. Bloki mogą być wykorzystywane w różnych zadaniach (z wyjątkiem zadań wyzwalanych zdarzeniami) i sekcjach aplikacji.



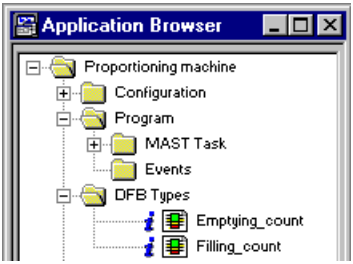
Przykład bloku DFB: Zawor_C1, Wielkosc_C1, Zegar_C1

6.2 Projektowanie bloku DFB

6.2-1 Tworzenie modelu bloku

Modele bloków DFB tworzy się w katalogu *DFB Types* programu *Application Browser* (Przeglądanie aplikacji).

Każdemu blokowi DFB nadaje się nazwę, która może składać się maksymalnie z 16 znaków (1).



6.2-2 Opis parametrów i zmiennych

Charakterystyka ogólna

Parametry i zmienne stanowią wewnętrzne dane bloków DFB i mają charakter czysto symboliczny (nie mają żadnego adresu). Podczas tworzenia modelu własnego bloku użytkownika, dla każdego zastosowanego parametru, czy zmiennej definiuje się:

- nazwę składającą się maksymalnie z 8 znaków (1),
- typ obiektu (patrz tabela poniżej),
- komentarz (opcjonalnie) o maksymalnie 80 znakach,
- wartość początkową (z wyjątkiem parametrów I/O).

Dozwolone typy obiektów

BOOL	Logiczny	AR_X	Tablica bitów
EBOOL	Logiczny rozszerzony (zbocza)	AR_R	Tablica rzeczywista
REAL	Rzeczywisty	AR_W	Tablica 16-bit.
WORD	Liczba całkowita 16-bit.	AR_D	Tablica 32-bit.
DWORD	Liczba calkowita 32-bit.	STRING	Łańcuch znaków

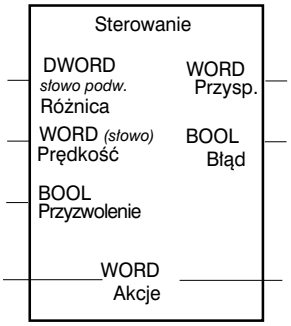
Uwaga:

- W przypadku parametrów i zmiennych typu EBOOL możliwe jest wykonywanie instrukcji w reakcji na wystąpnienie zbocza opadającego, czy narastającego tych parametrów i zmiennych (przykład obiektów typu PL7 EBOOL: %Mi,%Ixy.i,%Qxy.i). Jeżeli zbocza sygnałów nie mają być wykorzystywane, to należy użyć obiektów typu BOOL (przykład obiektu PL7 BOOL: %MWi:Xj), które pozbawione są tej cechy, a zajmują mniej pamięci.
- Jeżeli obiekt typu EBOOL jest powiązany z parametrem I/O, to w bloku DFB musi być zadeklarowane użycie obiektów typu EBOOL.
- Tablice: musi być zdefiniowana długość tablicy dla parametrów wyjść oraz zmiennych ogólnych i własnych ale nie ma potrzeby definiowania długości dla wejść i parametrów I/O.
- Wartości początkowe (inicjacyjne) definiuje się dla wejść (o ile nie są to tablice), wyjść i zmiennych ogólnych oraz własnych.

(1) Dopuszcza się stosowanie liter bez akcentów, cyfr oraz znaku "_". Pierwszy znak musi być literą. Nie wolno stosować słów kluczowych (kodów) oraz symboli.

Parametry:

- **Wejść**: (max 15 (1)) dane przesyłane przez program aplikacji do bloku DFB. Te parametry dostępne są tylko do odczytu i nie mogą być modyfikowane w bloku DFB.
- **Wyjść**: (max 15 (2)) dane generowane przez blok DFB i zwracane do programu aplikacji.
- **I/O**: (max 15 (1) i (2)) parametry wejściowe, które mogą być modyfikowane w bloku DFB.



Przykład bloku z:
3 wejściami, 2 wyjściami, 1 I/O

(1) gdzie Liczba Wejść + Liczba I/O ≤ 15
(2) gdzie Liczba Wyjść + Liczba I/O ≤ 15

Uwaga: Każdy blok typu DFB musi mieć co najmniej jedno wejście logiczne.

Zmienne

- **Zmiene ogólne**: (max 100) są to zmienne wewnętrzne wykorzystywane podczas przetwarzania. Są one dostępne dla użytkownika z poziomu trybu regulacji (*adjust mode*) lub za pośrednictwem programu, poza blokiem DFB (jako zmienne ogólne bloku zastępczego DFB, patrz rozdział 6.4-4).
  Dodatkowo, jest możliwość określania atrybutów (tylko do odczytu *read-only* lub praw do odczytu i zapisu *read*/*write*) oraz autoryzacji zapisywania zmiennych bieżących jako wartości początkowych (patrz rozdział 6.4-5).

- **Zmienne własne**: (max 100) zmienne wewnętrzne bloku funkcyjnego. Zmienne te są obliczane i wykorzystywane wewnętrz bloku DFB, a nie są dostępne poza nim. Służą one do programowania bloku ale leżą poza sferą zainteresowania użytkownika takiego bloku (np. wartość pośrednia wymieniana pomiędzy jednym wyrażeniem a drugim, wynik obliczeń pośrednich).

Uwaga: Zmienne ogólne lub parametry bloku własnego użytkownika DFB mogą być modyfikowane tylko wtedy, kiedy nie jest on używany przez aplikację i nie jest tworzony blok zastępczy.

6.2-3 Kod modelu bloku

Kod definiuje jakie operacje mają być wykonane w bloku z uwzględnieniem zadeklarowanych parametrów.

Kod DFB programuje się w języku ST (*Structured Text*).

Kod realizuje się w pojedynczym wyrażeniu języka ST o dowolnej długości.

Dopuszczalne jest stosowanie niemal wszystkich instrukcji językowych i funkcji złożonych, z wyjątkiem:

- wywoływania standardowych bloków funkcyjnych
- wywoływania innych bloków DFB
- skoku (JUMP)
- wywoływania procedury
- instrukcji HALT (zatrzymanie)
- instrukcji wykorzystujących zmienne modułu I/O (np. READ_STS, SMOVE,...).

W kodzie DFB wykorzysuje się parametry bloku zdefiniowane przez użytkownika.

```
CHR_200:=CHR_100;
CHR_114:=CHR_104;
CHR_116:=CHR_106;
RESET RESTART;
(* Zwiekszenie CHR_100 80 razy*)
FOR CHR_102:=1 TO 80 DO
        INC CHR_100;
        WHILE((CHR_104-
CHR_114)<100)DO
        IF(CHR_104>400) THEN
EXIT;
        END_IF;
        INC CHR_104;
        REPEAT
        IF(CHR_106>300) THEN
EXIT;
        END_IF;
        INC CHR_106;
        UNTIL ((CHR_100-
CHR_116)>100)
        END_REPEAT;
        END_WHILE;
        (* Loop CHR_106)
        IF (CHR_106=CHR_116)
        THEN EXIT;
        ELSE
          CHR_114:=CHR_104;
          CHR_116:=CHR_106;
        END_IF;
        INC CHR_200;
END_FOR;
```

Kod DFB nie może wykorzystywać obiektów I/O (%I,%Q...) oraz obiektów ogólnych (globalnych) aplikacji (%MW,%KW...) z wyjątkiem bitów oraz słów systemowych %S i %SW.

Niektóre z funkcji zaprojektowano z myślą o własnych blokach funkcyjnych użytkownika:

- FTON, FTOF, FTP i FPULSOR - funkcje opóźniające, które mogą być użyte zamiast bloków funkcyjnych typu zegar (*timer*),
- LW, HW i COCATW - instrukcje, które mogą być wykorzystane do obsługi słów oraz słów podwójnej precyzji,
- LENGTH_ARW, LENGTH_ARD i LENGTH_ARR - instrukcje, które mogą być wykorzystane do obliczania długości tablic.

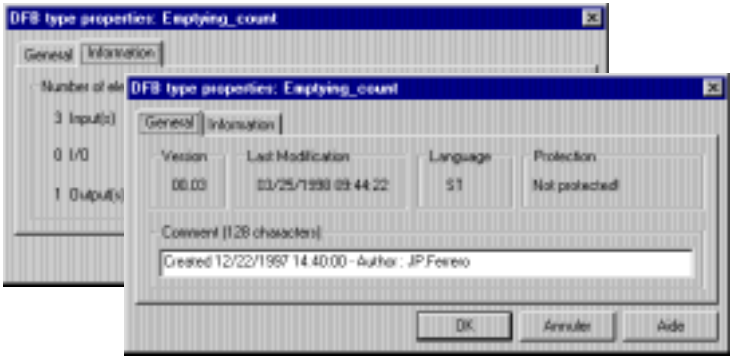
Uwaga: Nie można stosować etykiet.

6.2-4 Zatwierdzanie modelu bloku DFB

Podczas zatwierdzania edytor bloków DFB sprawdza poprawność składni kodu i dopasowanie do zmiennych wejściowych. W przypadku wychwycenia błędów, pierwszy z nich wyświetlany jest w inwersji.
Uwaga: jeżeli użytkownik chce wyjść z projektowania bloku bez jego zatwierdzenia, to należy wpierw wyeksportować blok (nastąpi jego zapisanie) i skasować kod przed porzuceniem projektowania.

6.2-5 Właściwości modelu bloku DFB

Za pośrednictwem okna dialogowego edytora bloków DFB użytkownik ma dostęp do następujących właściwości bloku:



- **karta "informacyjna" (Information)**, w której zarządza się:
  - ilością (rozmiarem) danych (*Data size*),
  - liczbą utworzonych bloków zastępczych (*Number of instances*),
  - identyfikacją ID aplikacji (zapewnienie, że model bloku DFB jest unikatowy),
  - liczbą elementów: parametrów i zmiennych (*Number of elements*),

- **karta "ogólna" (General)**, która zawiera :
  - wersję: wartość jest automatycznie zwiększana, gdy modyfikowany jest kod, parametry lub zmienne bloku DFB (*Version*),
  - datę ostaniej modyfikacji (*Last Modification*),
  - 2 rodzaje zabezpieczeń: zabezpieczenie przed możliwością modyfikacji bloku (udostępnienie bloku tylko do odczytu) lub ochrona hasłem przed dostępem do kodu modelu bloku i jego zmiennych własnych (*Protection*).
  - komentarz, który może składać się maksymalnie ze 128 znaków; domyślnie, komentarz zawiera datę utworzenia oraz twórcę modelu bloku DFB (*Comment*).

6.2-6 Plik opisowy

Plik opisowy będący polem ogólnie dostępnym, służącym do opisywania modelu bloku DFB, przy czym może on zawierać maksymalnie 32 000 znaków.

6.2-7 Import/export modelu bloku

Modele bloków utworzone w danej aplikacji mogą być wykorzystywane w innej aplikacji. Jest to możliwe dzięki funkcjom Export oraz Import.

Exportu można dokonywać w dwóch formatach:
- źródłowym: ten format może być poddawany edycji,
- binarnym: ten format nie może być edytowany. Zaimportowany w tym formacie blok DFB może być odczytywany lub nawet modyfikowany, w zależności od poziomu zastosowanego zabezpieczenia.

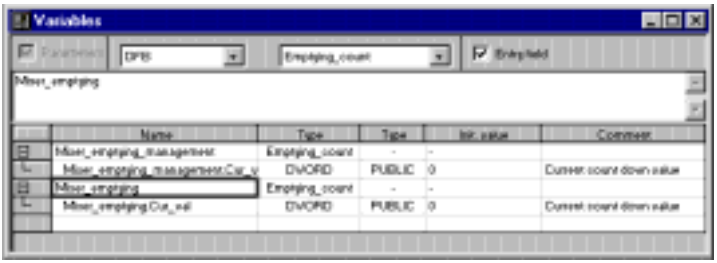
Wykorzystując te pliki binarne i źródłowe użytkownik może tworzyć własne biblioteki.

Uwaga:
Bez względu na to, że sekcja jest przeniesiona (import) do aplikacji, modele bloków własnych użytkownika DFB wykorzystanych w tej sekcji też muszą zostać przeniesione (import).

6.3 Tworzenie bloków zastępczych

6.3-1 Zasady ogólne

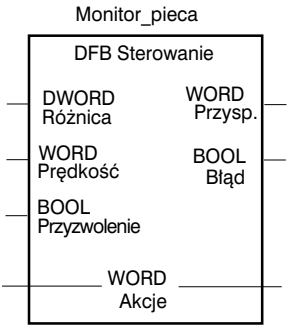
Po utworzeniu modelu bloku (lub zaimportowaniu), przed jego wykorzystaniem, należy wygenerować blok zastępczy dla danego modelu.
Każdemu blokowi zastępczemu użytkownik przypisuje nazwę, o maksymalnie 32 znakach, która umożliwia jego adresowanie.



Dla danego modelu bloku DFB można utworzyć dowolną liczbę bloków zastępczych (liczba ta jest ograniczona tylko pojemnością pamięci sterownika).
Dla każdego bloku zastępczego można dokonywać modyfikacji zarówno wartości początkowych (inicjujących) jak i zmiennych ogólnych.

Operacji tych dokonuje się w edytorze zmiennych (dla bloku zastępczego) lub w edytorze programu, po wywołaniu funkcji. Wszystkie bloki zastępcze są dostępne za pośrednictwem biblioteki funkcji (*function library*).

Blok zastępczy stanowi kopię modelu DFB:
- wykorzystuje kod modelu DFB (kod nie jest kopiowany),
- tworzy własny obszar danych dla tego bloku zastępczego, który jest kopią parametrów i zmiennych modelu DFB. Obszar ten jest usytuowany w strefie danych aplikacji.



Przykład bloku zastępczego "Monitor_pieca" na bazie modelu bloku DFB o nazwie *Sterowanie*

(1) Znaki jakich można używać są takie same jak w przypadku symboli (patrz rozdział 1.2-9)

6.4 Stosowanie bloków DFB

6.4-1 Ogólne zasady programowania

Bloki zastępcze DFB mogą być programowane we wszystkich językach (*Ladder*, *Structured Text* i *Instruction List*), we wszystkich częściach aplikacji: sekcjach, procedurach, module *Grafcet* (z wyjątkiem zadań wyzwalanych zdarzeniami).

Bez względu na język programowania obowiązują następujące reguły:

- Muszą być zdefiniowane wszystkie, będące tablicami, parametry wejść oraz I/O.
- Parametry nie podłączonych wejść zachowują wartości z poprzedniego wywołania lub wartości inicjujące (początkowe) jeżeli wejście tego bloku podczas poprzednich wywołań nie było nigdy podłączone lub zdefiniowane.
- Wszystkie obiekty przypisane do parametrów wejść, wyjść oraz I/O muszą być tego samego typu jak zostały zdefiniowane podczas tworzenia modelu bloku (np. jeśli parametr wejściowy nazwany Predkosc jest typu słowo, czyli WORD, to nie można mu przypisać słów podwójnej precyzji %MDi, czy %KDi).
  Jedynym wyjątkiem są obiekty typu BOOL i EBOOL dla parametrów wejść i wyjść (nie dotyczy parametrów I/O), które można "mieszać": np. parametr wejściowy "Przyzwolenie"może być zdefiniowany jako typu BOOL i zostać przypisany do bitu %Mi typu EBOOL. Wewnętrzny kod modelu bloku będzie miał właściwości obiektu typu BOOL i nie będzie mógł reagować na zbocza sygnałów.

W tabeli poniżej zestawiono wszystkie, dopuszczalne kombinacje:

Parametr	Typ	Przypisanie parametru	Przypisanie
Wejścia	Logiczny	Podłączenie (1)	opcjonalnie (2)
	Cyfrowy	Obiekt lub wyrażenie	opcjonalnie
	Tablica	Obiekt	koniecznie
I/O	Logiczny	Obiekt	koniecznie
	Cyfrowy	Obiekt	koniecznie
	Tablica	Obiekt	koniecznie
Outputs	Logiczny	Podłączenie (1)	opcjonalnie
	Cyfrowy	Obiekt	opcjonalnie
	Tablica	Obiekt	opcjonalnie

(1) Podłączenie do obiektu języka *Ladder*, ST lub obiektu logicznego.
(2) W języku *Ladder* wszystkie bloki DFB muszą mieć co najmniej jedno podłączone (binarnie) wejście logiczne.

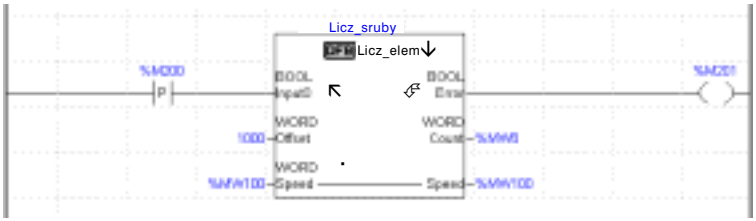
6.4-2 Programowanie w języku Ladder

Istnieją dwa sposoby wywoływania bloku typu DFB:

- wywołanie tekstowe w bloku operacyjnym. Składnia i ograniczenia odnośnie parametrów są takie same jak dla strukturalnego języka tekstowego ST (patrz: kolejny rozdział)
- wywołanie graficzne (patrz przykład poniżej).

  Graficzne bloki DFB mają wejścia/wyjścia (I/O) bezpośrednio przypisywane obiektom lub wyrażeniom. Te obiekty i wyrażenia zajmują jedną komórkę w labelce.

  Dwa bloki DFB połączone ze sobą szeregowo muszą być oddzielone co najmniej dwiema kolumnami.



Nazwa bloku DFB
↓ Nazwa modelu bloku

Parametr wchodzący na pierwsze wejście
↖ Parametry wejściowe (nazwa i typ)
↙ Parametry wyjściowe (nazwa i typ)
· Parametry I/O (nazwa i typ)

Komentarz:

- Blok DFB musi mieć podłączone co najmniej jedno wejście logiczne.
- Cyfrowych wejść, wyjść oraz wejść/wyjść (I/O) nie podłącza się. Obiekt umieszczony w komórce sąsiadującej z pinem (wejście, wyjście, I/O) bloku zostaje do niego przypisany.

6.4-3 Programowanie w języku ST i LIST

Wywołanie bloku DFB jest traktowane jak zwykła akcja, którą można umieszczać w wyrażeniu, czy sekwencji tak jak każdą inną akcję.

Składnia dla języka strukturalnego ST

DFB_Nazwa (I1,...,In,IQ1,...,IQn,Q1,...,Qn)

Przykład: Licz_nakretki (%I2.0,%MD10,%I2.1,%Q1.0);

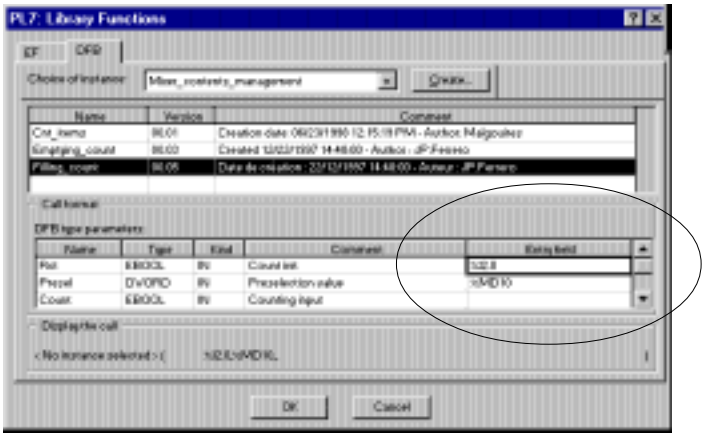
Składnia dla języka instrukcji LIST

[DFB_Nazwa (I1,...,In,IQ1,...,IQ,Q1,...,Qn)]

Przykład : Licz_nakretki (%I2.0,%MD10,%I2.1,%Q1.0);

gdzie:
I1, ..., In : wyrażenia (1), obiekty bezpośrednie lub wartości traktowane jako parametry efektywne dla parametrów wejściowych,
IQ1, ..., IQn : parametry efektywne odpowiadające wejściom/wyjściom I/O; są to zawsze obiekty językowe do zapisu i odczytu,
Q1, ..., Qn : parametry efektywne odpowiadające wyjściom; są to zawsze obiekty językowe do zapisu i odczytu,

Wprowadzanie tych parametrów odbywa się w językach ST oraz LIST przy pomocy okna dialogowego.



(1) z wyjątkiem obiektów typu BOOL/EBOOL.

6.4-4 Dostęp do zmiennych

W programie, poza obszarem bloku funkcyjnego dostępne (jako obiekty) są tylko parametry wyjściowe i zmienne ogólne. Składnia jest następująca:

DFB_nazwa.parametr_nazwa

gdzie: DFB_nazwa nazwą nadaną blokowi zastępczemu DFB (max 32 znaki), a parametr_nazwa jest nazwą nadaną parametrowi wyjściowemu lub zmiennej ogólnej (max 8 znaków).

Przykład:
Sterowanie.Roznica dla wyjścia *Roznica* bloku zastępczego nazwanego *Sterowanie*.

6.4-5 Zapisywanie i odczytywanie z pamięci zmiennych ogólnych

Zmienne ogólne po zmodyfikowaniu w programie lub podczas regulacji (*adjustment*) mogą być zapisywane w miejsce wartości inicjujących (zdefiniowanych w bloku zastępczym) poprzez nadanie bitowi systemowemu %S94 wartości 1. Taka podmiana jest możliwa tylko, gdy użytkownik ma autoryzację na poziomie każdej zmiennej modelu DFB.
Wartości zapisane tym sposobem są ponownie wczytywane, gdy bit systemowy %S95 przyjmuje wartość 1 lub w przypadku reinicjacji sterownika.

6.4-6 Wykonywanie bloków DFB

Blok zastępczy jest wykonywany w następujący sposób:

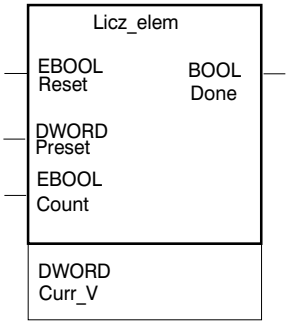
- Parametry wejściowe oraz I/O są ładowane za pomocą parametrów efektywnych. Każde, nie zdefiniowane wejście przyjmuje wartość inicjacyjną zdefiniowaną w modelu bloku podczas jego inicjacji lub podczas zimnego startu, a dopiero potem wartość bieżącą parametru.
  Parametry wejściowe (z wyjątkiem tablic) są traktowane różnie, w zależności od ich wartości. Parametry I/O są traktowane zgodnie z ich adresami.
- Wykonywany jest kod zapisany w języku ST.
- Zapisywane są parametry wyjściowe.

Język PL7 wyposażono w kilka narzędzi umożliwiających testowanie (*debugging*) programu i bloków typu DFB:

- tablica animacji: wszystkie parametry i zmienne ogólne wyświetlane są w tej tablicy w czasie rzeczywistym. Poszczególne obiekty mogą być modyfikowane, a ich stany wymuszane.
- przerwania, diagnostyka programu, wykonywanie krok po kroku,
- monitorowanie przebiegu programu: do testowania przy pomocy zunifikowanego ekranu.

6.5 Przykład

Przykład bloku DFB zaprogramowanego jako licznik.



Charakterystyka modelu bloku DFB

Nazwa : Licz_elem
Wejścia:
- Reset : kasowanie licznika
- Preset : wartość nastawiona licznika
- Count : wejście licznika

Wyjścia:
- Done : wyjście osiągnięcia wartości nastawionej

Zmienne ogólne:
- Curr_V : wartość bieżąca aktualizowana przez wejście *Count*.

Funkcja licznika: ten blok zlicza zbocza narastające sygnału na wejściu *Count*. Wynik zliczania jest rejestrowany w zmiennej *Curr_V*, której wartość może być kasowana zboczem narastającym sygnału na wejściu *Reset*. Zliczanie jest realizowane do momentu osiągnięcia wartości nastawionej. Gdy to następuje, to na wyjściu *Done* wystawiana jest 1. W przypadku pojawienia się zbocza narastającego na wejściu *Reset* wyjście *Done* jest kasowane do 0.

Kod

```
!(*Programowanie bloku DFB o nazwie Licz_elem*)
IF RE Reset THEN
        Curr_V:=0;
END_IF;
IF RE Count THEN
        Curr_V:=Curr_V+1;
END_IF;
IF(Curr_V>=Preset) THEN
        SET Done;
ELSE
        RESET Done;
END_IF;
```

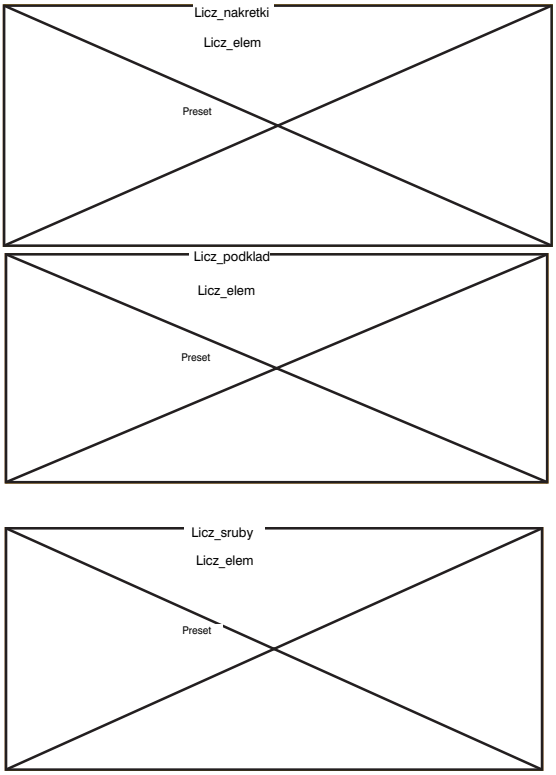
Przykładowe zastosowanie

W tym przykładzie model bloku DFB jest wykorzystany trzykrotnie (3 bloki zastępcze) do liczenia 3 rodzajów elementów. Kiedy następuje odliczenie zadanej ilości elementów (w słowach %MD10, %MD12 i %MD14) wyjście licznika powoduje zatrzymanie systemu odpowiedzialnego za podawanie elementów.

Program

Nazwy bloków użytkowych bazujących na modelu bloku DFB Licz_elem:

- Licz_nakretki
- Cnt_podklad
- Cnt_sruby



Rozdział **Strona**

Rozdział **Strona**

Rozdział **Strona**

Rozdział **Strona**

Rozdział **Strona**

1.1 Wprowadzenie

1.1-1 Wiadomości ogólne

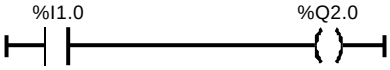
Opisane w niniejszym rozdziale instrukcje są zgodne instrukcjami opisanymi w normie IEC 1131.3.

Te instrukcje wywołują zawsze taki sam efekt, niezależnie od języka w jakim zostaną użyte. Różny jest tylko ich wygląd w programie.

Przykładowe równanie logiczne:

W języku *LIST* : LD %I1.0
ST %Q2.0

W języku *Ladder* :



W języku *ST* : %Q2.0 := %I1.0 ;

Te trzy równania są równoważne. Bit obiektowy %Q2.0 przyjmuje wartość (instrukcja przypisania) bitu obiektowego %1.0 (instrukcja wczytania).

Instrukcjami podstawowymi są:

- instrukcje logiczne (wykonywane na bitach),
- standardowe bloki funkcyjne typu zegar i licznik,
- numeryczne instrukcje dotyczące liczb całkowitych (wykonywane na słowach oraz słowach podwójnej precyzji),
- instrukcje związane z programowaniem.

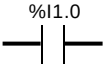
Pozostałe instrukcje zostały opisane w rozdziale 2 zatytułowanym "Instrukcje złożone".

1.2 Instrukcje logiczne

1.2-1 Wprowadzenie

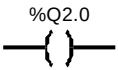
Instrukcje logiczne działają na bitach wszystkich typów (bity I/O, bity wewnętrzne, itd.).

• **Elementy warunków (*test elements*)**, przykład: styk N/O (normalnie otwarty). Styk zamyka się, kiedy związany z nim bit obiektowy, zmienia stan na 1.

LD %I1.0 

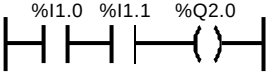
%I1.0

• **Elementy akcji (*action elements*)**, przykład: sprzężenie bezpośrednie (cewka). Sprzężony bit obiektowy przyjmuje logiczną wartość będącą wynikiem logicznego warunku (testowania).

ST %Q2.0 

%Q2.0 :=

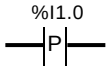
• **Równanie logiczne:**
Logiczny wynik elementów warunków jest przypisany do elementu akcji.

LD %I1.0
AND %I1.1
ST %Q2.0 

%Q2.0 := %I1.0 AND %I1.1 ;

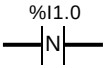
Zbocza sygnałów
Instrukcje testowania warunków mogą być wykorzystywane do wykrywania zmiany stanu (zbocza opadającego lub narastającego) bitów I/O oraz bitów wewnętrznych.

Styk reagujący na zbocze narastające:

LDR %I1.0 

RE %I1.0

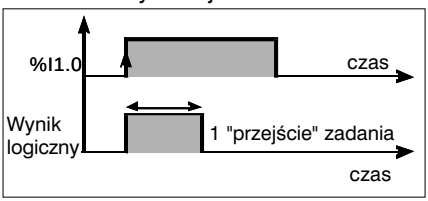
Styk reagujący na zbocze opadające:

LDF %I1.0 

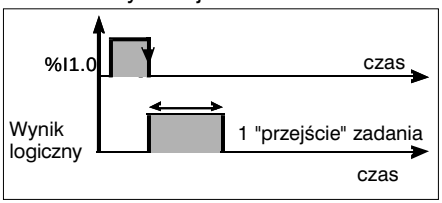
FE %I1.0

• **Wszystkie wejścia (dyskretne, liczniki, itp)**: wykrycie zbocza następuje, gdy bit zmienia stan pomiędzy "przejściem" *n-1,* a bieżącym "przejściem" *n.* Sygnalizacja wykrycia zbocza trwa przez całe "przejście" (patrz A, rozdz. 1.3-2).

Zbocze rosnące: wykrycie zmiany na kontrolowanym wejściu z 0 na 1.



Zbocze opadające: wykrycie zmiany na kontrolowanym wejściu z 1 na 0.



• **Wyjścia i bity wewnętrzne**: wykrywanie zbocza odbywa się niezależnie od "przejścia" zadania. Wykrycie zbocza bitu wewnętrznego %Mi następuje, gdy zmienia on stan pomiędzy dwiema kolejnymi operacjami odczytu. Sygnalizacja wystąpienia zbocza utrzymuje się do momentu uaktualnienia bitu wewnętrznego w strefie akcji.

• Nie można używać instrukcji SET i RESET w stosunku do obiektów, które są testowane pod kątem wykrywania zboczy (zarówno w języku *Ladder* jak i *LIST*).

1.2-2 Format instrukcji

Instrukcje logiczne opisuje się w następujący sposób:

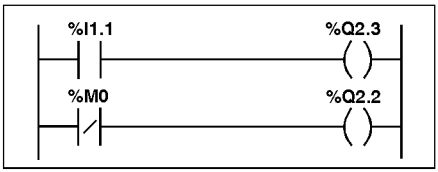
Uwaga:
Opisywana funkcja jest wytłuszczona.
Każde równanie zapisano w kilku językach.

Instrukcje wczytywania *Load*

Instrukcje te odnoszą się do:

- styków N/O: styk jest zamknięty, gdy sterujący nim bit obiektowy ma stan 1.
- ...

Język *Ladder*



Język instrukcji *List*

```
LD    %I1.1
ST    %Q2.3
LDN   %M0
ST    %Q2.2
```

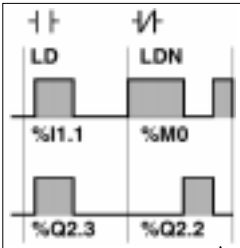
Język strukturalny *Structured Text*

```
%Q2.3 := %I1.1 ;
%Q2.2 := NOT %M0 ;
```

Dozwolone argumenty

Kod		Argument
┤├	**LD**	%I,%Q,%M,%S,%BLK,%•:Xk,%Xi
┤/├	**LDN**	%I,%Q,%M,%S,%BLK,%•:Xk,%Xi

Diagram czasowy

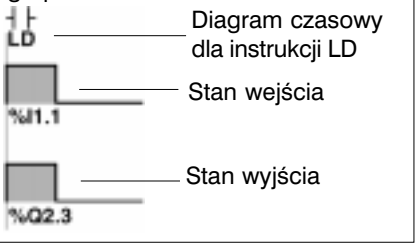


Lista argumentów

0/1	wartość bezp. 0 (fałsz), 1 (prawda)
%I	wejście sterownika %Ix.i
%Q	wyjście sterownika %Qx.i
%M	bit wewnętrzny %Mi
%S	bit systemowy %Si
%BLK	bit standardowego bloku funkcyjnego (np. %TMi.Q) lub bloku DFB
%•:Xk	bit wydz. ze słowa, np. %MWi:Xk
%Xi	bit kroku, bit makra (%XMi) lub bit kroku makra (%Xj.i).

Diagram czasowy

Cztery diagramy czasowe zostały zgrupowane.

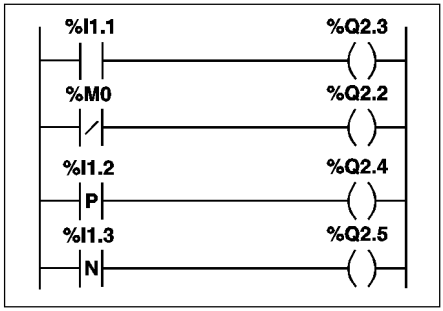


1.2-3 Instrukcje wczytywania *Load*

Instrukcje wczytywania odnoszą się do:

- styków N/O: styk zamknięty, gdy kontrolujący go bit ma stan 1,
- styków N/C: styk zamknięty, gdy kontrolujący go bit ma stan 0,
- styki reagujące na zbocze rosnące: reaguje na zmianę stanu bitu z 0 na 1,
- styki reagujące na zbocze opadające: reaguje na zmianę stanu bitu z 1 na 0.

Język *Ladder*



Język *List*

```
LD    %I1.1
ST    %Q2.3
LDN   %M0
ST    %Q2.2
LDR   %I1.2
ST    %Q2.4
LDF   %I1.3
ST    %Q2.5
```

Język *ST*

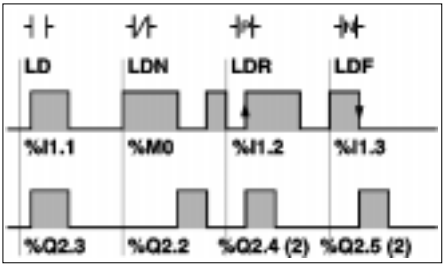
```
%Q2.3 := %I1.1 ;
%Q2.2 := NOT %M0 ;
%Q2.4 := RE %I1.2 ;
%Q2.5 := FE %I1.3 ;
```

Dozwolone argumenty

Kod		Argument
┤ ├	LD	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi (1)
┤/├	LDN	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi (1)
┤P├	LDR	%I,%Q,%M
┤N├	LDF	%I,%Q,%M

(1) W języku *LIST* oraz *ST* - Prawda (1)/Fałsz (0)

Diagram czasowy



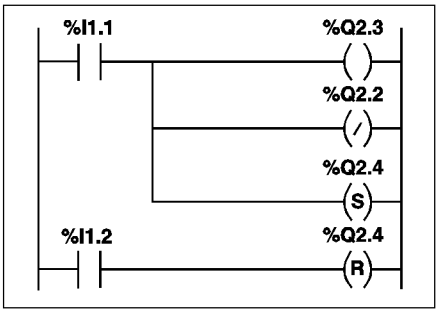
(2) Nadanie wartości w czasie 1 cyklu.

1.2-4 Instrukcje przypisani *Assignment*

Instrukcje przypisania stosuje się w odniesieniu do:

- sprzężeń prostych: sprzężony bit obiektowy przyjmuje wartość wynikającą z równania,
- sprzężeń odwrotnych: sprzężony bit obiektowy przyjmuje wartość odwrotną w stosunku do wynikającej z równania,
- przerzutników *Latch* (*Set*): sprzężony bit ma wartość 1, gdy wynik równania = 1,
- przerzutników (*Reset*): sprzężony bit obiektowy przyjmuje wartość 0, gdy wynikiem równania jest 1.

Język *Ladder*



Język *List*

```
LD      %I1.1
ST      %Q2.3

STN     %Q2.2

S       %Q2.4

LD      %I1.2
R       %Q2.4
```

Operacje równoważne w języku *ST*

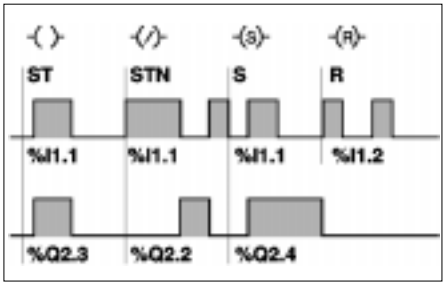
```
%Q2.3 := %I1.1 ;
%Q2.2 := NOT %I1.1 ;
IF %I1.1 THEN
    SET %Q2.4 ;
END_IF ;
IF %I1.2 THEN
    RESET %Q2.4 ;
END_IF ;
```

Dozwolone argumenty

Kod		Argument
-( )-	**ST**	%I,%Q,%M,%S,%•:Xk
-(/)-	**STN**	%I,%Q,%M,%S,%•:Xk
-(S)-	**S**	%I,%Q,%M,%S,%•:Xk, %Xi (1)
-(R)-	**R**	%I,%Q,%M,%S,%•:Xk, %Xi (1)

(1) Tylko przy przetwarzaniu wstępnym (*preprocessing*).

Diagram czasowy

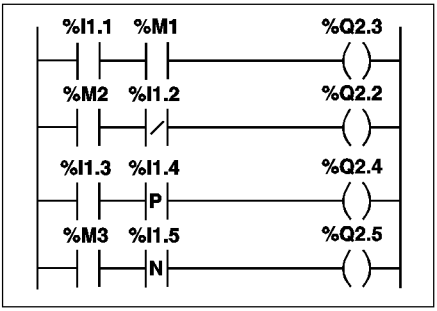


1.2-5 Koniunkcja AND

Funkcja ta obejmuje:

- koniunkcję pomiędzy argumentem a logicznym wynikiem wcześniejszej instrukcji,
- koniunkcję pomiędzy zaprzeczeniem argumentu a logicznym wynikiem wcześniejszej instrukcji,
- koniunkcję pomiędzy zboczem rosnącym argumentu a logicznym wynikiem wcześniejszej instrukcji,
- koniunkcję pomiędzy zboczem opadającym argumentu a logicznym wynikiem wcześniejszej instrukcji,

Język *Ladder*



Język *List*

```
LD      %I1.1
AND     %M1
ST      %Q2.3
LD      %M2
ANDN    %I1.2
ST      %Q2.2
LD      %I1.3
ANDR    %I1.4
ST      %Q2.4
LD      %M3
ANDF    %I1.5
ST      %Q2.5
```

Język *ST*

```
%Q2.3 := %I1.1   AND %M1 ;
%Q2.2 := %M2     AND (NOT %I1.2) ;
%Q2.4 := %I1.3   AND (RE %I1.4) ;
%Q2.5 := %M3     AND (FE %I1.5) ;
```

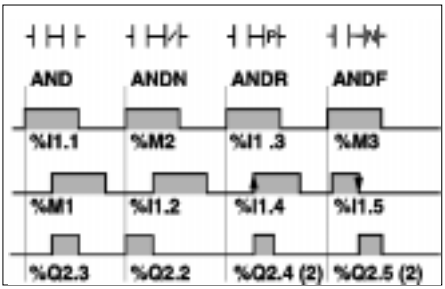
Uwaga: Stosowanie nawiasów nie jest obowiązkowe, ale czyni program bardziej przejrzystym.

Dozwolone argumenty

Kod	Argument
AND	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi (1)
ANDN	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi (1)
ANDR	%I,%Q,%M
ANDF	%I,%Q,%M

(1) W języku *LIST* oraz *ST* - Prawda (1)/Fałsz (0)

Diagram czasowy



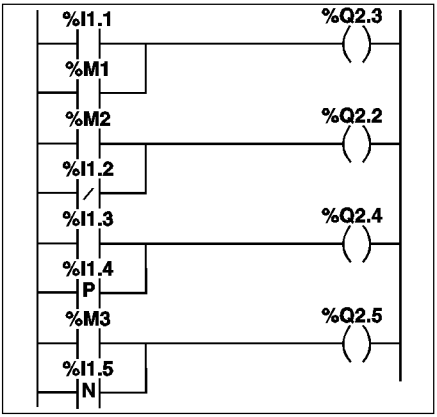
(2) Nadanie wartości 1 w czasie 1 cyklu

1.2-6 Alternatywa OR

Funkcja ta obejmuje:

- alternatywę pomiędzy argumentem a logicznym wynikiem wcześniejszej instrukcji,
- alternatywę pomiędzy zaprzeczeniem argumentu a logicznym wynikiem wcześniejszej instrukcji,
- alternatywę pomiędzy zboczem rosnącym argumentu a logicznym wynikiem wcześniejszej instrukcji,
- alternatywę pomiędzy zboczem opadającym argumentu a logicznym wynikiem wcześniejszej instrukcji,

Język *Ladder*



Język *List*

```
LD    %I1.1
OR    %M1
ST    %Q2.3

LD    %M2
ORN   %I1.2
ST    %Q2.2

LD    %I1.3
ORR   %I1.4
ST    %Q2.4

LD    %M3
ORF   %I1.5
ST    %Q2.5
```

Język *ST*

```
%Q2.3 := %I1.1   OR %M1 ;
%Q2.2 := %M2     OR (NOT %I1.2) ;
%Q2.4 := %I1.3   OR (RE %I1.4) ;
%Q2.5 := %M3     OR (FE %I1.5) ;
```

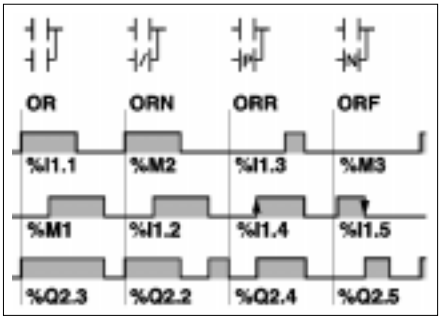
Uwaga: Stosowanie nawiasów nie jest obowiązkowe, ale czyni program bardziej przejrzystym.

Dozwolone argumenty

Kod	Argument
OR	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi (1)
ORN	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi (1)
ORR	%I,%Q,%M
ORF	%I,%Q,%M

(1) W języku *LIST* i *ST* - Prawda (1)/Fałsz (0)

Diagram czasowy



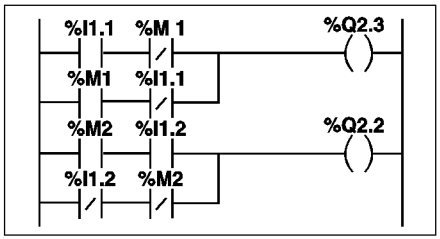
1.2-7 Nierównoważność XOR

Funkcja ta obejmuje:

- nierównoważność pomiędzy argumentem a logicznym wynikiem wcześniejszej instrukcji,
- nierównoważność pomiędzy zaprzeczeniem argumentu a logicznym wynikiem wcześniejszej instrukcji,
- nierównoważność pomiędzy zboczem rosnącym argumentu a logicznym wynikiem wcześniejszej instrukcji,
- nierównoważność pomiędzy zboczem opadającym argumentu a logicznym wynikiem wcześniejszej instrukcji,

Uwaga:
W języku *Ladder* nie ma specjalnych elementów graficznych dla tej funkcji (XOR). Pomimo tego, można zaprogramować tę funkcję przy pomocy kombinacji styków typu N/O (normalnie zamkniętych) i N/C (normalnie zamkniętych) - patrz przykład poniżej.

Zapis równoważny w języku *Ladder*



Język *LIST*

```
LD      %I1.1
XOR     %M1
ST      %Q2.3

LD      %M2
XORN    %I1.2
ST      %Q2.2

LD      %I1.3
XORR    %I1.4
ST      %Q2.4

LD      %M3
XORF    %I1.5
ST      %Q2.5
```

Język *ST*

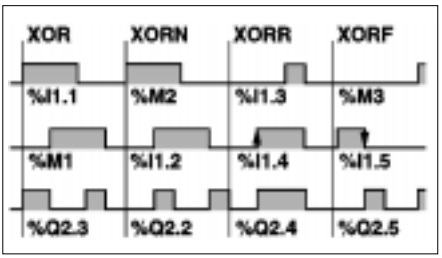
```
%Q2.3 := %I1.1   XOR %M1 ;
%Q2.2 := %M2     XOR (NOT %I1.2) ;
%Q2.4 := %I1.3   XOR (RE %I1.4) ;
%Q2.5 := %M3     XOR (FE %I1.5) ;
```

Uwaga: Stosowanie nawiasów nie jest obowiązkowe, ale czyni program bardziej przejrzystym.

Dozwolone argumenty

Kod	Argument
XOR	%I,%Q,%M,%S,%BLK,%•:Xk,%Xi
XORN	%I,%Q,%M,%S,%BLK,%•:Xk,%Xi
XORR	%I,%Q,%M
XORF	%I,%Q,%M

Diagram czasowy

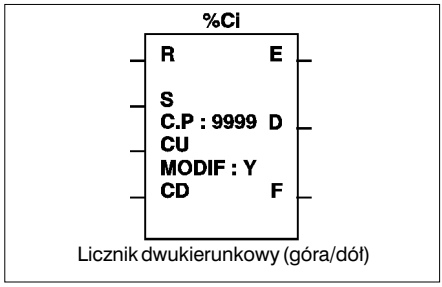


1.3 Standardowe bloki funkcyjne

1.3-1 Zasady programowania

Bloki funkcyjne posługują się bitami obiektowymi i słowami specjalnymi.

Standardowe bloki funkcyjne są wstępnie zaprogramowane w sterowniku, stąd też zajmują one ściśle określony obszar pamięci użytkowej sterownika.
Stąd, w celu optymalizacji wykorzystania pamięci, musi być zdefiniowana liczba oraz rodzaj stosowanych bloków funkcyjnych. Dokonuje się tego, z uwzględnieniem limitów narzuconych przez system, za pomocą edytora danych lub konfiguracji.



Licznik dwukierunkowy (góra/dół)

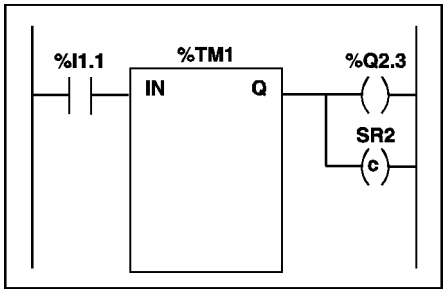
System zawiera sześć standardowych bloków funkcyjnych:

Rodzaj bloku	Max TSX 37	Max TSX 57	Rozdział
Zegar *Timer* %TMi	64 (1)	255 (1)	1.3-2
Licznik dwukierunkowy *Up/down* %Ci	32	255	1.3-3
Przerzutnik mono *Monostable* %MNi	8	255	2.2-1
Rejestr *Register* %Ri	4	255	2.2-2
Bęben *Drum controller* %DRi	8	255	2.2-3
Zegar Timer (Seria 7) %Ti	64 (1)	255 (1)	2.2-4

(1) Całkowita liczba zegarów %TMi + %Ti nie może przekraczać 64 dla sterowników TSX 37 i 255 dla sterowników TSX 57.

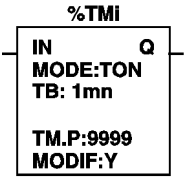
Każdy blok posiada:

- Wejścia (np. IN) służące do sterowania działaniem bloku.
- Wyjścia (np. Q) które odwzorowują stan bloku. Każdemu wyjściu przyporządkowany jest bit wyjściowy (np. %TM1.Q), który może być testowany przez program użytkowy. Dodatkowo, każde wyjście może sterować sprzężeniami (np. %Q2.3 i SR2).
- Parametry umożliwiające zaadaptowanie bloku do aplikacji (nastawianie wartości, podstawa czasu, itd.)



Parametry bloków funkcyjnych (nastawianie wartości *Preset*, podstawa czasu *Time base*, itd.) są wyświetlane we wnętrzu bloku. W języku *LIST* bloki standardowe programuje się przy pomocy odpowiednich instrukcji (patrz część A, rozdział 3.2-6).

1.3-2 Zegar *Timer* %TMi



Blok zegara

Zegar może pracować w trzech trybach:

- **TON**: jest to tryb służący do kontrolowania akcji z opóźnionym uaktywnieniem (*on-delay*). Opóźnienie to można programować i modyfikować za pośrednictwem terminala.
- **TOF**: jest to tryb służący do kontrolowania akcji z opóźnieniem dezaktywacji (*off-delay*). Opóźnienie to można programować i modyfikować za pośrednictwem terminala.
- **TP**: jest to tryb umożliwiający wygenerowanie impulsu o ściśle określonym czasie trwania. Czas trwania impulsu można programować i modyfikować za pośrednictwem terminala.

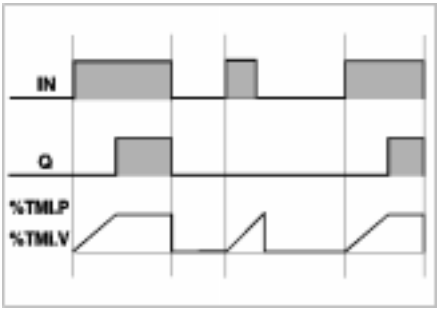
Charakterystyka

Liczba zegarów	**%TMi**	0 do 63 dla TSX 37, 0 do 254 dla TSX 57
Tryb pracy	**TON** **TOF** **TP**	• zegar opóźniający *on-delay* (domyślnie) • zegar opóźniający *off-delay* • przerzutnik *monostable*
Podstawa czasu *Time base*	**TB**	1min (domyślnie), 1s, 100ms, 10ms (max 16 zegarów dla 10ms). Im mniejsza wartość podstawy czasu, tym większa dokładność.
Wartość bieżąca *Current value*	**%TMi.V**	Słowo, którego wartość rośnie od 0÷%TMi.P, gdy zegar pracuje. Może ona być czytana i testowana, lecz nie może być zapisywana przez program (1).
Wartość nastawiona *Preset*	**%TMi.P**	0-%TMi.P-9999. Słowo, które może być czytane,testowane i zapisywane przez program. Domyślnie, przyjmuje ono wartość 9999. Generowany impuls lub wartość opóźnienia jest równa %TMi.P x TB.
Regulacja za pom. terminala *Adjust* (MODIF)	**Y/N**	Y: wartość nastawiona %TMi.P może być modyfikowana w trybie regulacji (*adjust mode*). N: nie można modyfikować wartości.
Wejście (instrukcja)	**IN**	Zegar startuje przy zboczu rosnącym sygnału (tryb TON lub TP) lub zboczu opadającym (tryb TOF).
Wyjście	**Q**	Bit %TMi.Q przyjmuje wartość 1 w warunkach zależących od typu funkcji (TON, TOF lub TP).

(1) Wartość %TMi.V może być modyfikowana za pośrednictwem terminala.

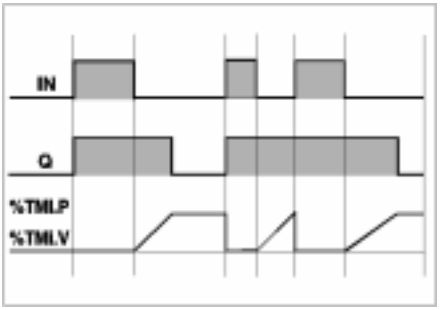
Zegar opóźniający (*on-delay*): tryb TON

Zegar uruchamia się w momencie pojawienia się zbocza rosnącego na wejściu IN: jego wartość %TMi.V rośnie od 0 do %TMi.P o jednostkę, przy każdym impulsie podstawy czasu TB. Bit wyjściowy %TMi.Q zmienia stan na 1, gdy wartość bieżąca osiąga wartość %TMi.P i utrzymuje wartość 1 tak długo, jak długo wejście IN ma stan 1.
Pojawienie się 0 na wejściu IN powoduje zatrzymanie zegara, nawet, gdy jego wartość ciągle jeszcze się zmieniała: %TMi.V przyjmuje wartość 0.



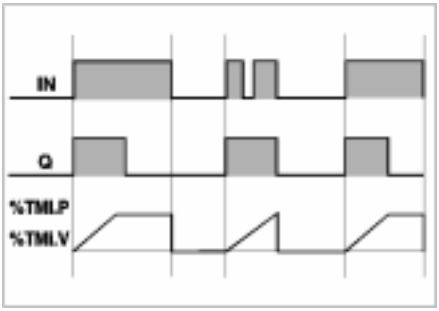
Zegar opóźniający (*off-delay*): tryb TOF

Wartość bieżąca %TMi.V przyjmuje wartość 0 przy pojawieniu się zbocza rosnącego sygnału na wejściu IN (nawet jeśli zegar jeszcze pracował). Zegar uruchamia się w momencie wykrycia zbocza opadającego sygnału na wejściu IN.
Wartość bieżąca rośnie do wartości %TMi.P zwiększana o 1, przy każdym impulsie podstawy czasu TB. Bit wyjściowy %TMi.Q zmienia stan na 1, w momencie wykrycia zbocza rosnącego sygnału na wejściu IN, a zegar kasuje się do 0 po osiągnięciu wartości %TMi.P.



Zegar pełniący funkcję przerzutnika: tryb TP

Zegar uruchamia się po wystąpieniu zbocza rosnącego na wejściu IN (o ile zegar nie wystartował już wcześniej): jego wartość bieżąca %TMi.V rośnie od 0 do %TMi.P o 1, przy każdym impulsie podstawy czasu TB. Bit wyjściowy %TMi.Q zmienia stan na 1, gdy zegar się uruchamia i przyjmuje wartość 0, gdy wartość bieżąca osiąga wartość %TMi.P.
Gdy wejście IN i wyjście %TMi.Q mają stan 0, to TMi.V również ma wartość 0. Ten stan nie podlega kasowaniu.



Programowanie i konfigurowanie

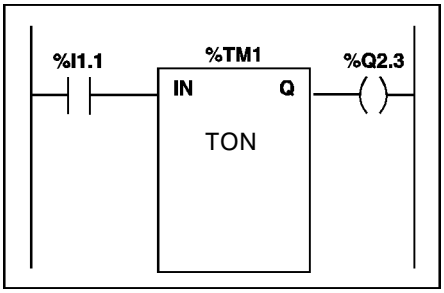
Zegary programuje się w jednolity sposób, niezależnie od wybranego trybu pracy. Wyboru trybu pracy (TON, TOF lub TP) dokonuje się za pomocą edytora zmiennych.

- **Konfigurowanie**

W edytorze zmiennych należy zdefiniować następujące parametry:
- tryb pracy *Mode*: TON, TOF lub TP,
- podstawę czasu *TB*: 1min, 1s, 100ms lub 10ms,
- wartość nastawioną %TMi.P: z przedziału od 0 do 9999.
- możliwość modyfikowania wartości *MODIF*: Y lub N.

- **Programowanie**

Język *Ladder*



Język *List*

```
LD      %I1.1
IN      %TM1
LD      %TM1.Q
ST      %Q2.3
```

Język *ST*

```
IF RE %I1.1 THEN
    START %TM1 ;
ELSIF FE %I1.1 THEN
    DOWN %TM1 ;
END_IF ;
%Q2.3 := %TM1.Q ;
```

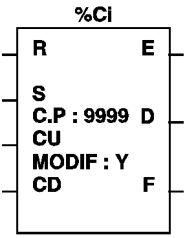
Instrukcja START %TMi powoduje wygenerowanie zbocza narastającego na wejściu IN bloku zegara.
Instrukcja DOWN %TMi powoduje wygenerowanie zbocza opadającego na wejściu IN bloku zegara.

Sytuacje szczególne

- **Reakcja na zimny start:** (%S0=1) skasowanie wartości bieżącej do 0, wystawienie 0 na wyjściu %TMi.Q i skasowanie wartości nastawionej (*preset value*) do wartości zdefiniowanej w konfiguracji.
- **Reakcja na gorący start:** (%S1=1) nie ma żadnego wpływu na wartość bieżącą zegara oraz wartość nastawioną. W czasie zaniku zasilania wartość bieżąca nie ulega zmianie.
- **Reakcja na zatrzymanie sterownika, dezaktywację zadania lub przerwanie (*break point*):** nie powodują zamrożenia wartości bieżącej.
- **Reakcja na skok w programie:** fakt, że instrukcje, w których zaprogramowany jest blok zegara nie są wykonywane (program "przeskoczył" do innej części) nie powoduje zamrożenia wartości bieżącej %TMi.V, która nadal rośnie aż do osiągnięcia wartości %TMi.P. Podobnie, bit %TMi.Q przyporządkowany wyjściu Q bloku zegara zachowuje swoje właściwości, w związku z czym może być testowany przez inną instrukcję. Jednakże, wyjście bezpośrednio podłączone do wyjścia bloku nie jest ani uaktywniane, ani skanowane (odczytywane) przez sterownik.
- **Testowanie bitu %TMi.Q:** zaleca się testowanie bitu %TMi.Q w programie tylko jeden raz.
- **Reakcja na zmianę wartości nastawionej %TMi.P:** modyfikowanie wartości nastawionej czy to za pomocą instrukcji, czy podczas pracy w trybie regulacji (*adjust mode*) przynosi efekt tylko wtedy, kiedy zegar zoatanie następnie uruchomiony. Zmiana wartości nastawionej w edytorze zmiennych jest uwzględniana dopiero po zimnym starcie (%S0=1).

1.3-3 Licznik dwukierunkowy %Ci

Licznik dwukierunkowy (*up/down counter*) służy do zliczania (*upcount*) lub odliczania (*downcount*) zdarzeń.
Te dwie operacje mogą być wykonywane przez licznik jednocześnie.



Blok licznika dwukierunkowego

Charakterystyka

Liczba liczników	**%Ci**	0 do 31 dla TSX 37, 0 do 254 dla TSX 57
Wartość bieżąca *Current*	**%Ci.V**	Słowo, którego wartość jest zmniejszana lub zwiększana w zależności od wejść CU i CD. Może ono być odczytywane lub testowane, lecz nie może być zapisywane przez program (1).
Wartość nastawiona *Preset*	**%Ci.P**	0-%Ci.P-9999. Słowo, które może być czytane, testowane i zapisywane. Domyślnie: 9999
Regulacja za pom. terminala (MODIF)	**Y/N**	Y: wartość nastawiona może być zmieniana w trybie regulacji (*adjust mode*). N: brak możliwości zmiany w trybie regulacji.
Wejście kasowania *Reset* (instrukcja)	**R**	Stan 1 : %Ci.V = 0.
Wejście nastawiania *Preset* (instrukcja)	**S**	Stan 1: %Ci.V = %Ci.P.
Wejście zliczające *Up* (instrukcja)	**CU**	Powoduje zwiększenie wartości %Ci.V przy zboczu rosnącym sygnału.
Wejście odliczające *Down* (instrukcja)	**CD**	Powoduje zmniejszenie wartości bieżącej %Ci.V przy zboczu rosnącym sygnału.
Wyjście *Licznik pusty*	**E** (Empty)	Bit %Ci.E=1 gdy podczas odliczania nastąpi zmiana wartości %Ci.V z 0 na 9999 (stan 1, gdy %Ci.V osiąga wartość 9999; jest kasowany do 0, gdy licznik kontynuuje odliczanie).(2)
Wyjście *Osiągnięcie wartości nastawionej*	**D** (Done)	Bit %Ci.D=1 gdy %Ci.V=%Ci.P.
Wyjście *Przepełnienie*	**F** (Full)	Bit %Ci.F =1 gdy %Ci.V zmienia wartość z 9999 na 0 (stan 1 gdy %Ci.V osiąga 0; jest kasowany do 0 gdy licznik kontynuuje zliczanie).

(1) Wartość %Ci.V może być modyfikowana za pomocą terminala.
(2) W przypadku, gdy licznik jest pełen lub jest pusty, bit %S18 zmienia stan na 1.

Sposób wykonywania operacji

- **Zliczanie *Upcount***: pojawienie się na wejściu CU zbocza rosnącego powoduje zwiększenie wartości bieżącej (o jednostkę). W momencie, gdy wartość bieżąca zrównuje się z wartością nastawioną %Ci.P, bit wyjściowy %Ci.D (osiągnięcie wartości nastawionej) przyporządkowany wyjściu D zmienia stan na 1. Bit wyjściowy %Ci.F (przepełnienie licznika) zmienia stan na 1, gdy wartość bieżąca %Ci.V zmienia się z 9999 na 0 i jest kasowany do 0 jeśli licznik kontynuuje zliczanie.
- **Odliczanie *Downcount***: pojawienie się na wejściu CD zbocza rosnącego powoduje zmniejszenie wartości bieżącej %Ci.V (o jednostkę). Bit wyjściowy %Ci.E (licznik pusty) zmienia stan na 1, gdy wartość bieżąca %Ci.V zmienia się z 0 na 9999 i jest kasowany do 0 jeśli licznik kontynuuje odliczanie.
- **Licznik dwukierunkowy *Up/down***: liczenie w dwu kierunkach wymusza konieczność kontrolowania stanów obydwu wejść CU i CD. Te dwa wejścia są kontrolowane kolejno. Jeżeli jednocześnie, obydwa wejścia mają stan 1, to wartość bieżąca pozostaje nie zmieniona.
- **Kasowanie *Reset***: gdy na wejściu R pojawia się 1, to wartość bieżąca %Ci.V jest kasowana do 0 (wymuszenie), a wyjścia %Ci.E, %Ci.D oraz %Ci.F przyjmują wartość 0. Wejście *Reset* ma najwyższy priorytet.
- **Nastawianie *Preset***: gdy na wejściu S (*preset*) pojawia się 1, a wejście R (*reset*) ma stan 0, to wartość bieżąca %Ci.V przyjmuje wartość %Ci.P, a na wyjściu %Ci.D jest wystawiana 1.

Uwaga

Przy kasowaniu (wejście R lub instrukcja R):

- w języku *Ladder*, do historii stanów (*log*) wejść CU i CD zapisywane są bieżące stany tych wejść.
- w języku *LIST* oraz *ST* historia stanów wejść CU i CD nie jest aktualizowana. Dla każdego z tych wejść zachowywana jest wartość sprzed operacji kasowania.

Sytuacje szczególne

- **Reakcja na zimny start:** (%S0=1)
  - wartość bieżąca %Ci.V jest kasowana do zera,
  - bity wyjść %Ci.E, %Ci.D i %Ci.F są kasowane do zera,
  - przywracana jest wartość nastawiona, która została zdefiniowana w konfiguracji.
- **Reakcja na gorący start (%S1=1), zatrzymanie sterownika, dezaktywację zadania lub przerwanie:** nie mają wpływu na bieżącą wartość licznika (%Ci.V).
- **Reakcja na zmianę wartości nastawionej %Ci.P:** zmiana wartości nastawionej czy to za pomocą instrukcji, czy też w trybie regulacji przynosi efekt w momencie, gdy następuje przetworzenie bloku przez aplikację (uaktywnienie jednego z wejść).

Konfigurowanie i programowanie - przykład

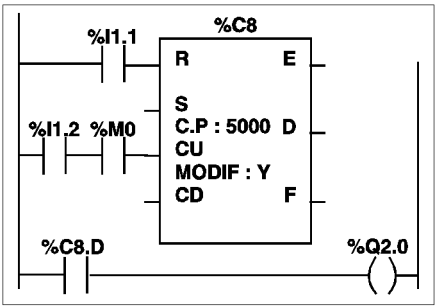
Zadanie polega na liczeniu elementów (do 5000 sztuk). Każdy impuls na wejściu %I1.2 (gdy bit wewnętrzny %M0 ma wartość 1) powoduje zwiększanie wartości licznika zliczającego %C8 aż do osiągnięcia wartości nastawionej (bit %C8.D=1). Licznik jest kasowany za pośrednictwem wejścia %I1.1.

- **Konfigurowanie**

  W edytorze zmiennych należy zdefiniować następujące parametry:
  - nastawienie %Ci.P na 5000 (dla tego przykladu),
  - MODIF: Y.

- **Programowanie**

Język *Ladder*



Język *List*

```
LD      %I1.1
R       %C8
LD      %I1.2
AND     %M0
CU      %C8
LD      %C8.D
ST      %Q2.0
```

Język *ST*

```
IF %I1.1 THEN
    RESET %C8 ;
END_IF ;
%M1 := %I1.2   AND %M0 ;
IF RE %M1 THEN
    UP %C8 ;
END_IF ;
%Q2.0 := %C8.D;
```

W języku *ST*, do programowania licznika dwukierunkowego służą 4 instrukcje:

- RESET %Ci : kasowanie wartości bieżącej,
- PRESET %Ci : zastąpienie wartości bieżącej wartością nastawioną,
- UP %Ci : zwiększenie wartości bieżącej,
- DOWN %Ci : zmniejszenie wartości bieżącej.

Jeżeli w programie napisanym w języku *ST* użyte są instrukcje UP i DOWN, to następuje skasowanie historii stanów (*log*) wejść CU i CD. Dlatego też, w przypadku tych dwu instrukcji, użytkownik musi samodzielnie zarządzać tymi sygnałami.

1.4 Operacje numeryczne na liczbach całkowitych

1.4-1 Wprowadzenie

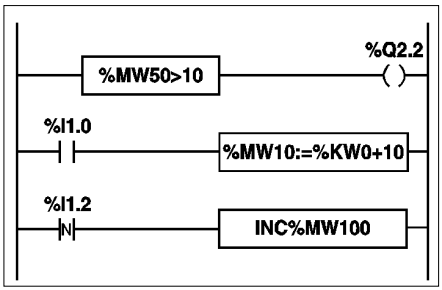
Opisane w niniejszym rozdziale operacje numeryczne dotyczą następujących obiektów:
- tablic bitowych,
- słów,
- słów podwójnej precyzji.

Instrukcje odnoszące się do pozostałych obiektów zostały opisane w rozdziale zatytułowanym "Instrukcje i funkcje złożone".

Język *Ladder*

Instrukcje numeryczne zapisuje się w blokach:
- porównywania (*comparison*) umieszczanych w strefie warunków (*test zone*),
- operacyjnych (*operation*), umieszczanych w strefie akcji (*action zone*).

Bloki te mogą zawierać:
- wyrażenia proste, np: Arg3:=Arg1+Arg2,
- wyrażenia złożone, np: Arg5:=(Arg1+Arg2)*Arg3-Arg4.



Język *List*

Instrukcje umieszcza się w nawiasach kwadratowych.
Wykonanie instrukcji następuje, gdy logiczny wynik poprzedzającej ją instrukcji testowania warunków jest równy 1.

```
LD      [%MW50>10]
ST      %Q2.2
LD      %I1.0
[%MW10:=%KW0+10]
LDF     %I1.2
[INC %MW100]
```

Język *ST*

Instrukcje numeryczne zapisuje się bezpośrednio.
Zastosowanie pętli warunkowej IF umożliwia uzależnienie wykonania instrukcji numerycznej od wyniku wyrażenia logicznego.

```
%Q2.2 := %MW50 > 10 ;
IF %I1.0 THEN
    %MW10 := %KW0 + 10 ;
END_IF ;
IF FE %I1.2 THEN
    INC %MW100 ;
END_IF ;
```

Lista argumentów:

Tablice bitowe

Skrót	Pełen adres	Rodzaj słów	Dostęp
%M:L	%Mi:L	tablica bitów wewnętrznych	R/W
%I:L	%Ixy.i:L	tablica bitów wejściowych	R/W
%Q:L	%Qxy.i:L	tablica bitów wyjściowych	R/W
%Xi:L	%Xi:L lub %Xj.i:L	tablica bitów kroków	R

Słowa pojedyncze

Skrót	Pełen adres	Rodzaj słów	Dostęp	Indeks
Immed. val.	-	wartości bezpośrednie	R	-
%MW	%MWi	słowo wewnętrzne	R/W	%MWi[index]
%KW	%KWi	wewnętrzna stała	R	%KWi[index]
%SW	%SWi	słowo systemowe	R/W (1)	-
%IW	%IWxy.i(.r)	słowo wejściowe	R	-
%QW	%QWxy.i(.r)	słowo wyjściowe	R/W	-
%NW	%NW{j}k	słowo wspólne	R/W	-
%BLK	np. %TMi.P	słowo wydzielone z bloku DFB lub standardowego	R/W (2)	-
%Xi.T	%Xi.T	czas aktywności kroku	R	%Xi.T [index]

(1) Możliwość zapisu zależy od *i.* (2) Możliwość zapisu zależy od rodzaju słowa, np. wartości nastawione (%Ci.P może być zapisywana, mimo, że wartość %Ci.V może być tylko odczytywana).

Słowa podwójnej precyzji

Skrót	Pełen adres	Rodzaj słów podwójnych	Dostęp	Indeks
Immed. val.	-	wartości bezpośrednie	R	-
%MD	%MDi	wewnętrzne słowo podw.	R/W	%MDi[index]
%KD	%KDi	wewnętrzna stała podw.	R	%KDi[index]
%SD	%SDi	systemowe słowo podw.	R/W (1)	-
%ID	%IDxy.i(.r)	wejściowe słowo podw.	R	-
%QD	%QDxy.i(.r)	wyjściowe słowo podw.	R/W	-

(1) Tylko słowo podwójne %SD18.

Uwagi

Pozostałe słowa takie, jak %MWxy.i %KWxy.i oraz %MDxy.i %KDxy.i, zachowują się odpowiednio: jak słowa i słowa podwójne %MWi %KWi oraz %MDi %KDi.

Niejawna (*Implicit*) konwersja: słowa <–> słowa podwójne

PL7 pozwala na mieszanie operacji na słowach i słowach podwójnej precyzji. Konwersja z jednego formatu na drugi dokonuje się niejawnie. Operacja dotycząca słowa podwójnego lub kilku wartości bezpośrednich jest automatycznie wykonywana w formacie podwójnym.

1.4-2 Instrukcje porównywania *Comparison*

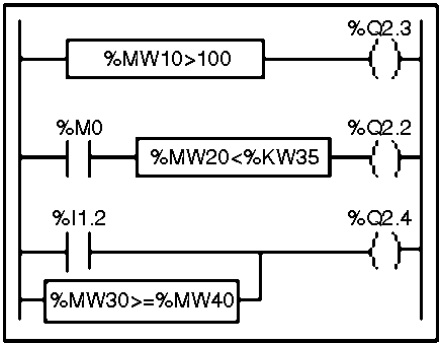
Instrukcje porównywania służą do porównywania dwóch argumentów.

> : sprawdzenie, czy argument 1 jest większy niż argument 2.
≥ : sprawdzenie, czy argument 1 wiekszy lub równy argumentowi 2.
< : sprawdzenie, czy argument 1 jest mniejszy niż argument 2.
≤ : sprawdzenie, czy argument 1 jest mniejszy lub równy argumentowi 2.
= : sprawdzenie, czy argument 1 jest równy argumentowi 2.
<> : sprawdzenie, czy argument 1 jest różny od argumentu 2.

Wynikiem sprawdzenia jest 1, gdy warunek jest spełniony (prawda).

Struktura

Język *Ladder*



Bloki porównywania umieszcza się w strefie warunków (*test zone*).

Język *List*

```
LD      [%MW10 > 100]
ST      %Q2.3
LD      %M0
AND     [%MW20 < %KW35]
ST      %Q2.2
LD      %I1.2
OR      [%MW30 >= %MW40]
ST      %Q2.4
```

Operacja porównywania jest zapisywana w nawiasie kwadratowym umieszczanym za funkcjami LD, AND lub OR.

Język *ST*

```
%Q2.3 := %MW10 > 100 ;
%Q2.2 := %M0 AND (%MW20 < %KW35) ;
%Q2.4 := %I1.2 OR (%MW30 >= %MW40) ;
```

Uwaga: Używanie nawiasów nie jest obowiązkowe, ale czyni ono program bardziej przejrzystym.

Składnia

Operatory: **>,>=,<,<=,=,<>**

```
Arg1 Operator Arg2
```

Argumenty

Typ	Argument 1 i 2 (Arg1 i Arg2)
Słowa indeksowalne	%MW,%KW
Słowa nieindeksowalne	Wartości bezpośrednie,%IW,%QW,%SW,%NW,%BLK, %Xi.T wyrażenia numeryczne
Indeksowalne słowa podwójne	%MD,%KD
Nieindeksowalne słowa podw.	Wartości bezpośrednie,%ID,%QD,%SD,wyrażenia num.

Uwagi

- W języku *Ladder* operacja porównywania może być również realizowana za pomocą pionowego bloku porównywania (patrz część B, rozdział 2.3).
- W języku *List* instrukcje porównywania mogą być zapisywane w nawiasach.

1.4-3 Instrukcje przypisania *Assignment*

Instrukcje przypisania służą do zapisywania argumentu Arg2 w argumencie Arg1.

Sk adnia: Arg1:=Arg2 <=> Arg2->Arg1

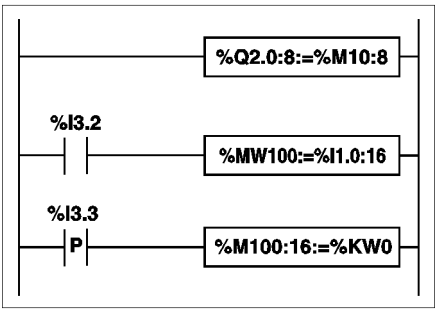
Operacje przypisania mogą być przeprowadzane na następujących obiektach:
- tablicach bitowych,
- słowach i słowach podwójnych.

W obrębie jednego bloku można zastosować kilka instrukcji przypisania:
Arg1:=Arg2:=Arg3:=Arg4:=...

Przypisywanie tablic bitowych (patrz: opis tablic bitowych w rozdz. 1.2-6, części A)
Na tablicach bitowych można wykonywać następujące operacje przypisania:
- tablica bitowa -> tablica bitowa Przykład 1
- tablica bitowa -> słowo lub słowo podwójne (indeksowane) Przykład 2
- słowo lub słowo podwójne (indeksowane) -> tablica bitowa Przykład 3

Struktura

Język *Ladder*



Język *List*

```
LD      TRUE
[%Q2.0:8:= %M10:8]     Przyk ad1

LD      %I3.2
[%MW100:= %I1.0:16 ]   Przyk ad2

LDR     %I3.3
[%M100:16:=%KW0]       Przyk ad3
```

Język *ST*

```
%Q2.0:8 := %M10:8 ;          Przykład 1
IF %I3.2 THEN
    %MW100 := %I1.0:16 ;Przykład 2
END_IF ;
IF RE %I3.3 THEN
    %M100:16 := %KW0 ;  Przykład 3
END_IF ;
```

Składnia

Operator **:=**

Arg1:=Arg2

Operands

Typ	**Argument 1 (Arg1)**	**Argument 2 (Arg2)**
Tablica bitowa	%M:L,%Q:L,%I:L	%M:L,%Q:L,%I:L, %Xi:L
Słowa indeksowalne	%MW	%MW,%KW
Słowa nieindeksowalne	%QW,%SW,%NW %BLK	Wart. bezp.,%IW,%QW,%SW %NW,%BLK,%Xi.T,Wyr.num.
Indeks. słowa podwójne	%MD	%MD,%KD
Nieindeks. słowa podwójne	%QD,%SD,	Wart. bezp.,%ID,%QD,%SD Wyrażenie numeryczne

Reguły dotyczące stosowania

- Tablice bitowe: źródłowa i docelowa nie muszą mieć koniecznie takich samych długości. Jeżeli tablica źródłowa jest dłuższa od tablicy docelowej, to przeniesione zostaną tylko bity mniej znaczące. W przypadku odwrotnym tablica docelowa uzupełniana jest zerami.
- Przypisywanie tablica bitowa -> słowo (lub słowo podwójne): bity z tablicy są przenoszone do słowa (do słowa o niższym priorytecie z dwóch słów tworzących słowo podwójne) począwszy od prawej strony (pierwszy bit tablicy jest zapisywany do bitu 0 słowa). Bity słowa, dla których nie zostały przeniesione wartości (długość < 16 lub 32) przyjmują wartości 0.
- Przypisanie słowo -> tablica bitowa: bity słowa są poddawane transferowi począwszy od prawej strony (bit 0 słowa jest zapisywany jako pierwszy bit tablicy).

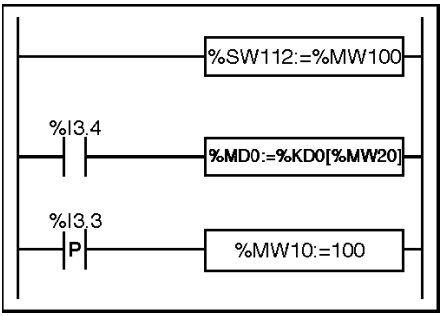
Przypisywanie słów

W odniesieniu do słów mogą być stosowane następujące operacje przypisania:

- słowo (indeksowane) -> słowo (indeks.) lub podwójne (indeks.) Przykład 1
- słowo podwójne (indeks.) -> słowo (indeks.) lub podwójne (indeks.) Przykład 2
- wartość bezp. -> słowo (indeks.) lub słowo podwójne (indeks.) Przykład 3

Struktura

Język *Ladder*



Język *List*

```
LD        TRUE
[%SW112 := %MW100]      Przyk ad 1
LD        %I3.2
[%MD0:= %KD0[%MW20] ]  Przyk ad 2
```

Język *ST*

```
IF RE %I3.3 THEN
    %MW10 := 100 ;        Przyk ad 3
END_IF ;
```

Składnia

Operator **:=**

Arg1:=Arg2

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MW	%MW,%KW
Słowa nieindeksowalne	%QW,%SW,%NW, %BLK	Wart. bezp.,%IW,%QW,%SW %NW,%BLK,%Xi.T, Wyr. num.
Indeksowalne słowa podwójne	%MD	%MD,%KD
Nieindeksowalne słowa podw.	%QD,%SD,	Wart. bezp.,%ID,%QD,%SD Wyrażenie numeryczne

Uwaga

Konwersja słowo <--> słowo podwójne odbywa się niejawnie podczas wykonywania operacji przypisania słowo podwójne --> słowo. Jeżeli wartość słowa podwójnego nie mieści się w słowie, to bit %S18 przyjmuje wartość 1.

Istnieje możliwość stosowania wielokrotnego przypisania.
Przykład: %MW0 := %MW2 := %MW4
Należy zwrócić uwagę, że w przykładzie %MD14 := %MW10 := %MD12, przypisanie %MD14 := %MD12 nie odpowiada prawdzie, gdyż podczas przypisania słowa podwójnego do słowa %MW10 słowo o wyższym priorytecie zostanie zignorowane (w efekcie konwersji słowa podwójnego na słowo pojedyncze).

1.4-4 Instrukcje arytmetyczne na liczbach całkowitych

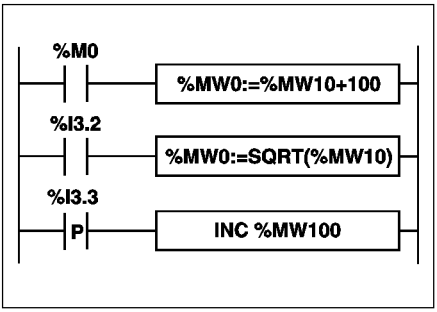
Opisane poniżej instrukcje umożliwiają wykonywanie operacji arytmetycznych pomiędzy dwoma argumentami lub na jednym argumencie.

+ : dodawanie dwóch argumentów
- : odejmowanie dwóch argumentów
***** : mnożenie dwóch argumentów
/ : dzielenie dwóch argumentów
REM : reszta z dzielenia dwóch argumentów

SQRT: pierwiastek z argumentu
INC : zwiększenie wartości argumentu
DEC : zmniejszenie wartości argumentu
ABS : moduł argumentu

Struktura

Język *Ladder*



Język *List*

```
LD      %MW0
[%MW0 := %MW10 + 100]

LD      %I3.2
[%MW0 := SQRT(%MW10)]

LDR     %I3.3
[INC %MW100]
```

Język *ST*

```
IF %M0 THEN
    %MW0 := %MW10 + 100 ;
END_IF ;
IF %I3.2 THEN
    %MW0 := SQRT (%MW10) ;
END_IF ;
IF RE %I3.3 THEN
    INC %MW100 ;
END_IF ;
```

Składnia

Operatory

- **+**, **-**, *****, **/**, **REM** — Arg1:=Arg2 Operator Arg3
- **SQRT**, **ABS** — Arg1:=Operator(Arg2)
- **INC**, **DEC** — Operator Arg1

Argumenty

Typ	Argument 1 (Arg1)	Argumenty 2 i 3 (Arg2 i 3)
Słowa indeksowalne	%MW	%MW,%KW
Słowa nieindeksowalne	%QW,%SW,%NW, %BLK	Wart. bezp.,%IW,%QW,%SW %NW,%BLK,%Xi.T,Wyr.num.
Indeksowalne słowa podw.	%MD	%MD,%KD
Nieindeks. słowa podwójne	%QD,%SD,	Wart. bezp.,%ID,%QD,%SD Wyrażenie numeryczne

Uwaga:

Operacji INC oraz DEC nie można używać w wyrażeniach numerycznych.

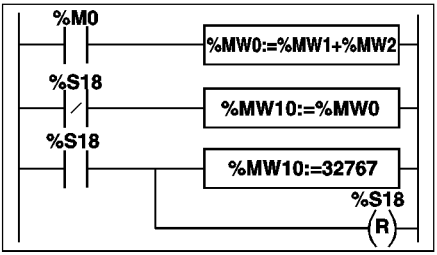
Reguły dotyczące operacji

- **Dodawanie: przekroczenie limitu**

Jeżeli suma przekroczy podane ograniczenia:
-32768 lub +32767 dla argumentu pojedynczej precyzji,
-2 147 483 648 lub +2 147 483 647 dla argumentu podwójnej precyzji,
to bit %S18 (przepełnienie) przyjmuje wartość 1. Wynik operacji jest wtedy nieznaczący. Stanem bitu %S18 zarządza program.

Przykład:

Język *Ladder*



Język *List*

```
LD          %M0
[%MW0 := %MW1+ %MW2]
LDN         %S18
[%MW10 := %MW0 ]
LD          %S18
[%MW10 := 32767 ]
R           %S18
```

Język *ST*

```
IF %M0 THEN
    %MW0 := %MW1 + %MW2 ;
END_IF ;
IF %S18 THEN
    %MW10 := 32767 ; RESET %S18 ;
ELSE
    %MW10 := %MW0 ;
END_IF ;
```

Jeśli %MW1=23241 i %MW2=21853, to rzeczywisty wynik (45094) nie może być wyrażony w słowie 16-bitowym. Bit %S18 przyjmuje wartość 1, a uzyskany wynik (-20442) jest niewłaściwy. W tym przykładzie: gdy wynik działania jest większy niż 32767, to przyjmuje się 32767.

- **Mnożenie:**
  Przekroczenie limitu podczas wykonywania operacji.
  Jeżeli wynik działania przekroczy pojemność słowa wynikowego, to bit %S18 (przepełnienie) przyjmuje wartość 1, a wynik działania jest nieznaczący.
- **Dzielenie/reszta:**
  Dzielenie przez 0.
  Jeżeli dzielnikiem jest 0, to wykonanie operacji dzielenia jest niemożliwe i bit systemowy %S18 przyjmuje wartość 1. Wynik operacji jest nieprawidłowy.
  Przekroczenie limitu podczas wykonywania operacji.
- **Obliczanie pierwiastka kwadratowego:**
  Operacji obliczania pierwiastka kwadratowego dokonuje się tylko na liczbach dodatnich tak, że wynik jest zawsze dodatni. Jeżeli argument jest ujemny, to bit systemowy %S28 przyjmuje wartość 1, a wynik operacji jest nieprawidłowy.

Uwaga:
- Jeżeli wynik operacji nie jest liczbą całkowitą (w przypadku dzielenia lub obliczania pierwiastka kwadratowego), to wynik jest zaokrąglany w dół do najbliższej liczby całkowitej.
- Znak reszty (REM) jest taki sam jak znak licznika.
- Za zarządzanie bitem systemowym %S18 odpowiada program użytkowy. Sterownik nadaje mu wartość 1, którą program musi skasować, by mógł on być powtórnie użyty (patrz przykład ze strony poprzedniej).

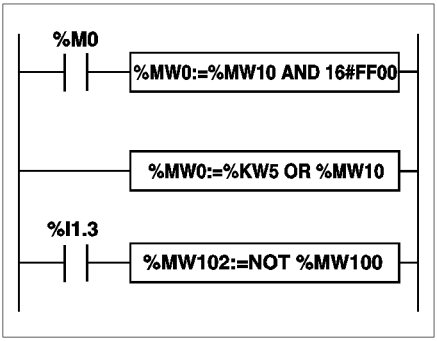
1.4-5 Instrukcje logiczne

Wymienione poniżej instrukcje umożliwiają wykonywanie operacji logicznych na dwóch argumentach lub na jednym argumencie.

AND : Koniunkcja (bit po bicie) dwóch argumentów
OR : Alternatywa (bit po bicie) dwóch argumentów
XOR : Nierównoważność (bit po bicie) dwóch argumentów
NOT : Logiczne dopełnienie (bit po bicie) argumentu

Struktura

Język *Ladder*



Język *List*

```
LD      %M0
[%MW0 := %MW10 AND 16#FF00]

LD      TRUE
[%MW0 := %KW5 OR %MW10]

LD      %I1.3
[%MW102:= NOT %MW100]
```

Język *ST*

```
IF %M0 THEN
    %MW0 := %MW10 AND 16#FF00 ;
END_IF ;
%MW0 := %KW5 OR %MW10 ;
IF %I1.3 THEN
    %MW102 := NOT %MW100 ;
END_IF ;
```

Składnia

Operatory

- **AND**, **OR**, **XOR**, Arg1:=Arg2 Operator Arg3
- **NOT**, Arg1:=NOT Arg2

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 i 3 (Arg2 i 3)
Słowa indeksowalne	%MW	%MW,%KW
Słowa nieindeksowalne	%QW,%SW,%NW, %BLK	Wart. bezp.,%IW,%QW,%SW NW,%BLK,%Xi.T,Wyr. num.
Indeksowalne słowa podw.	%MD	%MD,%KD
Nieindeksowalne słowa podw.	%QD,%SD	Wart. bezp.,%ID,%QD,%SD Wyrażenie numeryczne

1.4-6 Wyrażenia numeryczne

Wyrażenia numeryczne składają się z kilku argumentów numerycznych i, opisanych poniżej, operatorów logicznych lub arytmetycznych.

Przykład: %MW25 * 3 - SQRT(%MW10) + %KW8* (%MW15 + %MW18) AND 16#FF

Liczba argumentów i operatorów w wyrażeniach arytmetycznych nie jest ograniczona.

Wyrażenia numeryczne w odniesieniu do liczb całkowitych
Argumenty w danym wyrażeniu numerycznym mogą być zarówno pojedynczej jak i podwójnej precyzji.
%MW6 * %MW15 + SQRT(%DW6) /(%MW149[%MW8]) + %KD29) AND 16#FF

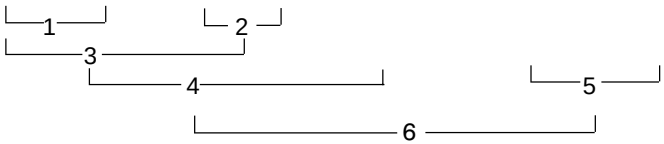
Argument lub operacja na argumencie może być poprzedzona znakiem + lub - (domyślnie przyjmuje się znak +).
Przykład: SQRT (%MW5) * - %MW9

W wyrażeniach arytmetycznych można używać wszystkich słów, niektóre mogą być indeksowane.

Priorytet wykonywania instrukcji
W wyrażeniach numerycznych instrukcje są wykonywane w następującej kolejności:

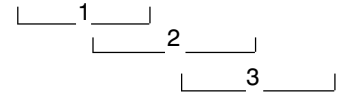
1 ->	2 ->	3 ->	4 ->	5 ->	6 ->	7 ->	8
operacja na	*	+	<>	=	AND	XOR	OR
argumencie	/	-	<=>=	<>			
	REM						

W przykładzie poniżej, instrukcje są wykonywane w następującym porządku:
SQRT (%MW3) + %MW5 * 7 AND %MW8 OR %MW5 XOR %MW10



Nawiasy
Dzięki zastosowaniu nawiasów można zmieniać kolejność wykonywania operacji. Zaleca się ich stosowanie w celu nadania właściwej struktury wyrażeniom numerycznym.

((%MW5 AND %MW6) + %MW7) * %MW8



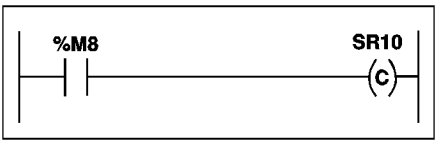
1.5 Instrukcje programowe

1.5-1 Wywołanie procedury *Subroutine call*

Instrukcja wywołania procedury umożliwia wywołanie modułu podprogramu w obrębie tego samego zadania.

Struktura

Język *Ladder*



Język *List*

```
LD    %M8
SR10
```

Język *ST*

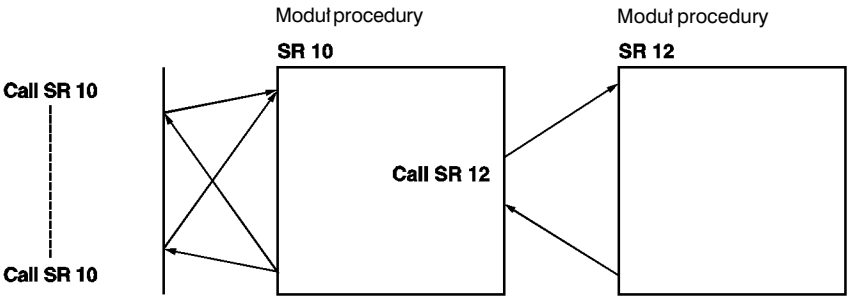
```
IF %M8 THEN
    SR10 ;
END_IF ;
```

Symbol *SRi* reprezentuje wywoływaną procedurę: *i* jest liczbą z przedziału 0 ÷ 253).

Reguły

- Wywołanie procedury może dotyczyć procedury już istniejącej.
- W akcji następującej po instrukcji wywołania procedury, wykonywany jest powrót do programu (*subroutine return*).
- W obrębie danej procedury można umieszczać wywołania innych procedur. Liczba takich wywołań ograniczona jest do 8 poziomów.
- Przocedury są przypisane do danego zadania. Stąd też mogą one być wywoływane tylko w obrębie tego zadania.

Zasada wywoływania procedur

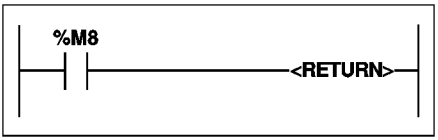


1.5-2 Powrót do programu *Subroutine return*

Instrukcja powrotu do programu jest zarezerwowana dla modułu procedury i umożliwia powrót do modułu wywołującego (gdy poprzedzający ją warunek logiczny jest spełniony).

Struktura

Język *Ladder*



Język *List*

```
LD      %M8
RETC
```

Język *ST*

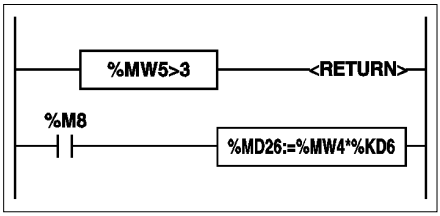
```
IF %M8 THEN
    RETURN ;
END_IF ;
```

Reguły

Instrukcja powrotu do programu wywołującego jest ukryta na końcu każdej procedury. Jej wcześniejsze użycie umożliwia powrót do programu przed końcem procedury.

Przykład:

Język *Ladder*



Język *List*

```
LD      [%MW5>3]
RETC
LD      %M8
[%MD26:=%MW4*%KD6]
```

Język *ST*

```
IF (%M5 > 3) THEN
    RETURN ;
END_IF ;
IF %M8 THEN
    %MD26 := %MW4 * %KD6 ;
END_IF ;
```

W języku *List* można stosować dodatkowo instrukcje:

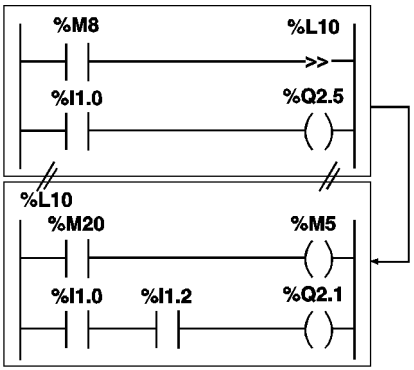
RETCN : powrót do programu, gdy poprzedzający warunek logiczny ma wartość 0.
RET : bezwarunkowy powrót do programu.

1.5-3 Instrukcje skoku *Jump*

Instrukcja skoku umożliwia przejście do linii programu oznaczonej etykietą %Li:
JMP : skok bezwarunkowy,
JMPC : skok, gdy wynikiem poprzedzajàcego warunku logicznego jest 1,
JMPCN : skok, gdy wynikiem poprzedzajàcego warunku logicznego jest 0,
Symbol %Li reprezentuje etykietę linii, do której ma nastąpić skok (*i* jest liczbą z przedziału od 1 do 999, maksymalnie 256 etykiet).

Struktura

Język *Ladder*



Język *List*

```
LD      %M8
JMPC    %L10
LD      %I1.0
ST      %Q2.5
.............
%L10 :
LD      %M20
ST      %M5
LD      %I1.0
AND     %I1.2
ST      %Q2.1
```

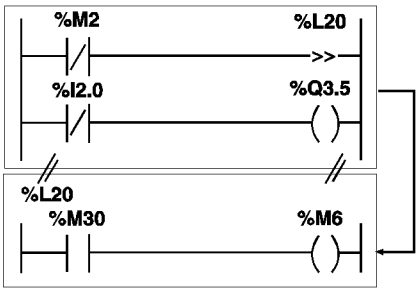
Skok do linii %L10 gdy %M8 ma wartość 1.

Język *ST*

```
IF %M8 THEN
    JUMP %L10 ;
END_IF ;
%Q2.5 := %I1.0 ;
%L10 :
    %M5 := %M20 ;
    %Q2.1 := %I1.0 AND %I1.2 ;
```

Skok do linii %L10, gdy %M8 ma wartość 1.

Język *Ladder*



Język *List*

```
LD      %M2
JMPCN   %L20
LDN     %I2.0
ST      %Q3.5
.....
%L20 :
LD      %M30
ST      %M6
```

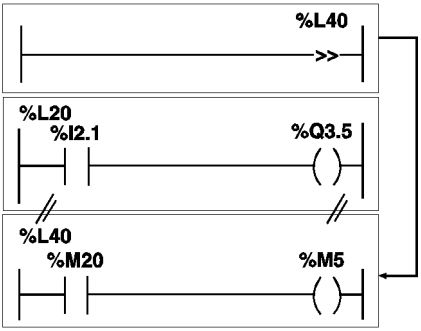
Skok do linii %L20 gdy %M2 ma wartość 0.

Język *ST*

```
IF  NOT %M2  THEN
    JUMP %L20 ;
END_IF ;
%Q3.5 := NOT %I2.0 ;

%L20 :
    %M6 := %M30 ;
```

Skok do linii %L20,
gdy %M2 ma wartość 0.

Język *Ladder*



Język *List*

```
JMP     %L40
%L20 :
LD      %I2.1
ST      %Q3.5
.....
%L40 :
LD      %M20
ST      %M5
```

Bezwarunkowy skok do linii %L40.

Język *ST*

```
JUMP %L40 ;

%L20 :
    %Q3.5 := %I2.1 ;

%L40 :
    %M5 := %M20 ;
```

Bezwarunkowy
skok do linii %L40,

Reguły

- Skok odbywa się w obrębie danego modułu programowego (program główny zadania głównego *MAIN*, procedura *%SRi*, itp).
- Skok może być wykonywany zarówno do linii wcześniejszej jak i do linii późniejszej (względem linii bieżącej).
  W przypadku skoku do linii wcześniejszej należy zwrócić uwagę na czas trwania przejścia programu (*scan time*): czas przejścia programu ulega wydłużeniu, co może oznaczać przekroczenie czasu wykonania zadania.

1.5-4 Instrukcje końca programu

Moment zakończenia wykonywania programu definiuje się przy pomocy instrukcji END, ENDC oraz ENDCN:

END : bezwarunkowy koniec programu
ENDC : koniec programu, gdy poprzedzający warunek logiczny ma wartość 1.
ENDCN : koniec programu, gdy poprzedzający warunek logiczny ma wartość 0.

Domyślnie (praca w trybie normalnym), zakończenie programu powoduje uaktualnienie stanów wyjść i ropoczęcie nowego "przejścia" programu.
Jeżeli program wykonywany jest periodycznie, to po uaktualnieniu stanów wyjść program czeka na zakończenie cyklu i dopiero po nim następuje nowe "przejście".

Uwaga:
W/w instrukcje mogą być stosowane tylko w języku *List* do programowania zadania głównego.

Przykład:

Język *List*

```
LD      %M1
ST      %Q2.1
LD      %M2
ST      %Q2.2
.....
END
```

```
LD      %M1
ST      %Q2.1
LD      %M2
ST      %Q2.2
.....
LD      %I1.2
ENDC    ⟶ Gdy %I1.2 =1, następuje koniec programu.
LD      %M2
ST      %Q2.2
.....
END
```

Gdy %I1.2 =0 program jest wykonywany aż do następnej instrukcji END.

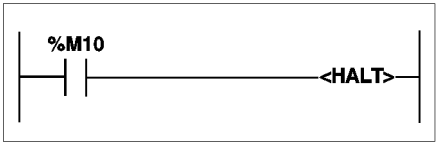
1.5-5 Zatrzymanie programu

Wykonywany program może zostać zatrzymany za pomocą instrukcji HALT (zatrzymanie wszystkich zadań). Powoduje ona zamrożenie zmiennych w programie.

Zatrzymany w ten sposób program uruchamia się poprzez jego ponowną inicjację (za pomocą komendy INIT). Żadna z instrukcji następujących po instrukcji HALT nie jest wykonywana.

Struktura

Język *Ladder*



Język *List*

```
LD    %M10
HALT
```

Język *ST*

```
IF %M10 THEN
    HALT ;
END_IF ;
```

1.5-6 Instrukcja maskowania zdarzeń

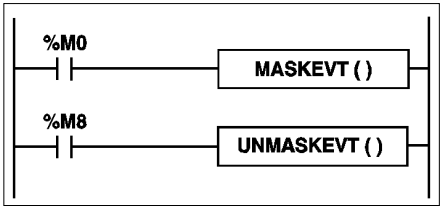
Instrukcja maskowania zdarzeń służy do ukrywania i odkrywania zdarzeń wyzwalających zadania.

MASKEVT : powoduje nałożenie maski na wszystkie zdarzenia. Zdarzenia są zapamiętywane przez sterownik lecz sprzężone z nimi zadania nie są wykonywane dopóty, dopóki nie zostanie zdjęta maska (instrukcja UNMASKEVT).

UNMASKEVT : powoduje odkrycie wszystkich zdarzeń. Następuje przetworzenie wszystkich zdarzeń, które zostały zignorowane za względu na zamaskowanie. Przetwarzanie zdarzeń trwa do momentu pojawienia się nowej instrukcji MASKEVT.

Struktura

Język *Ladder*



Język *List*

```
LD          %M0
[MASKEVT()]

LD          %M8
[UNMASKEVT()]
```

Język *ST*

```
IF %M0 THEN
    MASKEVT ( ) ;
END_IF ;
IF %M8 THEN
    UNMASKEVT ( ) ;
END_IF ;
```

1.5-7 Instrukcja NOP

Instrukcja **NOP** nie powoduje wykonania jakiejkolwiek operacji. Umożliwia ona "rezerwowanie" linii w programie w celu późniejszego uzupełnienia ich instrukcjami bez konieczności modyfikowania numeracji linii.

2.1 Wprowadzenie

2.1-1 Wiadomości ogólne

Instrukcje opisane w niniejszym rozdziale są wykorzystywane przy programowaniu bardziej skomplikowanych aplikacji.

Bez względu na język w jakim są one używane, powodują ten sam efekt. Różnią się tylko składnią. Są to:

- albo podstawowe instrukcje oprogramowania,
- albo funkcje traktowane jako rozszerzenie oprogramowania.

Instrukcje pełniące rolę funkcji dodatkowych służą do zwiększania możliwości oprogramowania poprzez zastosowanie specjalnych instrukcji.

- operacje na łańcuchach znaków, tablicach słów, itp.
- funkcje specjalne: Komunikacja, Sterowanie procesem, Interfejs Czlowiek - Maszyna (funkcja *MMI*), itp.

Instrukcje złożone obejmują następujące grupy:

- łańcuchy znaków,
- tablice słów,
- zarządzanie danymi i czasem trwania,
- konwersje,
- tablice bitowe,
- funkcje "*Orphee*"

- Komunikacja
- Sterowanie procesem
- Interfejs Człowiek - Maszyna
- Sterowanie ruchem

==> patrz: funkcje specjalne

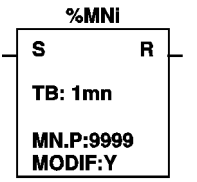
Uwagi dotyczące programowania

Instrukcje pełniące rolę funkcji zajmują dodatkowy obszar pamięci aplikacji (tylko pod warunkiem, że są zastosowane w programie).
Podczas programowania należy uwzględnić fakt, że każda funkcja (niezależnie od liczby wystąpień w programie) zajmuje dodatkowy obszar (należy obserwować wielkość zajętej pamięci sterownika).

2.2 Bloki funkcyjne złożone (standardowe)

2.2-1 Blok przerzutnika monostabilnego %MNi

Blok przerzutnika monostabilnego służy do generowania impulsu o ściśle określonym czasie trwania.
Czas trwania impulsu jest programowany i może być modyfikowany za pomocą terminala.



Blok przerzutnika monostabilnego

Charakterystka

Numer	%MNi	0 do 7 dla TSX 37, 0 do 254 dla TSX 57
Podstawa czasu	TB	1min, 1s, 100ms, 10ms (domyślnie - 1min).
Wartość bieżąca	%MNi.V	Słowo, którego wartość maleje od %MNi.P do 0 podczas pracy zegara. Może być ona czytana i testowana lecz nie może być zapisywana.
Wartość nastawiona	%MNi.P	0 < %MNi.P ≤ 9999. Słowo - można je czytać, testować i zapisywać. Czas trwania impulsu (PRESET) wynosi: %MNi.P x TB.
Edycja - terminal MODIF	Y/N	Y: możliwość zmiany wartości nastawionej podczas regulacji (tryb *adjust*). N: brak dostępu w trybie regulacji.
Wejście *Start* (lub instrukcja)	S(Start)	Na zboczu rosnącym %MNi.V = %MNi.P, potem %MNi.V jest zmniejszana do 0.
Wyjście	R(Praca)	Sprzężony z nim bit %MNi.R ma wartość 1, gdy %MNi.V > 0 (blok wykonuje operacje). %MNi.R = 0 jeśli %MNi.V = 0.

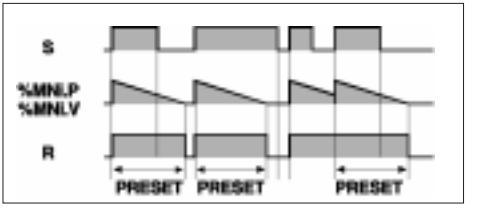
Sposób działania

Gdy na wejściu S bloku pojawia się wartość 1 (zbocze rosnące), wartość bieżąca %MNi.V przyjmuje wartość zapisaną w %MNi.P (wartość nastawiona), a blok zaczyna odliczanie do 0, zmniejszając wartość bieżącą o jednostkę przy każdym impulsie podstawy czasu TB. Bit wyjściowy %MNi.R (*Running*) przypisany do wyjścia R zmienia stan na 1 i utrzymuje tę wartość dopóki wartość bieżąca %MNi.V jest różna od 0. W momencie, gdy %MNi.V = 0, bit wyjściowy %MNi.R powraca do stanu 0.

Wejście S:

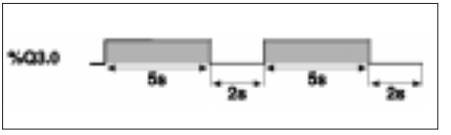
Wartość bieżąca %MNi.V:

Wyjście R:



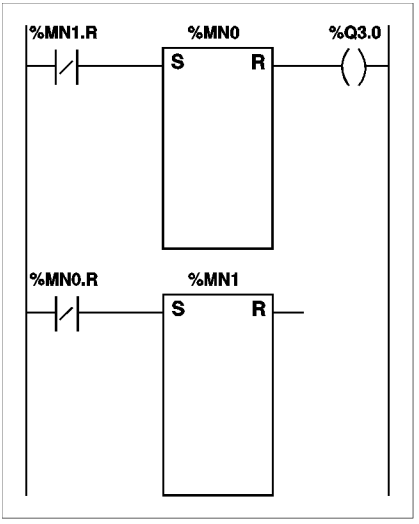
Programowanie i konfigurowanie

- Przykładowe zastosowanie: impulsowanie ze zmiennym cyklem: wartość nastawiona każdego bloku monostabilnego określa czas trwania poszczególnych impulsów.



- Konfigurowanie
  Przy pomocy edytora zmiennych należy zdefiniować następujące parametry:
  - TB : 1min, 1s, 100ms, 10ms lub 1ms (dla tego przykładu - 100ms).
  - %MNi.P : 0 do 9999 (dla tego przykładu %MN0.P=50 i %MN1.P=20).
  - MODIF : Y lub N.

- Programowanie
  Język *Ladder*



Język *List*

```
LDN     %MN1.R
ANDN    %Q3.0
S       %MN0
LD      %MN0.R
ST      %Q3.0
LDN     %MN0.R
S       %MN1
```

Język *ST*

```
%M0:=NOT %MN1.R ;
IF RE %M0 THEN
    START %MN0 ;
END_IF ;
%Q3.0 := %MN0.R ;
%M1 := NOT %MN0.R ;
IF RE %M1 THEN
    START %MN1 ;
END_IF ;
```

W tym przykładzie, wyjście %Q3.0 przyjmuje wartość 1 na czas 5s (%MN0.P) i kasuje się do 0 na 2s (%MN1.P).

W języku *ST*, instrukcja START %Mni służy do wystartowania bloku monostabilnego. Wymusza ona 1 na wejściu S bloku i w ten sposób powoduje jego inicjację. Stąd też uaktywnienie tej instrukcji powinno odbywać się podczas trwania impulsu.

Uwaga

Funkcja przerzutnika monostabilnego może być również realizowana przy pomocy bloku funkcyjnego %TMi pracującego w trybie TP (część B, rozdz.1.3-2).

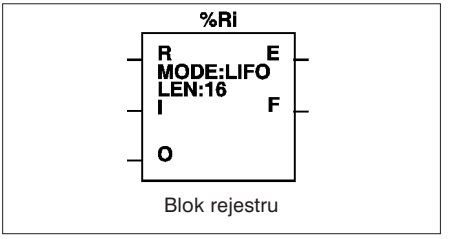
Przypadki szczególne

- Reakcja na zimny start: (%S0 = 1) wartość nastawiona %MNi.P jest ładowana do wartości bieżącej %MNi.V. Wyjście %MNi.R jest kasowane do 0, ponieważ wartość nastawiona mogła być zmodyfikowana, a w reakcji na zimny start nowa wartość zostaje utracona.
- Reakcja na gorący start: (%S1=1) nie ma żadnego wpływu na bieżącą wartość bloku monostabilnego (%MNi.V).
- Reakcja na zatrzymanie sterownika, dezaktywację zadania lub przerwanie: żadna z tych sytuacji nie powoduje zamrożenia wartości bieżącej.
- Reakcja na skok programu: fakt, że część programu zawierającego blok monostabilny nie jest skanowana (program nie "czyta" tej części programu) nie powoduje zamrożenia wartości bieżącej %MNi.V, która nadal maleje do 0.
  Podobnie, bit %MNi.R przypisany do wyjścia bloku przyjmuje wartości wynikające z normalnej pracy bloku i dlatego może on być testowany w innej części programu. Jednakże, sprzężenie (cewka) bezpośrednio "podpięte" do wyjścia bloku (np. %Q3.0) nie zostanie uaktywnione ponieważ sterownik go nie odczytuje.
- Testowanie bitu %MNi.R: w czasie jednego "przejścia" (*scan*) stan tego bitu może ulegać zmianie.

2.2-2 Rejestr %Ri

Rejestr jest blokiem pamięciowym, w którym można zapamiętywać, na dwa różne sposoby, do 255 słów 16-bitowych:

- w postaci kolejki (*queue*), zwanej stosem typu FIFO (pierwszy na wejściu, pierwszy na wyjściu *First In, First Out*).
- w postaci stosu (*stack*), zwanego stosem typu LIFO (ostatni na wejściu, pierwszy na wyjściu *Last In, First Out*).



Blok rejestru

Charakterystyka

Numer rejestru	%Ri	0 do 3 dla TSX 37, 0 do 254 dla TSX 57
Rodzaj	FIFO LIFO	Kolejka. Stos (ustawienie domyślne).
Długość w rejestrze	LEN	Liczba słów 16-bitowych (1- LEN- 255)
Słowo wejściowe	%Ri.I	Słowo wejściowe rejestru. Można je czytać, testować i zapisywać.
Słowo wyjściowe	%Ri.O	Słowo wyjściowe rejestru. Można je czytać, testować i zapisywać.
Wejście *Zapis* (lub instrukcja)	I (In)	Przy zboczu rosnącym powoduje zapisanie treści słowa %Ri.I w rejestrze.
Wejście *Odczytaj* (lub instrukcja)	O (Out)	Przy zboczu rosnącym powoduje zapisanie treści słowa z rejestru do słowa %Ri.O.
Wejście *Kasuj* (lub instrukcja)	R (Reset)	Wartość 1 powoduje inicjację rejestru.
Wyjście *Pusty*	E (Empty)	Bit wyjściowy %Ri.E sygnalizuje, że rejestr jest pusty. Można go testować.
Wyjście *Pełny*	F (Full)	Bit wyjściowy %Ri.F sygnalizuje, że rejestr jest pełny. Można go testować.

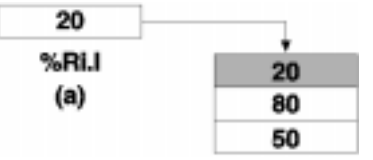
Uwaga:
Jednoczesne uaktywnienie wejść I oraz O powoduje wykonanie wpierw operacji zapisania, a dopiero potem odczytania zawartości rejestru.

FIFO (pierwszy na wejściu, pierwszy na wyjściu)

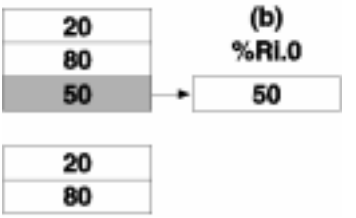
Dane wprowadzone jako pierwsze będą jako pierwsze odczytane. W odpowiedzi na żądanie zapisania (zbocze rosnące na wejściu I lub użycie instrukcji I), zawartość słowa wejściowego %Ri.I (która została wcześniej załadowana) zostaje zapisana na szczycie stosu (rys. a).
Jeżeli stos jest zapełniony (wyjście F=1), to nie ma możliwości zapisywania danych i bit systemowy %S18 ma stan 1.
W odpowiedzi na żądanie odczytania (zbocze rosnące na wejściu O lub użycie instrukcji O), dane słowa znajdującego się najniżej w stosie zostają zapisane w słowie %Ri.O a zawartość rejestru zmniejsza się o jedną pozycję (rys. b).
Jeżeli rejestr jest pusty (wyjście E=1), to nie ma możliwości dokonywania odczytu. Słowo wyjściowe %Ri.O się nie zmienia i zachowuje swoją zawartość. Zawartość stosu można w dowolnym momencie kasować (1 na wejściu R lub instrukcja R).

Przykład:

Zapisanie treści słowa %Ri.I na szczycie stosu.



Odczytanie pierwszej danej i załadowanie jej do %Ri.O.

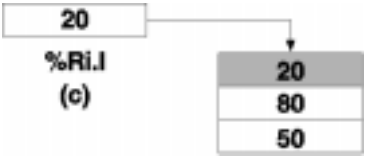


LIFO (ostatni na wejściu, pierwszy na wyjściu)

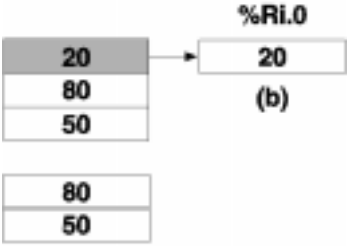
Dane zapisane jako ostatnie zostaną odczytane jako pierwsze. W odpowiedzi na żądanie zapisania (zbocze rosnące na wejściu I lub użycie instrukcji I), zawartość słowa wej. %Ri.I (która została wcześniej załadowana) zostaje zapisana na szczycie (rys. c). Jeżeli stos jest zapełniony (wyjście F=1), to nie ma możliwości zapisywania danych i bit systemowy %S18 ma stan 1.
W reakcji na żądanie odczytania (zbocze rosnące na wejściu O lub instrukcja O), dane słowa znajdującego się najwyżej w stosie zostają zapisane w słowie %Ri.O (rys. d).
Jeżeli rejestr jest pusty (wyjście E=1), to nie ma możliwości dokonywania odczytu. Słowo wyjściowe %Ri.O się nie zmienia i zachowuje swoją zawartość. Zawartość stosu można w dowolnym momencie kasować (1 na wejściu R lub instrukcja R). Zaznaczony element zostaje najwyższym elementem stosu.

Przykład:

Zapisanie zawartości słowa %Ri.I na szczycie stosu.



Odczytanie danej ze szczytu stosu.



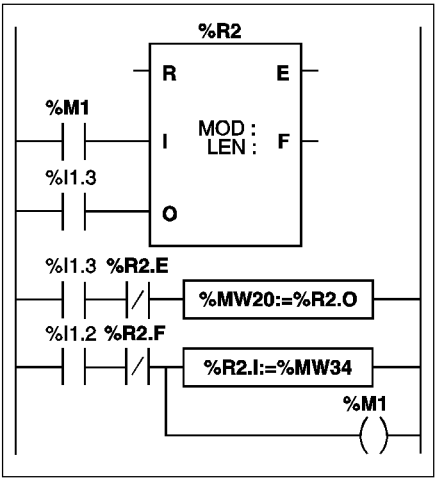
Programowanie i konfigurowanie

- Konfigurowanie
  Przy pomocy edytora konfiguracyjnego należy zdefiniować następujące parametry:
  - Numer : 1 do 4 dla TSX 37, 1 do 255 dla TSX 57,
  - Długość : 1 do 255.

  W edytorze zmiennych należy określić tryb pracy rejestru (FIFO czy LIFO).

- Programowanie

Język *Ladder*



Język *List*

```
LD      %M1
I       %R2
LD      %I1.3
O       %R2
LD      %I1.3
ANDN    %R2.E
[%MW20:=%R2.O]
LD      %I1.2
ANDN    %R2.F
[%R2.I:=%MW34]
ST      %M1
```

Język *ST*

```
IF RE %M1 THEN
    PUT %R2 ;
END_IF ;
IF RE %I1.3 THEN
    GET %R2 ;
END_IF ;
IF (%I1.3 AND NOT %R2.E) THEN
    %MW20 := %R2.O ;
END_IF ;
%M1 := %I1.2 AND NOT %R2.F ;
IF %M1 THEN
    %R2.I := %MW34 ;
END_IF ;
```

Zamieszczony powyżej przykład pokazuje ładowanie słowa %MW34 do rejestru %R2.I, w reakcji na żądanie zapisania %I1.2, pod warunkiem, że rejestr R2 nie jest przepełniony (%R2.F=0). Żądanie zapisania w rejestrze realizowane jest za pomocą %M1. Żądanie odczytanie realizowane jest za pośrednictwem wejścia %I1.3, %R2.O jest ładowane do %MW20 pod warunkiem, że rejestr nie jest pusty (%R2.E=0).

W języku strukturalnym *ST* do programowania bloków rejestru służą 3 instrukcje:

- RESET %Ri : powoduje inicjację rejestru,
- PUT %Ri : powoduje zapisanie treści słowa %Ri.I w rejestrze,
- GET %Ri : powoduje zapisanie danych z rejestru w słowie %Ri.O.

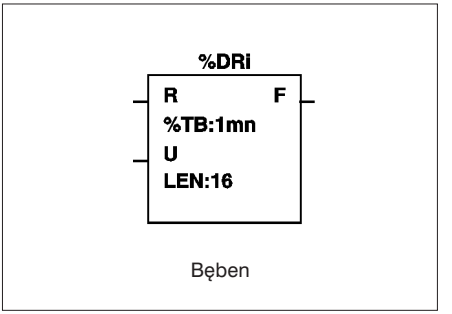
Instrukcje PUT i GET wywołują zbocze rosnące na odpowiadających im wejściach I oraz O. Użycie tych instrukcji powinno się więc odbywać podczas trwania impulsu.

Sytuacje szczególne

- Reakcja na zimny start: (%S0=1) powoduje skasowanie bieżącej zawartości rejestru. Bit wyjściowy %Ri.E przypisany do wyjścia E, przyjmuje wartość 1.
- Reakcja na gorący start: (%S1=1) nie ma wpływu ani na zawartość rejestru, ani na stan jego bitów wyjściowych.
- Skasowanie do 0 (wejście R lub instrukcja R)
  - W języku *Ladder*, zapamiętane wartości wejść I oraz O są aktualizowane z wartościami bieżącymi.
  - W języku *List*, zapamiętane wartości wejść I oraz O nie są aktualizowane. Każde z nich zachowuje wartości jakie miały przed wywołaniem.
  - W *ST*, zapamiętane wartości wejść I oraz O są uaktualniane z wartością 0.

2.2-3 Bęben *Drum controller* %DRi

Blok funkcyjny "Bęben" działa na podobnej zasadzie jak walec elektromechaniczny, który zmienia kroki w reakcji na wystąpienie określonych warunków (zdarzeń). Przy każdym obrocie krzywka i styki wskazują systemowi kolejność wykonywania operacji. W przypadku bloku "Bębna" krzywkę i styki symbolizuje wartość 1 dla danego kroku (obrotu walca) przyporządkowywana bitom wyjściowym %Qi.j lub bitom wewnętrznym%Mi, zwanych bitami sterującymi.



Bęben

Charakterystyka

Numer	%DRi	0 do 7 dla TSX 37, 0 do 254 dla TSX 57
Numer kroku	LEN	1 do 16 (domyślnie 16).
Podstawa czasu	TB	1min, 1s, 100ms, 10ms (domyślnie 1min).
Taktowanie lub czas trwania okresu bieżącego krokui	%DRi.V	0≤%DRi.V≤9999. Słowo kasowane przy każdej zmianie kroku. Może być czytane i testowane ale nie może być zapisywane. Czas trwania jest równy %DRi.V x TB.
Numer bieżącego kroku	%DRi.S	0≤%DRi.S≤15. Słowo to może być czytane i testowane. Może być zapisane tylko za pomocą wartości bezpośredniej.
Wejście "Powrót do kroku 0"	R (RESET)	Jeżeli ma wartość 1, to powoduje powrót do kroku 0 (inicjacja bloku).
Wejście "Przesuń krok"	U (UP)	Przy zboczu rosnącym, powoduje obrót bębna o 1 krok i uaktualnia bity sterujące.
Wyjście	F (FULL)	Sygnalizuje, że bieżący krok jest ostatnim zdefiniowanym krokiem. Przyporządkowany mu bit %DRi.F można testować (%DRi.F=1 gdy %DRi.S=zdefiniowana liczba kroków - 1).
Status kroku	%DRi.Wj	16-bitowe słowo odwzorowujące stan kroku *j* bębna *i*. Może być czytane i testowane ale nie może być zapisywane.
Bity sterujące		Bity wyjściowe lub wewnętrzne przypisane do kroku (16 bitów sterujących).

Uwaga: Bit %S18 zmienia stan na 1, w razie zapisania nie skonfigurowanego kroku.

Sposób działania

Bęben składa się z:

- matrycy stałych danych (krzywek) uporządkowanej:
  - w kolumny: ponumerowane od 0 do N-1 (*N* jest zdefiniowaną liczbą kroków). Każda kolumna odwzorowuje stan kroku za pomocą 16 bitów danych ponumerowanych od 0 do F.
- listy bitów sterujących (po 1 na linię) odpowiadających wyjściom %Qxy.i, lub bitom wewnętrznym %Mi. Dla bieżącego kroku, bity sterujące przyjmują wartości dla niego zdefiniowane.

Przedstawiona poniżej tabela zawiera główną charakterystykę bębna, dla którego zdefiniowano 16 kroków.

Krok

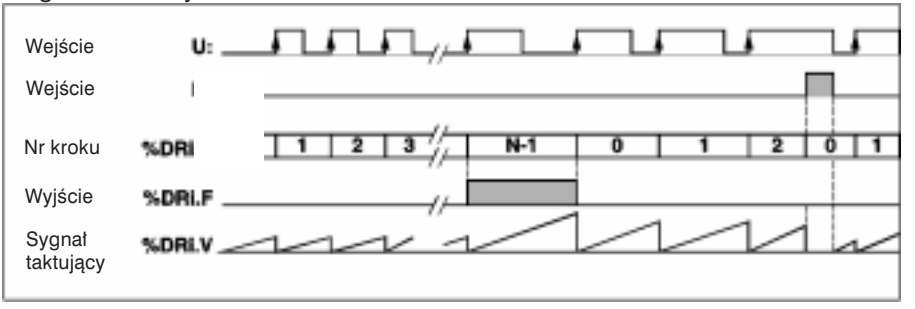
Bit

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Address
0	0	0	0	1	0	1	0	1	0	1	1	0	1	1	0	0	%Q2.0
1	0	1	0	0	1	1	1	0	0	0	1	0	0	0	1	0	%Q2.1
2	0	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0	%M23
3	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0	%M32
4	1	0	0	0	1	1	1	0	0	0	1	1	0	1	0	1	%Q2.9
5	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	%Q2.10
6	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	%Q2.11
7	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	%Q2.12
8	1	0	0	0	1	0	0	0	0	1	0	1	0	1	0	0	%M8
9	0	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	%M9
A	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	%M1
B	1	0	0	1	1	0	0	0	1	0	0	0	0	1	0	1	%Q2.6
C	1	0	1	0	1	0	1	0	0	1	1	1	0	0	0	0	%Q2.7
D	0	1	0	0	1	0	1	1	1	0	0	1	0	0	1	1	%Q2.8
E	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	%M100
F	0	0	1	0	0	0	0	1	0	1	0	0	1	1	1	0	%M13

Bity sterujące

W przykładzie powyżej, dla kroku 1, bity sterujące %Q2.1;%Q3.5; %Q2.8;%Q3.6;%M5 i %M6 przyjmują wartość 1. Pozostałe bity sterujące przyjmują wartość 0.
Numer kroku bieżącego jest zwiększany przy każdym wystąpieniu zbocza rosnącego na wejściu U (lub przy wykonaniu instrukcji U). Numer ten może być modyfikowany przez program.

Diagram czasowy



Programowanie i konfigurowanie

W niniejszym przykładzie, pierwszych 5 wyjść %Q2.0 do %Q2.4 uaktywnia się kolejno, za każdym razem, gdy na wejściu %I1.1 pojawia się 1.
Wejście I1.0 kasuje wyjścia i cofa bęben do kroku 0.

Konfigurowanie

Przy pomocy edytora zmiennych definiuje się następujące parametry:

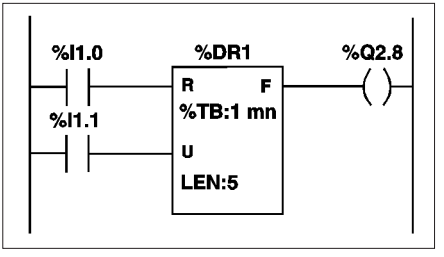
- Liczbę kroków: 5 (LEN:5).
- Stan wyjść (bity sterujące) dla każdego kroku bębna.

	Krok	Przyporządkowanie bitów sterujących
	0 1 2 3 4	
	0 : 1 0 0 0 0	%Q2.0
	1 : 0 1 0 0 0	%Q2.1
Bit	2 : 0 0 1 0 0	%Q2.2
	3 : 0 0 0 1 0	%Q2.3
	4 : 0 0 0 0 1	%Q2.4

- Podstawę czasu (TB:1 min).

Programowanie

Język *Ladder*



Język *List*

```
LD      %I1.0
R       %DR1
LD      %I1.1
U       %DR1
LD      %DR1.F
ST      %Q2.8
```

Język *ST*

```
IF %I1.0 THEN
    RESET %DR1 ;
END_IF ;
IF RE %I1.1 THEN
    UP %DR1 ;
END_IF ;
%Q2.8 := %DR1.F ;
```

W języku *ST* do programowania bloków bębna służą 2 instrukcje:

- RESET %DRi : powrót do kroku 0 bębna,
- UP %DRi : obrót bębna o 1 krok i uaktualnienie wyjść. Ta instrukcja powoduje wygenerowanie zbocza rosnącego na wejściu U bloku, tak więc uaktywnienie tej funkcji powinno odbywać się podczas impulsu.

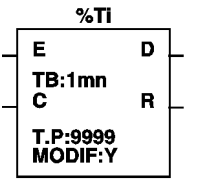
Uwaga

Skasowanie do 0 (wejście R, instrukcja R lub RESET)
- W języku *Ladder* zapamiętana wartość dla wejścia U jest zastępowana wartością bieżącą.
- W języku *List* zapamiętana wartość wejścia U nie jest uaktualniana. Zachowywana jest wartość sprzed operacji kasowania.
- W języku *ST* zapamiętana wartość wejścia U jest zastępowana wartością 0.

Systuacje szczególne
- Reakcja na zimny start: (%S0=1) powoduje powrót bębna do kroku 0 (z uaktualnieniem bitów sterujących).
- Reakcja na gorący start: (%S1=1) powoduje uaktualnienie bitów sterujących, zgodnie z wartościami dla bieżącego kroku.
- Reakcja na skok, dezaktywację zadania lub przerwanie: fakt, że blok bębna nie jest odczytywany przez program nie oznacza skasowania bitów sterujących do 0.
- Uaktualnienie bitów sterujących: tylko w reakcji na zmianę kroku oraz zimny i gorący start.

2.2-4 Blok zegara %Ti (Seria 7)

Blok zegara, który jest kompatybilny z blokami dla sterowników Serii 7 PL7-2/3 służy do realizacji sterowania akcjami wykonywanymi z opóźnieniem.
Wartość tego opóźnienia jest programowalna i może być modyfikowana przy użyciu terminala.



Blok zegara

Charakterystyka

Numer	%Ti	0 do 63 dla TSX 37, 0 do 254 dla TSX 57
Podstawa czasu	TB	1min, 1s, 100ms, 10ms (domyślnie 1min).
Wartość bieżąca	%Ti.V	Słowo, którego wartość, podczas pracy zegara, maleje od %Ti.P do 0. Może być testowane i czytane ale nie może być zapisywane.
Wartość nastawiona	%Ti.P	0 < %Ti.P ≤ 9999. Słowo - może być czytane, testowane i zapisywane. Domyślnie ma wartość 9999. Czas trwania odliczania wynosi: %Ti.PTB.
Regulacja - terminal MODIF	Y/N	Y: wartość nastawiona może być modyfikowana w trybie regulacji (*adjust mode*). N: brak dostępu w trybie regulacji.
Wejście nastawiania	E(Enable)	Wartość 0, kasuje zegar (%Ti.V = %Ti.P).
Wejście sterujące	C(Control)	Wartość 0, zamraża wartość bieżącą %Ti.V.
Wyjście zegara "Koniec liczenia"	D(Done)	Bit %Ti.D = 1 oznacza, że zegar skończył odliczanie (%Ti.V = 0).
Wyjście "Zegar aktywny"	R(Running)	Bit %Ti.R = 1 gdy zegar liczy (%Ti.P > %Ti.V > 0) i gdy wejście C ma wartość 1.

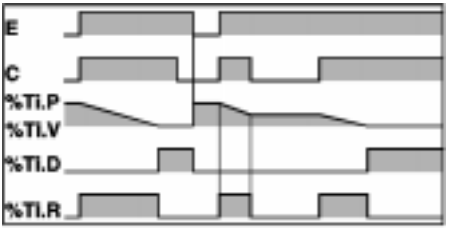
Uwaga:
Blok funkcyjny typu zegar %Ti nie może być programowany w języku *List*. Jednakże obiekty związane z blokami zegara %Ti (%Ti.V, %Ti.P, %Ti.D i %Ti.R) są dostępne.

Całkowita liczba zegarów %TMi + %Ti powinna być mniejsza niż 64 dla sterowników TSX 37 i mniejsza od 255 dla TSX 57.

Sposób działania

Zegar zmienia swoją wartość, gdy na dwóch wejściach (E i C) pojawia się 1. Zachowuje się on jak licznik odliczający.

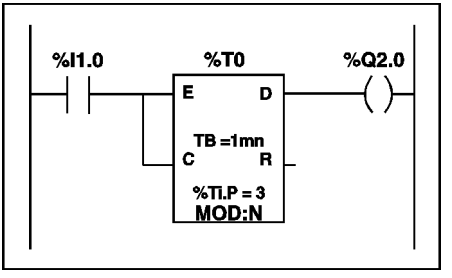
- Wartość bieżąca %Ti.V maleje od wartości nastawionej %Ti.P do 0, o jednostkę przy każdym impulsie podstawy czasu TB.
- Bit wyjściowy %Ti.R (Praca zegara) przypisany do wyjścia R ma wtedy wartość 1, a bit wyjściowy %Ti.D (Koniec pracy zegara) przypisany do wyjścia D ma w tym czasie wartość 0.
- Gdy wartość bieżąca %Ti.V=0, %Ti.D zmienia stan na 1, a %Ti.R wraca do 0.

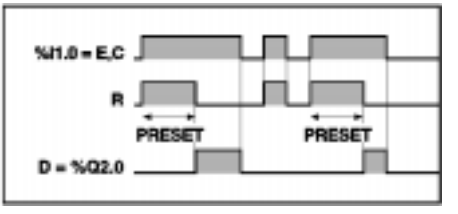


E	0	0	1	1
C	0	1	0	1
%Ti.P %Ti.V	%Ti.V = %Ti.P	%Ti.V = %Ti.P	%Ti.V zamroż.	%Ti.V maleje od %Ti.P -> 0
%Ti.D	0	0	0	1 gdy zegar wyliczył
%Ti.R	0	0	0	1 gdy zegar pracuje

Standardowe funkcje zegara
Blok zegara może spełniać następujące funkcje:

- Opóźnienie *On-time delay*
  Język *Ladder*

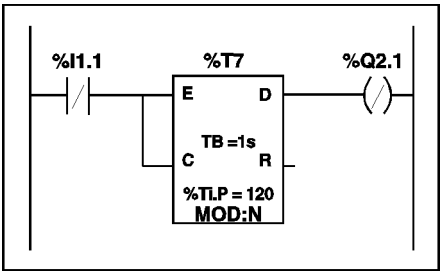


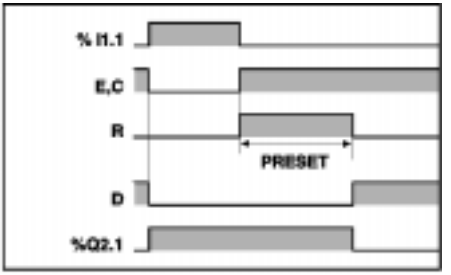


Język *ST*

```
IF %I1.0 THEN
    START %T0 ;
ELSE
    PRESET %T0 ;
END_IF ;
%Q2.0 := %T0.D ;
```

- **Opóźnienie *Off-time delay***
  Język *Ladder*

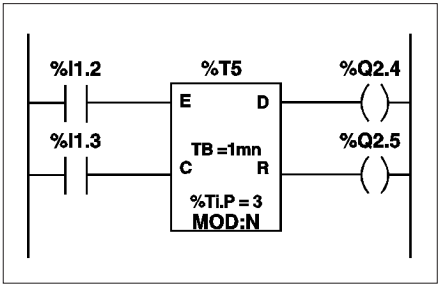


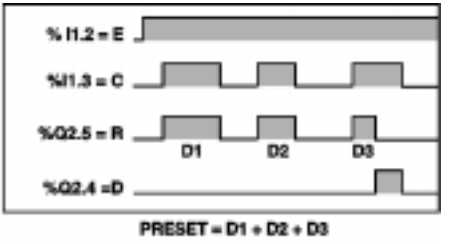


Język *ST*

```
IF %1.1 THEN
    PRESET %T7 ;
ELSE
    START %T7 ;
END_IF ;
%Q2.1 := NOT %T7.D ;
```

- Skumulowane opóźnienie *On-time delay*
  Język *Ladder*



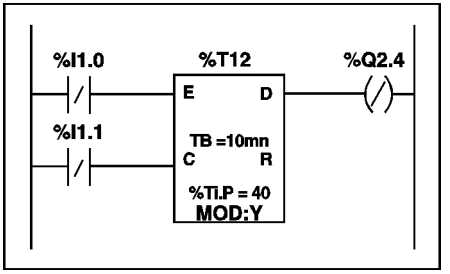


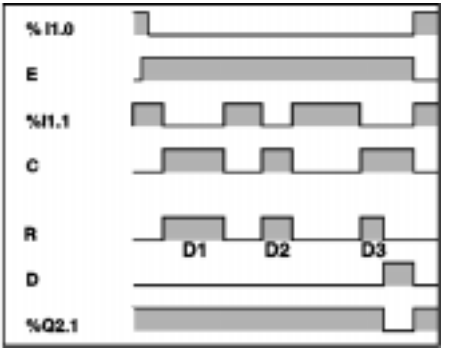
Język *ST*

```
IF %I1.2 THEN
    IF %I1.3 THEN
        START %T5 ;
    ELSE
        STOP %T5 ;
    END_IF ;
ELSE
        PRESET %T5 ;
END_IF ;
%Q2.4 := %T5.D ;
%Q2.5 := %T5.R ;
```

- **Skumulowane opóźnienie *Off-time delay***

Język *Ladder*





PRESET = D1 + D2 + D3

Język *ST*

```
IF %I1.0 THEN
        PRESET %T12 ;
ELSE
    IF %I1.1 THEN
        STOP %T12 ;
    ELSE
        START %T12 ;
    END_IF ;
END_IF ;
%Q2.4 := NOT %T12.D ;
```

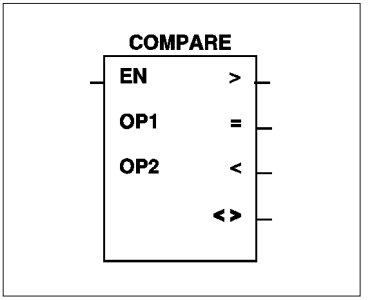
W języku *ST*, do programowania bloków zegara %Ti służą 3 instrukcje:
- PRESET %Ti : kasuje zegar,
- START %Ti : uruchamia zegar,
- STOP %Ti : zamraża wartość bieżącą zegara.

Sytuacje szczególne
- Reakcja na zimny start: (%S0 = 1) wartość nastawiona (zdefiniowana w edytorze zmiennych) jest ładowana do wartości bieżącej, a wyjście %Ti.D ma wartość 0, ponieważ ewentualnie zmodyfikowana wartość nastawiona jest ignorowana.
- Reakcja na gorący start: (%S1=1) nie ma wpływu na bieżącą wartość zegara.
- Reakcja na zatrzymanie sterownika: zatrzymanie sterownika, dezaktywacja bieżącego zadania lub przerwanie, nie powodują zamrożenia wartości bieżącej.
- Reakcja na skok programu: fakt, że labelka, w której zapisano blok zegara nie jest czytana przez program nie powoduje zamrożenia wartości bieżącej %Ti.V, która nadal maleje do 0.
  Podobnie, bity %Ti.D i %Ti.R przypisane do wyjść zegara D i R pracują normalnie tak, że mogą one być testowane w innych labelkach.
  Jednakże, sprzężenia (cewki) bezpośrednio podłączone do wyjść bloku nie będą uaktywniane, ponieważ labelki te nie są czytane przez program.
- Testowanie bitów %Ti.D i %Ti.R: stan tych bitów zmienia się w czasie "przejścia" programu.

2.3 Blok porównywania - złożony

Pionowe bloki porównywania służą do porównywania dwóch argumentów (Arg).

Tymi argumentami mogą być 16-bitowe słowa (mogą być indeksowane) lub wartości bezpośrednie.



Liczba pionowych bloków porównywania nie jest ani ograniczana, ani definiowana.

Charakterystyka

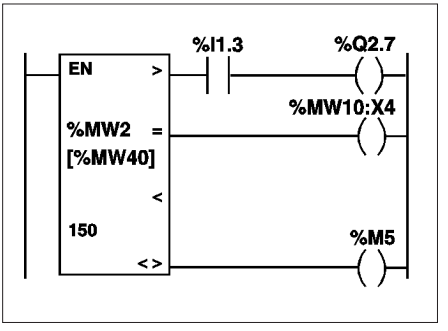
Wejście "Wykonaj"	EN	Wartość 1 - porównanie dwóch argumentów.
Wyjście "Większe niż"	>	Wartość 1, gdy Arg1 jest większy niż Arg2.
Wyjście "Równe"	=	Wartość 1, gdy Arg1 jest równy Arg2.
Wyjście "Mniejsze niż"	<	Wartość 1, gdy Arg1 jest mniejszy niż Arg2.
Wyjście "Różne"	<>	Wartość 1, gdy Arg1 jest różny od Arg2.
Argument nr 1	OP1	Ten argument jest słowem pojedynczej precyzji (może być słowem indeksowanym).
Argument nr 2	OP2	Ten argument jest słowem pojedynczej precyzji (może być słowem indeksowanym).

Sposób działania

Pojawienie się 1 na wejściu EN, powuje porównanie dwóch argumentów i uaktywnienie 4 wyjść bloku zgodnie z wynikiem operacji porównania. Pojawienie się 0 na tym wejściu powoduje skasowanie uaktywnionych wyjść.

- **Przykładowe zastosowanie**
  Zamieszczony poniżej program obrazuje operację porównywania słowa %MW2 indeksowanego za pomocą słowa %MW40 z wartością bezpośrednią 150.
  Jeżeli treść %MW2[%MW40] jest większa od 150, a %I1.3 = 1, to następuje uaktywnienie sprzężenia %Q2.7.
  Jeżeli treść tego słowa jest równa 150, to następuje uaktywnienie sprzężenia %MW10:X4. Sprzężenie %M5 jest uaktywniane wtedy, kiedy treść słowa jest różna od 150 (< lub >).

Język *Ladder*



Blok funkcyjny tego typu nie istnieje w językach *List* oraz *ST*.
Stosuje się w nich operacje porównywania >, <, =, <>

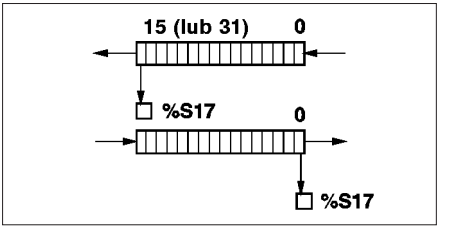
Sytuacje szczególne

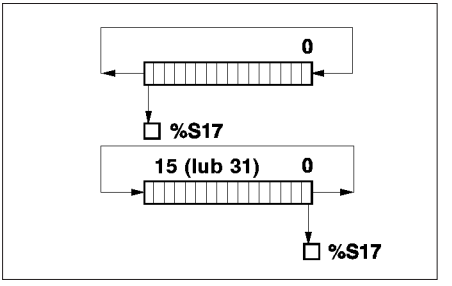
- Reakcja na zimny start: (%S0) argument OP1 i być może OP2 (o ile OP2 jest słowem wewnętrznym) są kasowane, a wyjścia są aktualizowane zgodnie z wynikiem porównywania nowych wartości.
- Reakcja na gorący start: (%S1) nie ma wpływu na blok porównywania.

2.4 Instrukcje przesuwania *Shift*

Instrukcje przesuwania umożliwiają przesuwanie bitów argumentów słowa lub słowa podwójnego o pewną liczbę pozycji (w prawo lub w lewo).

- Przesuwanie logiczne *Logic shift*:
  - SHL(arg2,i) logiczne przesunięcie w lewo o *i* pozycji,
  - SHR(arg2,i) logiczne przesunięcie w prawo o *i* pozycji,
- Przesuwanie okrężne:
  - ROL(arg2,i) okrężne przesunięcie w lewo o *i* pozycji,
  - ROR(arg2,i) okrężne przesunięcie w prawo o *i* pozycji,

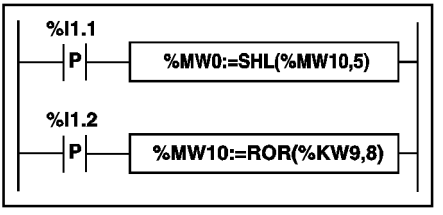




W przypadku przesuwania bitów w słowie pojedynczej precyzji wartość *i* będzie mieściła się w przedziale od 1 do 16.
W przypadku słów podwójnej precyzji zmienna *i* przyjmuje wartości z przedziału od 1 do 32.
Stan ostatniego bitu wyjściowego jest zapisywany w bicie %S17.

Struktura

Język *Ladder*



Język *List*

```
LDR       %I1.1
[%MW0 := SHL(%MW10,5)]
```

Język *ST*

```
IF RE %I1.2 THEN
    %MW10 := ROR (%KW9,8) ;
END_IF ;
```

Składnia

Operatory: SHL,SHR,ROL,ROR

```
Arg1:=Operator(Arg2,i)
```

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MW	%MW,%KW
Nieindeksowalne słowa	%QW,%SW,%NW, %BLK	Wart. bezp.,%IW,%QW,%SW %NW,%BLK, Wyr. num.
Indeksowalne słowa podw.	%MD	%MD,%KD
Nieindeks. słowa podwójne	%QD,%SD,	Wart.bezp.,%ID,%QD,%SD Wyrażenie numeryczne

2.5 Instrukcje na obiektach zmiennoprzecinkowych

2.5-1 Wiadomości ogólne

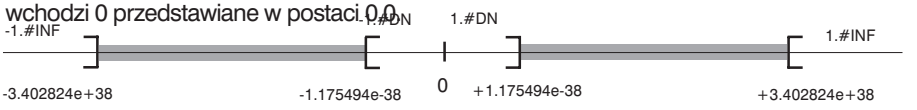
Język PL7 umożliwia wykonywanie operacji na obiektach zmiennoprzecinkowych.

Format obiektów zmiennoprzecinkowych jest zgodny z normą IEEE STD 734-1985 (równoważna normie IEC 559). Słowa są 32-bitowe, co odpowiada pojedynczej długości liczb zmiennoprzecinkowych.

Wartości zmiennoprzecinkowe mogą być reprezentowane z użyciem exponentu lub bez niego, lecz zawsze muszą zawierać przecinek (ruchomy przecinek).

Przykładowe wartości zmiennoprzecinkowe:
bez exponentu : 1285.28
z exponentem : 1.28528e3

Wartości mieszczą się w przedziale od -3.402824e+38 do -1.175494e-38 i od 1.175494e-38 do 3.402824e+38 (obszary zacienione na wykresie poniżej). W skład tego przedziału wchodzi 0 przedstawiane w postaci 0.0.



Jeżeli wynik obliczeń mieści się w przedziale od -1.175494e-38 do 1.175494e-38, to jest on zaokrąglany do 0. Jeżeli wartość z tego przedziału została zapisana w innym formacie, to nie można jej zapisać w formacie zmiennoprzecinkowym. Przy próbie zapisania takiej liczby pojawi się symbol 1.#DN lub - 1.#DN.

Jeżeli wynik obliczeń jest:
- mniejszy niż -3.402824e+38, to wyświetlany jest symbol -1.#INF (-),
- większy niż +3.402824e+38, to wyświetlany jest symbol 1.#INF (+

Jeżeli wynik operacji jest niedefiniowalny (np. pierwiastek kwadratowy z liczby ujemnej) wyświetlany jest symbol 1.#NAN lub -1.#NAN.

Bit systemowy %S18 przyjmuje wartość 1, gdy wynik nie mieści się w podanych przedziałach. O przyczynie błędu w operacji informują bity słowa statusu %SW17:
%SW17:X0 = operacja nieprawidłowa, wynik nie jest liczbą (1.#NAN lub -1.#NAN),
%SW17:X1 = niestandardowy argument (pomiędzy -1.175494e-38 a 1.175494e-38), wynik jest zaokrąglany do 0,
%SW17:X2 = dzielenie przez 0, wynik jest nieokreślony ± (-1.#INF lub 1.#INF),
%SW17:X3 = wynik większy niż +3.402824e+38 (wartość bezwzględna), wynikiem jest ± (-1.#INF lub 1.#INF),
%SW17:X4 = wynik jest mniejszy od 1.175494e-38, wynikiem jest 0,
%SW17:X5 = wynik jest nieprecyzyjny.
Słowo to jest kasowane do 0 przez system po zimnym starcie lub przez program, w celu przygotowania do powtórnego użycia.

Odwzorowanie jest dokładne do wartości 2^{-24}. Przy podglądaniu wartości zmiennoprzecinkowych wyświetlanych jest najwyżej 6 cyfr po przecinku.

Uwagi

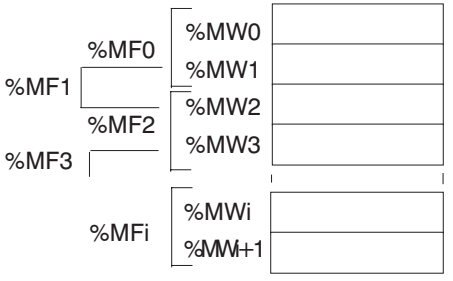
- Wartość "1285" jest traktowana jako liczba całkowita. Jeżeli ma być traktowana jako liczba zmiennoprzecinkowa, to należy ją zapisać: "1285.0",
- Do zmiany jednego formatu liczby na inny służy kowersja:
  Liczba całkowita <--> Liczba zmiennoprzecinkowa.

Adresowanie obiektów zmiennoprzecinkowych

Skróty	Pełen adres	Rodzaj obiektu zmiennoprzecinkowego	Dostęp	Sposób indeksowania
Wart. bezp.	-	wartości bezpośrednie	R	-
%MF	%MFi	wewn. wartość zmiennop.	R/W	%MFi[index]
%KF	%KFi	stała zmiennop.	R	%KFi[index]

Możliwość nakładania sie obiektów:
Słowa pojedyncze, podwójne i zmiennoprzecinkowe są przechowywane w obszarze danych, w pojedynczej strefie pamięci. Stąd, słowo zmiennoprzecinkowe %MFi odpowiada słowom pojedynczym %MWi i %MWi+1 (slowo %MWi zawiera bity mniej znaczące a słowo %MWi+1 zawiera bardziej znaczące bity słowa %MFi).

Przykłady:
%MF0 odpowiada %MW0 i %MW1,
a %KF543 odpowiada słowom %KW543 i %KW544.



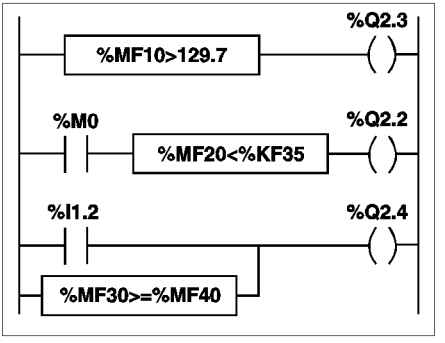
2.5-2 Instrukcje porównywania

Instrukcja porównywania umożliwia porównywanie dwóch argumentów będących obiektami zmiennoprzecinkowymi.

> : prawda, gdy argument 1 jest większy od argumentu 2,
>= : prawda, gdy argument 1 jest większy lub równy argumentowi 2,
< : prawda, gdy argument 1 jest mniejszy od argumentu 2,
<= : prawda, gdy argument 1 jest mniejszy lub równy argumentowi 2,
= : prawda, gdy argument 1 jest równy argumentowi 2,
<> : prawda, gdy argument 1 jest różny od argumentu 2.
Jeżeli warunek jest spełniony, to wynikiem porównania jest 1.

Struktura

Język *Ladder*



Bloki porównywania programuje się w strefie warunków (*test zone*).

Język *List*

```
LD      [%MF10 > 129.7]
ST      %Q2.3
LD      %M0
AND     [%MF20 < %KF35]
ST      %Q2.2
LD      %I1.2
OR      [%MF30 >= %MF40]
ST      %Q2.4
```

Operacja porównania zapisywana jest w nawiasach kwadratowych następujących po instrukcjach LD, AND i OR.

Język *ST*

```
%Q2.3 := %MF10 > 129.7 ;
%Q2.2 := (%MF20 < %KF35) AND %M0 ;
%Q2.4 := (%MF30 >= %MF40) OR %I1.2 ;
```

Składnia

Operatory: >,>=,<,<=,=,<>

Arg1 Operator Arg2

Argumenty

Typ	Argument 1i 2 (Arg1 i Arg2)
Indeksowalne obiekty zmiennoprzecinkowe	%MF,%KF
Nieindeks. obiekty zmiennoprzecinkowe	Zmiennoprzecinkowa wartość bezpośrednia. Zmiennoprzecinkowe wyrażenie numeryczne.

Uwaga

W języku *List*, instrukcje porównywania można zapisywać w nawiasach.

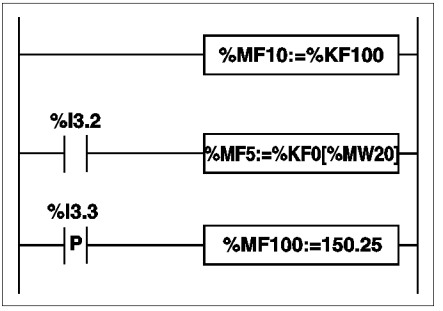
2.5-3 Instrukcje przypisania

Można wykonywać następujące operacje przypisania obiektów:

- zmiennoprzecinkowy (indeks.) -> zmiennoprzecinkowy (indeks.) Przykład 1
- zmiennoprzec. wartość bezp. -> zmiennoprzecinkowy (indeks.) Przykład 2

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[%MF10 := %KF100]              Przykład 1

LD          %I3.2
[%MF5:= %KF0[%MW20] ]          Przykład 1

LDR         %I3.3
[%MF100:=150.25]               Przykład 2
```

Język *ST*

```
%MF10 := %KF100 ;                  Przykład 1
IF %I3.2 THEN
    %MF5 := %KF0 [%MW20] ;         Przykład 1
END_IF ;
IF RE %I3.3 THEN
    %MF100 := 150.25 ;             Przykład 2
END_IF ;
```

Składnia

Operator :=

Arg1:=Arg2

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Indeks.wart. zmiennoprzec.	%MF	%MF,%KF
Nieindeksowalne wartości zmiennoprzecinkowe		Zmiennoprzecinkowa wartość bezp. Zmiennoprzecinkowe wyrażenie num.

Można stosować operacje wielokrotnego przypisywania.
Przykład: %MF0 := %MF2 := %MF4

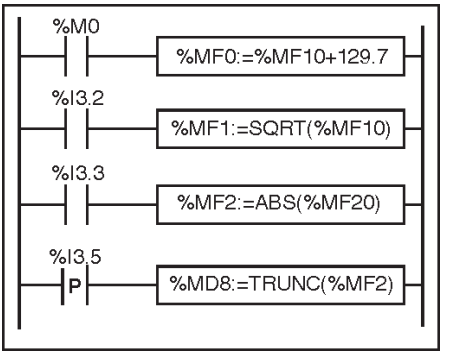
2.5-4 Instrukcje arytmetyczne

Sa to instrukcje umożliwiające wykonywanie operacji arytmetycznych na dwóch lub na jednym argumencie.

+	:	dodawanie dwóch argumentów	SQRT	: pierwiastek kwadratowy
-	:	odejmowanie dwóch argumentów	ABS	: wartość bezwzględna
*	:	mnożenie dwóch argumentów	TRUNC	: część całkowita wartości zmiennoprzecinkowej
/	:	dzielenie dwóch argumentów		

Struktura

Język *Ladder*



Jezyk *List*

```
LD          %M0
[%MF0 := %MF10 + 129.7]

LD          %I3.2
[%MF1 := SQRT(%MF10)]

LDR         %I3.3
[%MF2 := ABS(%MF20)]

LDR         %I3.5
[%MD8 := TRUNC(%MF2)]
```

Język *ST*

```
IF %M0 THEN
    %MF0 := %MF10 + 129.7 ;
END_IF ;
IF %I3.2 THEN
    %MF1 := SQRT (%MF10) ;
END_IF ;
IF RE %I3.3 THEN
    %MF2 := ABS (%MF20) ;
END_IF ;
IF RE %I3.5 THEN
    %MD8 := TRUNC (%MF2) ;
END_IF ;
```

Składnia

Operatory

- +,-,*,/ — Arg1:=Arg2 Operator Arg3
- SQRT, ABS, TRUNC — Arg1:=Operator(Arg2)

Argumenty

Typ	Argument 1 Arg1)	Argument 2 i 3 (Arg2 i 3)
Słowa indeksowalne	%MF (1)	%MF,%KF
Słowa nieindeksowalne		Zmiennoprzec. wartość bezp. Zmiennoprzec. wyr. numeryczne

(1) %MD dla instrukcji TRUNC.

Reguły

- Nie można bezpośrednio mieszać operacji na wartościach zmiennoprzecinkowych z operacjami na liczbach całkowitych. Poprzez zastosowanie konwersji można dopasowywać formaty wartości (patrz rozdział 2.6, część B).
- Bit systemowy %S18 pełni taką samą funkcję jak dla operacji na liczbach całkowitych (patrz rozdział 1.4-4, część B), słowo %SW17 zawiera informację o przyczynach wystąpienia błędu (patrz rozdział 2.5-1, część B).

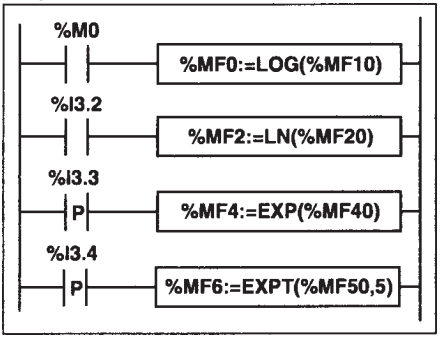
2.5-5 Instrukcje logarytmiczne i wykładnicze

Są to instrukcje umożliwiające wykonywanie operacji logarytmicznych i wykładniczych.

LOG : logarytm dziesiętny,
LN : logarytm naturalny,
EXP : exponent naturalny,
EXPT: operacja wykładnicza na wartości rzeczywistej z wykładnikiem całkowitym,

Struktura

Język *Ladder*



Jezyk *List*

```
LD          %M0
[%MF0 := LOG(%MF10)]

LD          %I3.2
[%MF2 := LN(%MF20)]

LDR         %I3.3
[%MF4 := EXP(%MF40)]

LDR         %I3.4
[%MF6 := EXPT(%MF50,5)]
```

Język *ST*

```
IF %M0 THEN
    %MF0 := LOG(%MF10) ;
END_IF ;
IF  %I3.2 THEN
    %MF2 := LN(%MF20) ;
END_IF ;
IF RE %I3.3 THEN
    %MF4 := EXP(%MF40) ;
END_IF ;
IF RE %I3.4 THEN
    %MF6 := EXPT(%MF50,5) ;
END_IF ;
```

Składnia

Operatory

- LOG, EXP, LN — Arg1:=Operator(Arg2)
- EXPT — Arg1:=Operator(Arg2,Arg3)

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)	Argument 3 (Arg3)
Słowa indeks.	%MF	%MF,%KF	%MW, %KW
Słowa nieindeks.		Zmiennop. wart. bezp. Zmiennop. wyr. num.	Całkowite wart. bezp. Całkowite wyr. num.

Reguły

- Jeżeli argumentem funkcji jest niewłaściwa wartość (np. logarytm z liczby ujemnej), to wynik jest nieokreślony lub nieskończony, co powoduje zmianę bitu %S18 na 1. Słowo %SW17 zawiera informację o przyczynach błędu (rozdział 2.5-1, część B).
- W przypadku funkcji logarytmicznych, dla wartości bliskich 1.0 (pomiędzy 0.99 a 1.0 lub 1.0 a 1.01) wynikiem jest 0, a bity %S18 i %SW17:X5 mają wartość 1.

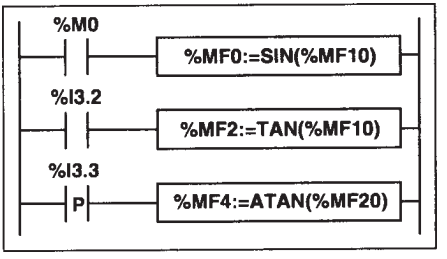
2.5-6 Instrukcje trygonometryczne

Te instrukcje umożliwiają wykonywanie obliczeń trygonometrycznych.

SIN : sinus kąta zapisanego w radianach
COS : kosinus kąta zapisanego w radianach
TAN : tangens kąta zapisanego w radianach
ASIN : arc sin (wynik od $-\pi/2$ do $\pi/2$)
ACOS : arc cos (wynik od 0 do π)
ATAN : arc tan (wynik od $-\pi/2$ do $\pi/2$)

Struktura

Język *Ladder*



Język *List*

```
LD        %M0
[%MF0 := SIN(%MF10)]

LD        %I3.2
[%MF2 := TAN(%MF10)]

LDR       %I3.3
[%MF4 := ATAN(%MF20)]
```

Język *ST*

```
IF %M0 THEN
    %MF0 := SIN(%MF10) ;
END_IF ;
IF %I3.2 THEN
    %MF2 := TAN(%MF10) ;
END_IF ;
IF RE %I3.3 THEN
    %MF4 := ATAN(%MF20) ;
END_IF ;
```

Składnia

Operatory

- SIN, COS, TAN
  ASIN, ACOS, ATAN

Arg1:=Operator(Arg2)

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MF	%MF,%KF
Słowa nieindeks.		Zmiennoprzec. wart. bezpośrednia Zmiennoprzec. wyrażenie num.

Reguły

- Jeżeli argument jest nieprawidłowy (np. arc cos liczby większej niż 1), to wynik jest nieokreślony, co powoduje nadanie bitowi %S18 wartości 1. Słowo %SW17 zawiera informację o przyczynach błędu (patrz rozdział 2.5-1, część B).
- Funkcje SIN/COS/TAN pozwalają na obliczenie wyniku dla wartości z przedziału od -4096π do 4096π ale dokładność bardzo mocno się zmniejsza dla kątów spoza przedziału od -2π do $+2\pi$, ze względu na modulo 2π, obliczane przed każdą operacją.

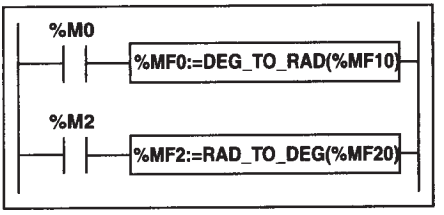
2.5-7 Instrukcje konwersji

Są to instrukcje umożliwiajce przeprowadzenie operacji konwersji:

DEG_TO_RAD : przeliczenie ze stopni na radiany, wynikiem jest wartość kąta z przedziału $0 \div 2\pi$,
RAD_TO_DEG : przeliczenie ze radianów na stopnie, wynikiem jest wartość kąta z przedziału od 0 do 360 stopni.

Struktura

Język *Ladder*



Jezyk *List*

```
LD          %M0
[%MF0 := DEG_TO_RAD(%MF10)]

LD          %M2
[%MF2 := RAD_TO_DEG(%MF20)]
```

Język *ST*

```
IF %M0 THEN
    %MF0 := DEG_TO_RAD(%MF10) ;
END_IF ;
IF %I3.2 THEN
    %MF2 := RAD_TO_DEG(%MF20) ;
END_IF ;
```

Składnia

Operatory

- DEG_TO_RAD
  RAD_TO_DEG

Arg1:=Operator(Arg2)

Argumenty

Typ	Argument1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MF	%MF,%KF
Słowa nieindeksowalne		Zmiennoprzec. wart. bezpośrednia Zmiennoprzec. wyrażenie numeryczne

Reguły

- Kąt po konwersji musi mieścić się w przedziale od -737280.0 do +737280.0 (dla konwersji DEG_TO_RAD) lub od -4096π do 4096π (dla konwersji RAD_TO_DEG). Dla wartości wykraczających poza ten limit, wynikiem jest + 1.#NAN, przy czym bity %S18 i %SW17:X0 mają wartość 1.

2.6 Instrukcje konwersji numerycznej

2.6-1 Konwersja BCD <--> Kod binarny

Dla tego typu konwersji można stosować 6 instrukcji:

- BCD_TO_INT : konwersja 16-bit. liczba BCD --> 16-bit. liczba całkowita,
- INT_TO_BCD : konwersja 16-bit. liczba całkowita --> 16-bit. liczba BCD,
- DBCD_TO_DINT : konwersja 32-bit. liczba BCD --> 32-bit liczba całkowita,
- DINT_TO_DBCD : konwersja 32-bit. liczba całkowita --> 32-bit. liczba BCD,
- DBCD_TO_INT : konwersja 32-bit. liczba BCD --> 16-bit. liczba całkowita,
- INT_TO_DBCD : konwersja 16-bit. liczba całkowita --> 32-bit. liczba BCD.

Kod BCD - przypomnienie:
Kod BCD (czyli *Binary Coded Decimal*) polega na zapisaniu cyfr dziesiętnych (0 do 9) za pomocą zapisu binarnego, przy użyciu 4 bitów. Słowo 16-bitowe może być liczbą 4-cyfrową ($0 \leq N \leq 9999$).

Zapis dziesiętny	0	1	2	3	4	5	6	7	8	9
Kod BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

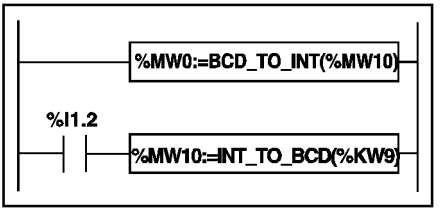
Przykład:

- Słowo %MW5 wyraża wartość BCD "2450", która odpowiada wartości binarnej: 0010 0100 0101 0000.
- Słowo %MW12 wyraża dziesiętną wartość "2450", w zapisie binarnym: 0000 1001 1001 0010.

Słowo %MW5 jest poddawane konwersji BCD_TO_INT do słowa %MW12.
Słowo %MW12 jest poddawane konwersji INIT_TO_BCD do słowa %MW5.

Struktura
Operacje wykonywane są nastepująco:

Język *Ladder*



Język *List*

```
LD          TRUE
[%MW0 := BCD_TO_INT(%MW10)]

LD          %I1.2
[%MW10 := INT_TO_BCD(%KW9)]
```

Język *ST*

```
%MW0 := BCD_TO_INT (%MW10) ;
IF %I1.2 THEN
    %MW10 := INT_TO_BCD (%KW9) ;
END_IF ;
```

Składnia
Operatory (konwersja liczby 16-bit)

- BCD_TO_INT
  INT_TO_BCD
  INT_TO_DBCD

Arg1:=Operator(Arg2)

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MW	%MW,%KW, %Xi.T
Słowa nieindeksowalne	%QW,%SW,%NW, %BLK	Wart. bezp.,%IW,%QW,%SW %NW,%BLK, Wyr. num.
Indeksowalne słowa podw.	%MD	
Nieindeksowalne słowa podw.	%QD, %SD	

Składnia
Operatory (konwersja liczby 32-bitowej)

- DBCD_TO_DINT
  DINT_TO_DBCD
  DBCD_TO_INT

Arg1:=Operator(Arg2)

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MW	
Słowa nieindeksowalne	%QW,%SW,%NW, %BLK	
Indeksowalne słowa podw.	%MD	%MD, %KD
Nieindeksowalne słowa podw.	%QD, %SD	Wart. bezp.,%ID,%QD,%SD Wyrażenia numeryczne

Przykładowe zastosowania
Instrukcja BCD_TO_INT służy do przetwarzania wartości nastaw dla wejść sterownika dokonywanych za pomocą pokręteł kodowanych w BCD.
Instrukcja INT_TO_BCD służy do wyświetlania wartości numerycznych (np. wynik obliczania wartości bieżącej dla bloku funkcyjnego) na wyświetlaczach kodowanych w BCD.

Reguły

- Konwersja BCD -->Kod binarny
  Przy konwersji BCD -->Kod binarny musi być pewność, że operator konwersji działa na wartości zakodowanej w BCD. Jeżeli wartość nie jest kodowana w BCD, to bit systemowy %S18 przyjmuje wartość 1, a wynik zawiera pierwszy błędny bajt 4-bitowy.
  Np. BCD_TO_INT(%MW2) gdzie: %MW2=4660 daje wynik 1234.
  Jednak, %MW2=242 (16#00F2) nadaje %S18 wartość 1, a wynikiem jest 15.

  Jeżeli dla instrukcji DBCD_TO_INT liczba w BCD jest większa niż 32767, to bit systemowy %S18 przyjmuje wartość 1, a wynikiem jest -1.

- Konwersja Kod binarny --> BCD
  W przypadku instrukcji INT_TO_BCD musi być pewność, że operator konwersji działa na wartości z przedziału od 0 do 9999 (lub 0 do 9999 9999). W innym razie bit systemowy %S18 przyjmuje wartość 1, a wynikiem jest wartość parametru wejściowego.
  Np. INT_TO_BCD(%MW2), gdzie: %MW2=2478 daje wynik 9336.
  Jednak, %MW2=10004 powoduje nadanie bitowi %S18 wartości 1, a wynikiem jest 10004.
  Jeżeli dla instrukcji INT_TO_DBCD parametr wejściowy jest ujemny, to bit systemowy %S18 przyjmuje wartość 1, a wynikiem jest wartość parametru wejściowego.

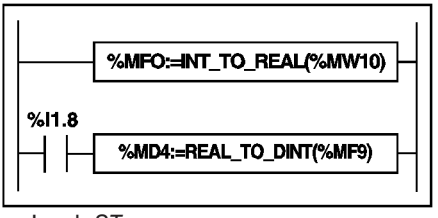
2.6-2 Konwersja Liczba całkowita <--> Liczba zmiennoprzecinkowa

Dla tego typu konwersji użytkownik ma do dyspozycji 4 instrukcje:

- INT_TO_REAL : Konwersja słowo całkowite --> słowo zmiennoprzecinkowe,
- DINT_TO_REAL : Konwersja podwójne słowo całkowite --> słowo zmiennoprzec.,
- REAL_TO_INT : Konwersja słowo zmiennoprzecinkowe --> słowo całkowite (wynikiem jest najbliższa wartość algebraiczna),
- REAL_TO_DINT : Konwersja słowo zmiennoprzec. -->podwójne słowo całkowite (wynikiem jest najbliższa wartość algebraiczna).

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[%MF0 := INT_TO_REAL(%MW10)]

LD          %I1.8
[%MD4 := REAL_TO_DINT(%MF9)]
```

Język *ST*

```
%MF0 := INT_TO_REAL (%MW10) ;
IF %I1.8 THEN
    %MD4 := REAL_TO_DINT (%MF9) ;
END_IF ;
```

Składnia
Operator

Arg1:=INT_TO_REAL(Arg2)

Argumenty

Typ	Argument 1 (Arg1)	Argumeny 2 (Arg2)
Słowa indeksowalne		%MW,%KW,%Xi.T
Nieindeksowalne słowa		Wart. bezp.,%IW,%QW,%SW %NW,%BLK,Wyr. num.
Indeks. słowa zmiennoprzec.	%MF	

Przykład: Konwersja słowo całkowite --> słowo zmiennoprzecinkowe: 147 --> 1.47e+02

Operator

Arg1:=DINT_TO_REAL(Arg2)

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne		%MD,%KD
Nieindeksowalne słowa		Wart. bezp.,%ID,%QD,%SD Wyrażenia numeryczne
Indeks. słowa zmiennoprzec.	%MF	

Przykład: Konwersja podwójne słowo indeks. -->słowo zmiennop.: 68905000 --> 6.8905e+07

Operator

Arg1:=REAL_TO_INT(Arg2)
Arg1:=REAL_TO_DINT(Arg2)

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MW	
Nieindeksowalne słowa	%QW,%NW,%BLK	
Indeks. słowa podwójne	%MD	
Nieindeksowalne słowa	%QD	
Indeks. słowa zmiennoprzec.		%MF,%KF
Nieindeks. słowa zmiennop.		Wartość bezp. zmiennoprzec.

Przykład: Konwersja słowo zmiennoprzec --> słowo całkowite 5978.6 --> 5978
Konwersja słowo zmiennop. --> całkowite słowo podw. -1235978.6 --> -1235979

Uwaga: Jeżeli podczas konwersji słowa zmiennoprzecinkowego na liczbę całkowitą (lub liczby zmiennoprzecinkowej na całe słowo podwójne), wartość zmiennoprzecinkowa przekracza limit słowa (lub słowa podwójnego), to bit %S18 przyjmuje wartość 1.

2.6-3 Konwersja Kod Gray'a --> Liczba całkowita

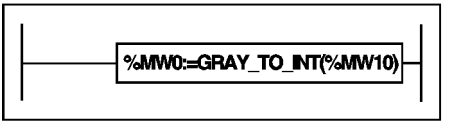
Instrukcja GRAY_TO_INT powoduje konwersję słowa zapisanego w kodzie Gray'a na liczbę całkowitą (czysty kod binarny).

Kod Gray'a - przypomnienie: Kod Gray'a lub "cykliczny kod binarny" służy do kodowania zmiennych wartości numerycznych za pomocą szeregu układów binarnych, które różnią się od siebie zmianą stanu pojedynczego bitu. Ten kod może np. służyć zapobieganiu przypadkowości: dla czystego kodu binarnego, zmiana wartości z 0111 na 1000 może spowodować wygenerowanie wartości 0 i 1000 ponieważ bity nie zmieniają swoich warości jednocześnie.

Kod dziesiętny	0	1	2	3	4	5	6	7	8	9
Kod BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Kod Gray'a	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101

Struktura

Język *Ladder*



Jezyk *List*

```
LD          TRUE
[%MW0 := GRAY_TO_INT(%MW10)]
```

Język *ST*

```
%MW0 := GRAY_TO_INT (%MW10) ;
```

Składnia

Operator

```
Arg1:=GRAY_TO_INT(Arg2)
```

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Słowa indeksowalne	%MW	%MW,%KW
Słowa nieindeksowalne	%QW,%SW,%NW,%BLK	Wart. bezp.,%IW,%QW,%SW %NW,%BLK,%Xi.T,Wyr. num.

2.6-4 Konwersja słowo <--> słowo podwójne

Opisane poniżej instrukcje mogą być stosowane w odniesieniu do obiektów o charakterze czysto symbolicznym (np. bloki funkcyjne DFB). W przypadku obiektów adresowalnych występuje zjawisko nadpisywania (np. słowo podwójne %MD0 składa się ze słów %MW0 i %MW1), co oznacza, że tych instrukcji nie można w stosunku do nich stosować.

- Wydzielanie słowa mniej znaczącego ze słowa podwójnego
  Instrukcja LW powoduje wydzielenie mniej znaczącego słowa ze słowa podwójnego i jego transfer do słowa pojedynczego.
  Składnia | Arg1:=LW(Arg2) |

  Arg1 = słowo pojedyncze (typu WORD)
  Arg2 = słowo podwójne (typu DWORD)

  Przykład: Cisnienie_paliwa:=LW(Parametr_1)
  jeśli Parametr_1 = 16#FFFF1234 to Cisnienie_paliwa = 16#1234

- Wydzielanie słowa bardziej znaczącego ze słowa podwójnego
  Instrukcja HW powoduje wydzielenie bardziej znaczącego słowa ze słowa podwójnego i jego transfer do słowa pojedynczego.
  Składnia | Arg1:=HW(Arg2) |

  Arg1 = słowo pojedyncze (typu WORD)
  Arg2 = słowo podwójne (typu DWORD)

  Przykład: Cisnienie_paliwa:=HW(Parametr_1)
  jeśli Parametr_1 = 16#FFFF1234 to Cisnienie_paliwa = 16#FFFF

- Łączenie dwóch słów
  Instrukcja CONCATW powoduje połączenie dwóch słów pojedynczych i zapisanie ich w słowie podwójnym.
  Składnia | Arg1:=CONCATW(Arg2,Arg3) |

  Arg1 = słowo podwójne (typu DWORD)
  Arg2 = słowo pojedyncze (typu WORD)
  Arg3 = słowo pojedyncze (typu WORD)

  Przykład: Cisnienie_paliwa:=CONCATW(Parametr_1,Parametr_2)
  jeśli Parametr_1 = 16#1234, Parametr_1 = 16#FFFF,
  to Cisnienie_paliwa = 16#FFFF1234

2.7 Tablice słów

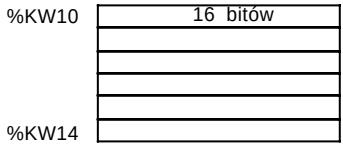
2.7-1 Wprowadzenie

Jezyk PL7 pozwala na wykonywanie operacji na tablicach:

- słów,
- słów podwójnej precyzji,
- słów zmiennoprzecinkowych.

Tablica słów o długości L, to sekwencja kolejnych słów tego samego typu.

Przykładowa tablica słów: %KW10:5



Typ	Format	Adres końcowy	Wielkość	Możliwość zapisu
Słowa wewnętrzne	Pojedyncze Podwójne Zmiennoprzecinkowe	%MWi:L %MDi:L %MFi:L	i+L≤ Nmax (1) i+L≤ Nmax-1 (1) i+L≤ Nmax-1 (1)	Tak Tak Tak
Słowa stałe	Pojedyncze Podwójne Zmiennoprzecinkowe	%KWi:L %KDi:L %KFi:L	i+L≤ Nmax (1) i+L≤ Nmax-1 (1) i+L≤ Nmax-1 (1)	Nie Nie Nie
Słowa systemowe	Pojedyncze	%SW50:4 (2)	-	Tak

(1) Nmax = maksymalna liczba słów zdefiniowana w konfiguracji.
(2) W formie tablic mogą być adresowane tylko słowa %SW50 do %SW53.

Ogólne zasady wykonywania operacji na tablicach

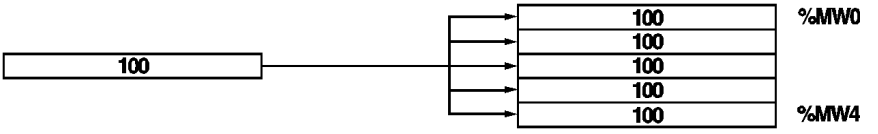
- Operacje wykonuje się tylko na tablicach zawierających obiekty jednego typu.
- Operacje mogą być wykonywane maksymalnie na dwóch tablicach.
- Jeżeli dwie tablice, na których są wykonywane operacje mają różne rozmiary, to tablica wynikowa będzie miała rozmiar odpowiadający tablicy mniejszej.
- Należy tak dobierać operacje na tablicach, by zapobiegać zjawisku nadpisywania (na przykład: %MW100[20]:=%MW90[20]+%KW100[20]).
- W przypadku operacji na dwóch tablicach działania są wykonywane po kolei na elementach tego samego rzędu tablic, a wynik jest zapisywany w tym samym rzędzie tablicy wynikowej.
- Jeżeli podczas wykonywania operacji bit systemowy %S18 zmieni stan na 1, to oznacza, że wynik danej operacji jest nieprawidłowy, ale operacje na kolejnych elementach tablic będą wykonywane poprawnie.
- Jeżeli argument jest wyrażeniem numerycznym, to należy go umieścić w nawiasie.
- Rząd tablicy odpowiada pozycji słowa w tablicy. Pierwsza pozycja odpowiada rzędowi 0.

2.7-2 Przypisywanie tablic słów

W odniesieniu do tablicy słów można stosować następujące operacje przypisania:

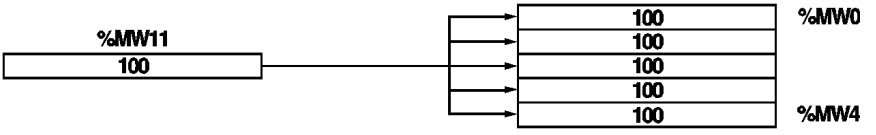
- wartość bezpośrednia -> tablica słów (indeksowana) Przykład 1
  podwójna wartość bezpośrednia -> podwójna tablica słów (indeksowana)
  zmiennoprzec. wartość bezpośrednia -> tablica słów zmiennoprzec. (indeksowana)

  Przykład 1: %MW0 :5:= 100



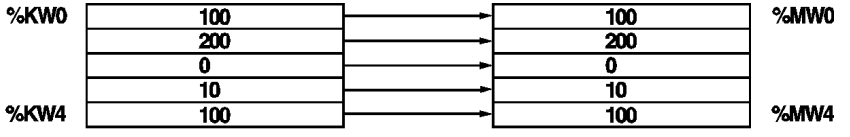
- słowo (indeksowane) -> tablica słów (indeksowana) Przykład 2
  słowo podwójne (indeksowane) -> tablica słów podwójnych (indeksowana)
  słowo zmiennoprzec. (indeksowane) -> tablica słów zmiennoprzec. (indeksowana)

  Przykład 2: %MW0 :5:= %MW11



- tablica słów (indeksowana) -> tablica słów (indeksowana) Przykład 3
  tablica słów podwójnych (indeksowana) -> tablica słów podwójnych (indeksowana)
  tablica słów zmiennop. (indeksowana) -> tablica słów zmiennoprzec. (indeksowana)

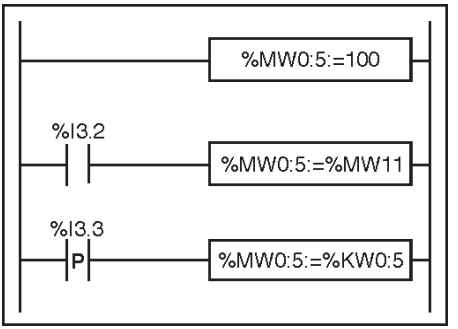
  Przykład 3: %MW0 :5:= %KW0:5



Uwaga:
Operacja wielokrotnego przypisania jest dopuszczalna dla przykładów 1 i 2
(%MW0:4 := %MW10:6 := %MW100)
ale nie dla przykładu 3.

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[%MW0 :5:= 100]          Przykład 1

LD          %I3.2
[%MW0:5 := %MW11]        Przykład 2
```

Język *ST*

```
IF RE %I3.3 THEN
    %MW0:5 := %KW0:5 ;  Przykład 3
END_IF ;
```

Składnia

Operator :=

Arg1 := Arg2

Tablica słów

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Tablica słów indeks.	%MW:L	%MW:L,%KW:L,%Xi.T:L
Słowa indeksowalne		%MW,%KW,%Xi.T
Słowa nieindeksowalne		Wart. bezp.,%IW,%QW,%SW %NW,%BLK,Wyr. numeryczne.

Tablica słów podwójnych

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Tablica słów indeks.	%MD:L	%MD:L,%KD:L
Słowa indeksowalne podw.		%MD,%KD
Słowa nieindeks. podw.		Wart. bezp.,%ID,%QD, Wyrażenie numeryczne

Tablica słów zmiennoprzecinkowych

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Tablica słów zmiennoprzec.	%MF:L	%MF:L,%KF:L
Indeks. słowa zmiennoprzec.		%MF,%KF
Nieindeksowalne słowa zmiennoprzecinkowe		Zmiennoprzec. wartość bezp. Zmiennoprzec. wyrażenie num.

Uwaga:
Wielokrotne przypisywanie tablic jest niedozwolone.

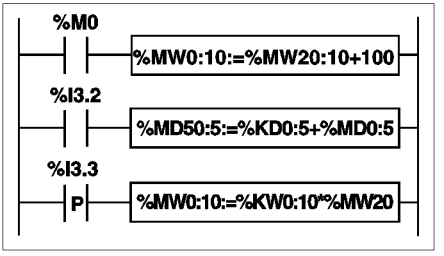
2.7-3 Operacje arytmetyczne na tablicach

Wymienione poniżej instrukcje umożliwiają wykonanie operacji arytmetycznych pomiędzy dwiema tablicami słów (albo między słowem a tablicą słów).

+ : dodawanie
- : odejmowanie
REM : reszta z dzielenia
* : mnożenie
/ : dzielenie

Struktura

Język *Ladder*



Jezyk *List*

```
LD          %M0
[%MW0:10:=%MW20:10+100]

LD          %I3.2
[%MD50:5:=%KD0:5 + %MD0:5]
```

Język *ST*

```
IF RE %I3.3 THEN
    %MW0:10 := %KW0:10 * %MW20 ;
END_IF ;
```

Składnia

Operatory

• +,-,*,/,REM

Arg1:=Arg2 Operator Arg3

Argumenty

Tablice słów

Typ	Argument 1 (Arg1)	Argumenty 2 i 3 (Arg2 i 3)
Tablica słów indeks.	%MW:L	%MW:L,%KW:L,%Xi.T:L
Słowa indeksowane		%MW,%KW,%Xi.T
Słowa nieindeksowane		Wart. bezp.,%IW,%QW,%SW %NW,%BLK,Wyr. num.

Tablice słów podwójnych

Typ	Argument 1 (Arg1)	Argumenty 2 i 3 (Arg2 i 3)
Tablica słów indeks.	%MD:L	%MD:L,%KD:L
Słowa indeksowane podw.		%MD,%KD
Słowa nieindeksowane podwójne		Wart.bezp.,%ID,%QD, Wyrażenia numeryczne

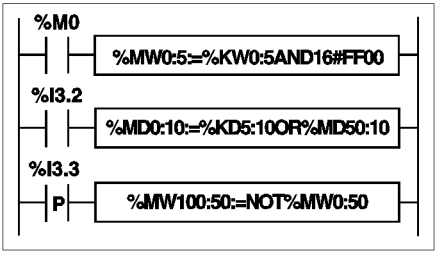
2.7-4 Operacje logiczne na tablicach

Przedstawione poniżej instrukcje służą do wykonywania operacji logicznych na dwóch tablicach słów (lub na słowie i tablicy słów).

AND : Koniunkcja AND (bit po bicie).
OR : Alternatywa OR (bit po bicie).
XOR : Nierównoważność (bit po bicie).
NOT : Logiczne dopełnienie (bit po bicie) tablicy (tylko 1 argument).

Struktura

Język *Ladder*



Jezyk *List*

```
LD          %M0
[%MW0:5:=%KW0: 5 AND 16#FF00]
```

Język *ST*

```
IF %I3.2 THEN
    %MD0:10 := %KD5:10 OR %MD50:10 ;
END_IF ;
IF RE%I3.3 THEN
    %MW100:50 := NOT %MW0:50 ;
END_IF ;
```

Składnia

Operatory

AND, OR, XOR `Arg1:=Arg2 Operator Arg3`

NOT `Arg1:=NOT Arg2`

Argumenty

Tablice słów

Typ	Argument 1 (Arg1)	Argumenty 2 i 3 (Arg2 i 3)
Tablica słów indeks.	%MW:L	%MW:L,%KW:L,%Xi.T:L
Słowa indeksowalne		%MW,%KW,%Xi.T
Słowa nieindeksowalne		Wart.bezp.,%IW,%QW,%SW %NW,%BLK,Wyr. numeryczne

Tablica słów podwójnych

Typ	Argument 1 (Arg1)	Argumenty 2 i 3 (Arg2 i 3)
Tablica słów indeks.	%MD:L	%MD:L,%KD:L
Słowa indeksowalne podw.		%MD,%KD,%SD
Podwójne słowa nieindeksowalne		Immed.val.,%ID,%QD, Numeric expr.

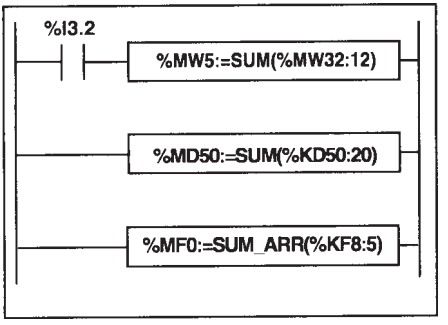
2.7-5 Sumowanie elementów tablicy

Funkcje SUM i SUM_ARR powodują dodawanie elementów tablicy:

- jeżeli tablica składa się ze słów pojedynczych, to wynik jest podany w formie słowa pojedynczego (funkcja SUM),
- jeżeli tablica składa się ze słów podwójnych, to wynik jest podany w formie słowa podwójnego (funkcja SUM),
- jeżeli tablica składa się ze słów zmiennoprzecinkowych, to wynik jest podany w formie słowa zmiennoprzecinkowego (funkcja SUM_ARR),

Struktura

Język *Ladder*



Jezyk *List*

```
LD          %I3.2
[%MW5:=SUM(%MW32:12)]
```

Język *ST*

```
%MD50 := SUM (%KD50:20) ;

%MF0 := SUM_ARR (%KF8:5) ;
```

Składnia

Funkcja

Wynik:=SUM(Tab)	Wynik:=SUM_ARR(Tab)

Parametry

Rodzaj	Wynik	Tablica (Tab)
Tablica słów indeksowalnych		%MW:L,%KW:L,%Xi.T:L
Słowa indeksowane	%MW	
Słowa nieindeksowane		%QW,%SW,%NW
Tablica indeks. słów podwójnych		%MD:L,%KD:L
Indeksowalne słowa podwójne	%MD	
Nieindeksowalne słowa podw.	%QD,%SD	
Tablica indeks. słów zmiennop.		%MF:L,%KF:L
Indeks. słowa zmiennoprzec.	%MF	

Note : Bit %S18 przyjmuje wartość 1, gdy wynik przekracza limity określone dla formatu słowa pojedynczego lub podwójnego (w zależności od rodzaju argumentu)

Przykład %MW5:=SUM(%MW30:4), z %MW30= 10, %MW31= 20, %MW32= 30, %MW33= 40, %MW5=10+20+30+40=100

2.7-6 Funkcja porównywania tablic

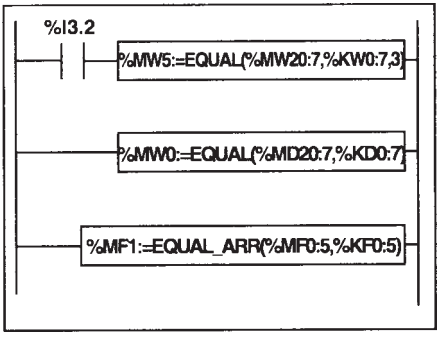
Funkcje EQUAL (dla liczb całkowitych) i EQUAL _ARR (dla liczb zmiennoprzecinkowych) pozwalają na porównywanie, element po elemencie, dwóch tablic.

W razie wykrycia jakiejś różnicy następuje zapisanie, w formie słowa, numeru rzędu pierwszego różniącego się elementu. Jeżeli z porównania wynika, ze nie ma różnic, to wynikiem funkcji jest -1.

Trzeci parametr zawiera informację o rzędzie tabeli, od którego ma się rozpocząć porównywanie (np. 0 aby rozpocząć porównywanie od początku tablic). Ten trzeci parametr jest opcjonalny (nie można go użyć w przypadku funkcji EQUAL_ARR). Jego pominięcie powoduje wykonanie porównania dla całej tablicy.

Struktura

Jezyk *Ladder*



Język *List*

```
LD          %I3.2
[%MW5:=EQUAL(%MW20:7,%KW0:7,3)]
```

Język *ST*

```
%MW0 := EQUAL (%MD20:7,%KD0:7)

%MW1 := EQUAL_ARR(%MF0:5,%KF0:5)
```

Składnia

Funkcja

```
Wynik:=EQUAL(Tab1,Tab2,rząd)
```

```
Wynik:=EQUAL_ARR(Tab1,Tab2)
```

Parametry

Typ	Wynik	Tablica (Tab)	Rząd
Tablica słów		%MW:L,%KW:L,%Xi.T:L	
Słowa indeksowalne	%MW		%MW,%KW, %Xi.T
Słowa nieindeksowalne	%QW,%SW, %NW		Wart. bezp. %QW, %IW,%SW,%NW Wyr. numeryczne
Tablica słów podwójnych		%MD:L,%KD:L	
Podwójne słowa indeks.	%MD		%MD,%KD
Podwójne słowa nieindeks.	%QD,%SD		Wart. bezp. %QD, %ID,%SD Wyr. numeryczne
Tablica słów zmiennoprzec.		%MF:L,%KF:L	
Słowa zmiennoprzecinkowe	%MF		

Uwaga:
- Tablice muszą mieć jednakowe długości.
- Jeżeli numer rzędu (parametr Rząd) jest większy niż długość tablicy, to wynik jest równy temu rzędowi.

Przykład %MW5:=EQUAL(%MW30:4,%KW0:4,1)

0 %MW30= 10 %KW0= 20
1 %MW31= 20 %KW1= 20
2 %MW32= 30 %KW2= 30
3 %MW33= 40 %KW3= 60 ==> %MW33 • %KW3==>%MW5= 3

2.7-7 Funkcje przeszukiwania tablic *Find*

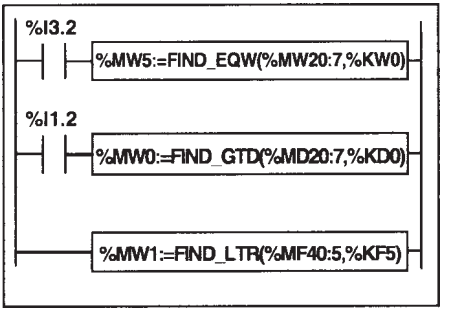
Oprogramowanie oferuje 11 funkcji przeszukiwania tablic:

- FIND_EQW : wyszukiwanie w tablicy, pozycji pierwszego elementu równego zadanej wartości,
- FIND_GTW : wyszukiwanie w tablicy, pozycji pierwszego elementu większego od zadanej wartości,
- FIND_LTW : wyszukiwanie w tablicy, pozycji pierwszego elementu mniejszego od zadanej wartości,
- FIND_EQD : wyszukiwanie w tablicy słów podwójnych, pozycji pierwszego elementu równego zadanej wartości,
- FIND_GTD : wyszukiwanie w tablicy słów podwójnych, pozycji pierwszego elementu większego od zadanej wartości,
- FIND_LTD : wyszukiwanie w tablicy słów podwójnych, pozycji pierwszego elementu mniejszego od zadanej wartości,
- FIND_EQR : wyszukiwanie w tablicy słów zmiennoprzecinkowych, pozycji pierwszego elementu równego zadanej wartości,
- FIND_GTR : wyszukiwanie w tablicy słów zmiennoprzecinkowych, pozycji pierwszego elementu większego od zadanej wartości,
- FIND_LTR : wyszukiwanie w tablicy słów zmiennoprzecinkowych, pozycji pierwszego elementu mniejszego od zadanej wartości,
- FIND_EQWP : wyszukiwanie w tablicy słów, pozycji pierwszego elementu równego wartości z danego rzędu,
- FIND_EQDP : wyszukiwanie w tablicy słów podwójnych, pozycji pierwszego elementu równego wartości z danego rzędu.

Wynikiem działania tych funkcji jest numer rzędu, w którym znajduje się pierwszy element w tablicy spełniający podany warunek lub wartość -1, gdy poszukiwanie kończy się fiaskiem.

Struktura

Język *Ladder*



Jezyk *List*

```
LD          %I3.2
[%MW5:=FIND_EQW(%MW20:7,%KW0)]
```

Język *ST*

```
IF %I1.2 THEN
    %MW0:=FIND_GTD(%MD20:7,%KD0) ;
END_IF ;

%MW1:=FIND_LTR(%MF40:5,%KF5) ;

%MW9:=FIND_EQWP(%MW30:8,%KF5,%MW4);
```

Składnia
Funkcja
FIND_EQW,FIND_GTW,FIND_LTW
FIND_EQD,FIND_GTD,FIND_LTD
FIND_EQR,FIND_GTR,FIND_LTR

Wynik:=Funkcja(Tab,Wart)

FIND_EQWP,FIND_EQDP

Wynik:=Funkcja(Tab,Wart,Rząd)

Parametry

Tablice słów (FIND_EQW,FIND_GTW,FIND_LTW,FIND_EQWP)

Typ	Wynik	Tablica (Tab)	Wartość (Wart), Rząd
Tablica słów indeks.		%MW:L,%KW:L,%Xi.T:L	
Słowa indeksowalne	%MW		%MW,%KW, %Xi.T
Słowa nieindeksowalne	%QW,%SW %NW		Wart. bezp.%QW, %IW,%SW,%NW, Wyr. numeryczne

Tablice słów podwójnych (FIND_EQD,FIND_GTD,FIND_LTD,FIND_EQDP)

Typ	Wynik	Tablica (Tab)	Wartość (Wart)
Tablica słów indeks.		%MD:L,%KD:L	
Słowa indeksowalne (podw.)	%MW		%MD,%KD
Słowa nieindeksowalne (podwójne)	%QW,%SW %NW		Wart. bezp. %QD, %ID,%SD Wyr. numeryczne

Uwaga: Dla parametru Rząd patrz tablica słów (jak dla FIND_EQWP)

Tablice słów zmiennoprzecinkowych (FIND_EQR,FIND_GTR,FIND_LTR)

Typ	Wynik	Tablica (Tab)	Wartość (Wart)
Tablica słów zmiennoprzec.		%MF:L,%KF:L	
Słowa indeks. zmiennoprzec.	%MW		%MF,%KF
Nieindeksowalne słowa zmiennoprzecinkowe	%QW,%SW %NW		Wartości bezp. Wyrażenia num.

Przykład %MW5:=FIND_EQW(%MW30:4,%KW0)
Rząd
0 %MW30= 10
1 %MW31= 20
2 %MW32= 30 ==> %KW0= 30 ==> %MW5= 2
3 %MW33= 40

2.7-8 Wyszukiwanie najmniejszej i największej wartości w tablicy

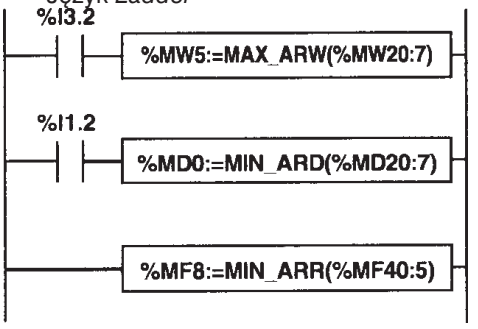
Oprogramowanie oferuje 6 funkcji:

- MAX_ARW : wyszukiwanie wartości maksymalnej w tablicy słów,
- MIN_ARW : wyszukiwanie wartości minimalnej w tablicy słów,
- MAX_ARD : wyszukiwanie wartości maksymalnej w tablicy słów podwójnych,
- MIN_ARD : wyszukiwanie wartości minimalnej w tablicy słów podwójnych,
- MAX_ARR : wyszukiwanie wartości maksymalnej w tablicy słów zmiennoprzec.,
- MIN_ARR : wyszukiwanie wartości minimalnej w tablicy słów zmiennoprzec.

Wynikiem tych funkcji jest wartość równa wartości maksymalnej (lub minimalnej) znalezionej w tablicy.

Struktura

Język *Ladder*



Język *List*

```
LD          %I3.2
[%MW5:=MAX_ARW(%MW20:7)]
```

Język *ST*

```
IF %I1.2 THEN
    %MD0 := MIN_ARD (%MD20:7) ;
END_IF ;
%MF8 := MIN_ARR (%MF40:5) ;
```

Składnia

Funkcja
MAX_ARW,MIN_ARW
MAX_ARD,MIN_ARD
MAX_ARR,MIN_ARR

```
Wynik:=Funkcja(Tab)
```

Parametry

Typ	Wynik	Tablica (Tab)
Tablica słów indeksowalnych		%MW:L,%KW:L,%Xi.T:L
Słowa indeksowalne	%MW	
Słowa nieindeksowalne		%QW,%SW,%NW
Tablica indeks. słów podwójnych		%MD:L,%KD:L
Indeksowalne słowa podwójne	%MD	
Nieindeksowalne słowa podwójne	%QD,%SD	
Tablica słówa zmiennoprzec.		%MF:L,%KF:L
Indeksowalne słowa zmiennoprzec.	%MF	

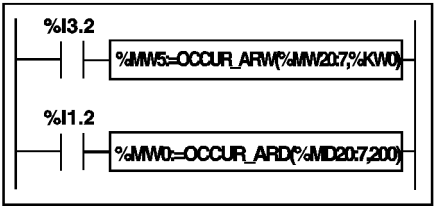
2.7-9 Liczba wystąpień danej wartości w tablicy

Oprogramowanie daje do dyspozycji 3 funkcje:

- OCCUR_ARW : przeszukiwanie tablicy słów, w celu obliczenia ilości elementów równych zadanej wartości,
- OCCUR_ARD : przeszukiwanie tablicy słów podwójnych, w celu obliczenia ilości elementów równych zadanej wartości,
- OCCUR_ARR : przeszukiwanie tablicy słów zmiennoprzecinkowych, w celu obliczenia ilości elementów równych zadanej wartości.

Struktura

Język *Ladder*



Język *List*

```
LD          %I3.2
[%MW5:=OCCUR_ARW(%MW20:7,%KW0)]
```

Język *ST*

```
IF %I1.2 THEN
   %MW0:=OCCUR_ARD(%MD20:7,200) ;
END_IF ;
```

Składnia
Funkcja
OCCUR_ARW
OCCUR_ARD
OCCUR_ARR

Wynik:= Funkcja (Tab,Wart)

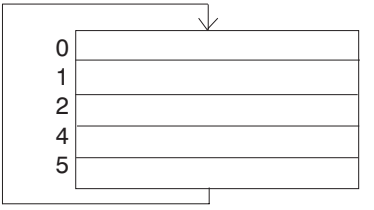
Parametry

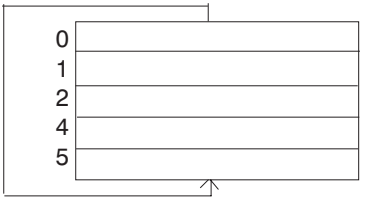
Typ	Wynik	Tablica (Tab)	Wartość (Wart)
Tablica słów indeksowalnych		%MW:L,%KW:L,%Xi.T:L	
Słowa indeksowalne	%MW		MW,%KW,%Xi.T
Słowa nieindeksowalne	%QW,%SW, %NW		Wart. bezp. %QW, %IW,%SW,%NW Wyr. numeryczne
Tablica podwójnych słów ind.		%MD:L,%KD:L	
Indeks. słowa podwójne	%MW		%MD,%KD
Nieindeks. słowa podwójne	%QW,%SW, %NW		Wart. bezp. %QD, %ID,%SD Wyr. numeryczne
Tablica słów zmiennop.		%MF:L,%KF:L	
Indeks. słowa zmiennoprzec.	%MW		%MF,%KF
Nieindeksowalne słowa zmiennoprzecinkowe	%QW,%SW, %NW		Wart. bezpośrednie Wyr. numeryczne

2.7-10 Okrężne przesuwanie elementów tablicy

Z przesuwaniem elementów tablicy związanych jest 6 funkcji:

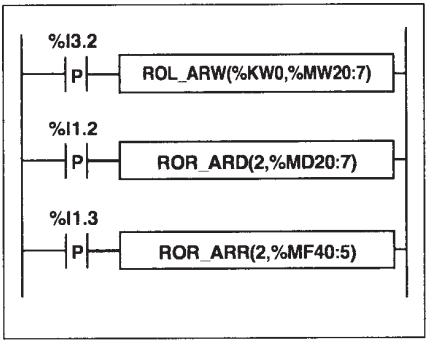
- ROL_ARW: powoduje okrężne przesunięcie z góry do dołu elementów tablicy słów o *n* pozycji.
- ROL_ARD: powoduje okrężne przesunięcie z góry do dołu elementów tablicy słów podwójnych o *n* pozycji.
- ROL_ARR: powoduje okrężne przesunięcie z góry do dołu elementów tablicy słów zmiennoprzec. o *n* pozycji.
- ROR_ARW: powoduje okrężne przesunięcie z dołu do góry elementów tablicy słów o *n* pozycji.
- ROR_ARD: powoduje okrężne przesunięcie z dołu do góry elementów tablicy słów podwójnych o *n* pozycji.
- ROR_ARR: powoduje okrężne przesunięcie z dołu do góry elementów tablicy słów zmiennoprzec. o *n* pozycji.





Struktura

Język *Ladder*



Język *List*

```
LDR         %I3.2
[ROL_ARW(%KW0,%MW20:7)]
```

Język *ST*

```
IF RE%I1.2 THEN
    ROR_ARD (2,%MD20:7) ;
END_IF ;
IF RE%I1.3 THEN
    ROR_ARR (2,%MF40:5) ;
END_IF ;
```

Składnia

Funkcje ROL_ARW,ROR_ARW

Funkcja(n,Tab)

Parametry
Tablice słów

Typ	Liczba pozycji (n)	Tablica (Tab)
Tablica słów indeksowalnych		%MW:L
Słowa indeksowalne	%MW,%KW,%Xi.T	
Słowa nieindeksowalne	Wart. bezp.%QW,%IW,%SW %NW,Wyr. numeryczne	

Funkcje ROL_ARD,ROR_ARD

Funkcja(n,Tab)

Parametry
Tablice słów podwójnych

Typ	Liczba pozycji (n)	Tablica (Tab)
Tablica słów indeksowalnych		%MD:L
Słowa indeksowalne	%MW,%KW,%Xi.T	
Słowa nieindeksowalne	Wart. bezp.%QW,%IW,%SW %NW,Wyr. numeryczne	

Funkcje ROL_ARR,ROR_ARR

Funkcja(n,Tab)

Parametry
Tablice słów zmiennoprzecinkowych

Typ	Liczba pozycji (n)	Tablica (Tab)
Tablice indeksowalnych słów zmiennoprzecinkowych		%MF:L
Słowa indeksowalne	%MW,%KW,%Xi.T	
Słowa nieindeksowalne	Wart. bezp.%QW,%IW,%SW %NW,Wyr. numeryczne	

Uwaga: Jeżeli wartość *n* jest ujemna albo równa zeru, to nie wykonuje się żadnego przesunięcia.

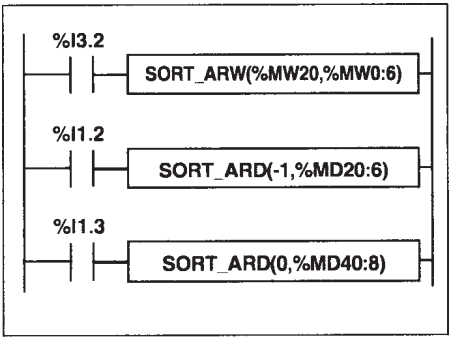
2.7-11 Funkcje sortowania elementów tablicy

Oprogramowanie zawiera 3 funkcje sortowania:

- SORT_ARW : sortowanie elementów tablicy słów, w porządku rosnącym albo malejącym; wynik operacji jest zapisywany w tej samej tablicy,
- SORT_ARD : sortowanie elementów tablicy słów podwójnych w porządku rosnącym albo malejącym; wynik operacji jest zapisywany w tej samej tablicy,
- SORT_ARR : sortowanie elementów tablicy słów zmiennoprzecinkowych, w porządku rosnącym albo malejącym; wynik operacji jest zapisywany w tej samej tablicy,

Struktura

Język *Ladder*



Język *List*

```
LD          %I3.2
[SORT_ARW(%MW20,%MW0:6)]
```

Język *ST*

```
IF %I1.2 THEN
    SORT_ARD (-1,%MD20:6) ;
END_IF ;
IF %I1.3 THEN
    SORT_ARR (0,%MF40:8) ;
END_IF ;
```

Składnia

Funkcja
SORT_ARW
SORT_ARD
SORT_ARR

```
Funkcja (Kier,Tab)
```

- Parametr "*Kier*" określa sposób sortowania: jeżeli jest ≥ 0, to tablica jest sortowana w porządku rosnącym. Gdy < 0, to sortowanie odbywa się w porządku malejącym.
- Wynik (posortowana tablica) zawiera się w parametrze *Tab* (tablica do sortowania).

Parametry

Tablice słów

Typ	Kierunek sortowania	Tablica (Tab)
Tablica słów (SORT_ARW)		%MW:L
Tablica słów podw. (SORT_ARD)		%MD:L
Tablica słów zmiennoprz. (SORT_ARF)		%MF:L
Słowa indeksowalne	%MW,%KW	
Słowa nieindeksowalne	Wart. bezp.%QW,%IW,%SW %NW,Wyr. numeryczne	

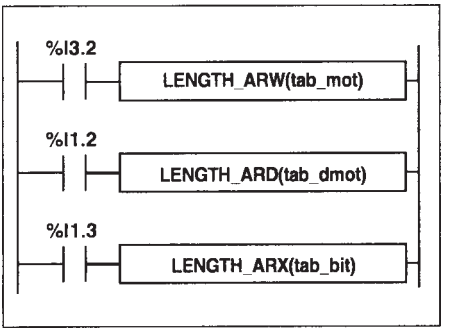
2.7-12 Obliczanie długości tablicy

Są 4 funkcje obliczania długości tablicy, które mogą być wykorzystywane do programowania bloków DFB, gdy długość tablicy nie została jawnie zdefiniowana:

- LENGTH_ARW : oblicza liczbę elementów (długość) tablicy słów,
- LENGTH_ARD : oblicza liczbę elementów (długość) tablicy słów podwójnych,
- LENGTH_ARR : oblicza liczbę elementów (długość) tablicy słów zmiennoprzecinkowych,
- LENGTH_ARX : oblicza liczbę elementów (długość) tablicy bitowej.

Struktura

Język *Ladder*



Język *List*

```
LD          %I3.2
[LENGTH_ARW(tab_mot)]
```

Język *ST*

```
IF %I1.2 THEN
    LENGTH_ARD(tab_dmot)]
END_IF ;
IF %I1.2 THEN
    LENGTH_ARX(tab_bit)]
END_IF ;
```

Składnia

Funkcja
LENGTH_ARW
LENGTH_ARD
LENGTH_ARR
LENGTH_ARX

Wynik = Funkcja (Tab)

Parametry

Tablica słów

Typ	Tablica (Tab)	Wynik
Tablica (LENGTH_ARW)	słów	
Tablica (LENGTH_ARD)	słów podwójnych	
Tablica (LENGTH_ARR)	słów zmiennoprzecinkowych	
Tablica (LENGTH_ARX)	bitowa	
Słowa indeksowalne		%MW
Słowa nieindeksowalne		%QW,%SW,%NW

Uwaga: Parametry tablicy są obiektami o charakterze czysto symbolicznym.

2.8 Operacje na łańcuchach znaków

2.8-1 Format łańcucha lub tablicy znaków

Tablica znaków składa się z serii bajtów, w których mogą być łańcuchy znaków. Rozmiar tablicy określa maksymalną długość łańcucha znaków (do 255 znaków).

Przykład: %MB4:6 jest tablicą składającą się z 6 bajtów zawierającą łańcuch do 6 znaków.

Pierwszy bajt tablicy musi być parzysty (nie można zapisać tablicy bajtów zaczynającej się od bajtu nieparzystego, np. %MB5:6).

Tablice bajtów zajmują ten sam obszar pamięci co słowa %MW i %MD. Stąd też występuje niebezpieczeństwo nadpisania (*overlap*): patrz rozdział 1.2-4, część A.

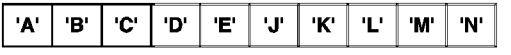
Określeniem "łańcuch znaków" nazywa się wszystkie znaki pomiędzy początkiem tablicy, a pierwszym znakiem końca łańcuchów znaków.

Znak NUL (kod 00) nazywany jest znakiem końca łańcucha. W niniejszej sekcji będzie on oznaczany symbolem .

Przykłady:

- Poniższa tablica (12-elementowa) zawiera łańcuch znaków 'ABCDE' (5 znaków).

- Tablica (10-elementowa) zawiera łańcuch znaków 'ABCDEJKLMN' (10 znaków).



Tak więc długość łańcucha znaków jest określona albo przez liczbę znaków znajdujących sie przed znakiem końca łańcucha , albo przez rozmiar tablicy (gdy nie ma znaku końca).

Uwagi:

Bit systemowy %S15 przyjmuje wartość 1 w następujących okolicznościach:

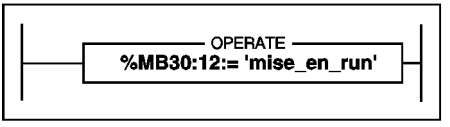
- podczas zapisywania łańcucha znaków dłuższego niż rozmiar tablicy (nie można umieścić znaku końca łańcucha znaków).
- gdy użytkownik próbuje uzyskać dostęp do znaku nie wchodzącego w skład danego łańcucha.
- gdy użyte parametry są niewłaściwe:
  zerowa długość łańcucha, który ma być usunięty (funkcja DELETE), zerowa długość łańcucha, z którego mają być wydzielone znaki (funkcja MID), zerowa długość łańcucha, który ma być zastąpiony (funkcja REPLACE), próba wyszukania ciągu znaków, którego długość przekracza długość łańcucha (funkcja FIND).

2.8-2 Przypisywanie łańcuchów znaków

Funkcja ta umożliwia transfer łańcucha znaków do tablicy bajtów o długości *L*.

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[ %MB30:10 := 'set_to_run']
```

Język *ST*

```
%MB30:10:='set_to_run' ;
```

Przykład Transfer łańcucha znaków 'set_to_run' do tablicy bajtów o długości 10 znaków

%MB	30	31	32	33	34	35	36	37	38	39
	's'	'e'	't'	'_'	't'	'o'	'_'	'r'	'u'	'n'

Składnia

Operator Arg1:=Arg2

Argumenty

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)
Tablice bajtów	%MB:L	%MB:L,%KB:L Wartość bezpośrednia

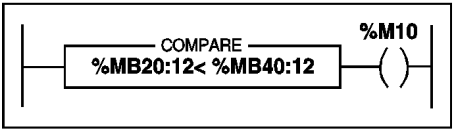
2.8-3 Porównywanie alfanumeryczne

Funkcja ta umożliwia porównywanie dwóch łańcuchów znaków (parametry funkcji) zapisanych w tablicy bajtów. Porównywanie odbywa się znak po znaku. W wyniku, otrzymuje się bit, który przyjmuje wartość 1, gdy oba łańcuchy spełniają określony przez funkcję warunek (znak po znaku). W przeciwnym razie bit ma wartość 0.

Porządek znaków określa tablica kodów ASCII (ISO 646). Na przykład łańcuch 'Z' jest większy niż łańcuch 'AZ', który jest większy niż łańcuch 'ABC'.

Struktura

Język *Ladder*



Język *List*

```
LD    [ %MB20:12 < %MB40:12]
ST          %M10
```

Bloki porównywania programuje się w strefie warunków (*test zone*).

Operacje porównywania zapisuje się w nawiasach kwadratowych umieszczanych przed instrukcjami LD, AND i OR.

Język *ST*

```
%M10 := %MB20:12 < %MB40:12 ;
```

Przykład: %MB20:12 < %MB40:12 ==> TAK jeśli wynikiem jest 1

gdzie

Elementy następujące po znaku końca łańcucha nie są brane pod uwagę.

Składnia

Operator

<, >, <=, >=, =, < >

```
Arg1 Operator Arg2
```

Argumenty

Typ	Argument 1 (Arg1) i Argument 2 (Arg2)
Tablice bajtów	%MB:L,%KB:L, wartość bezpośrednia

2.8-4 Konwersja Wartość numeryczna <---> Kod ASCII

Funkcja ta umożliwia konwersję wartości numerycznej (lub zmiennoprzecinkowej) na łańcuch znaków ASCII i w kierunku odwrotnym.
Wynik konwersji musi być zapisany, za pomocą operacji przypisania, w obiekcie PL7: tablica bajtów, słowo pojedynczej lub podwójnej precyzji, słowo zmiennoprzecinkowe.

Możliwe są następujące konwersje:

INT_TO_STRING	Kod binarny --> Kod ASCII
DINT_TO_STRING	Kod binarny --> Kod ASCII
STRING_TO_INT	Kod ASCII --> Kod binarny
STRING_TO_DINT	Kod ASCII --> Kod binarny
REAL_TO_STRING	Obiekt zmiennoprzecinkowy --> Kod ASCII
STRING_TO_REAL	Kod ASCII --> Obiekt zmiennoprzecinkowy.

Format obiektów zmiennoprzecinkowych: ==> patrz rozdział 2.5, część B

Kod ASCII:
Wszystkie 256 znaków alfanumerycznych i sterujących można zapisać w 8 bitach. Kod ten określany mianem ASCII (*American Standard Code for Information Interchange*) jest zgodny z ideą bajtu. Tak więc można stworzyć tablicę składającą się z *n* bajtów za pomocą *n* kodów ASCII definiujących *n* znaków.

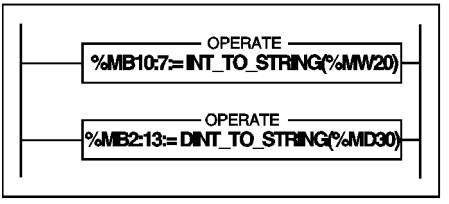
2.8-5 Konwersja Kod binarny ---> Kod ASCII

Ta funkcja służy do przeliczania wartości numerycznej (słowo pojedynczej lub podwójnej precyzji) na łańcuch kodów ASCII.
W kodzie ASCII odwzorowywana jest każda cyfra oraz znak wartości (parametry). Tworzą one wynikową tablicę bajtów.

- Funkcja INT_TO_STRING: słowo pojedyncze może mieć wartość z przedziału od -32768 do +32767, czyli 5 cyfr plus znak. Wynikiem więc będzie tablica o długości 6 znaków plus znak końca łańcucha znaków. Znak '+' lub '-' jest zapisywany w pierwszym znaku, cyfry jednostkowe w znaku szóstym, dziesiątki w piątym, itd.
- Funkcja DINT_TO_STRING: słowo podwójne może mieć wartość z przedziału od -2147483648 do +2147483647, czyli 10 cyfr plus znak. Wynikiem więc będzie tablica o długości 12 znaków plus znak końca łańcucha znaków. Znak '+' lub '-' jest zapisywany w pierwszym znaku, cyfry jednostkowe w znaku dwunastym, dziesiątki w jedenastym, itd. Drugim znakiem jest zawsze '0'.

Struktura

Język *Ladder*



Język *List*

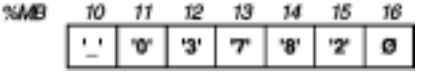
```
LD          TRUE
[ %MB10:7 := INT_TO_STRING (%MW20)]
```

Język *ST*

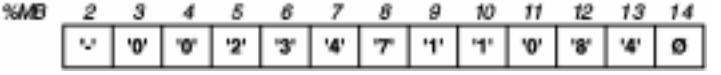
```
%MB2:13:=DINT_TO_STRING (%MD30) ;
```

Przykład: Konwersja Kod binarny ---> Kod ASCII

%MB10:7 := INT_TO_STRING (%MW20) gdzie %MW20 = - 3782 w kodzie dziesiętnym ==> wynik zapisywany jest w 7-bajtowej tablicy %MB10:7



Przykład: %MB2:13 := DINT_TO_STRING (%MD30) gdzie %MD30 = - 234701084



Składnia

Operator

```
Wynik := INT_TO_STRING (wartość)
```

Argumenty

Typ	Wynik	Wartość
Tablica 6-bajtowa + znak końca łańcucha	%MB:7	
Słowa indeksowalne		%MW, %KW, %Xi.T
Słowa nieindeksowalne	Wart. bezp., Wyr. num.	%IW,%QW,%SW,%NW

Operator

```
Wynik := DINT_TO_STRING (wartość)
```

Argumenty

Typ	Wynik	Wartość
Tablica 12-bajtowa + znak końca łańcucha	%MB:13	
Podwójne słowa indeksowalne		%MD, %KD
Podwójne słowa nieindeks.		%ID,%QD,%SD, Wart. bezp., Wyrażenie num.

2.8-6 Konwersja ASCII ---> Kod binarny

Funkcja ta umożliwia konwersję łańcucha znaków reprezentującego wartość numeryczną na kod binarny (wynik jest zapisywany w słowie pojedynczym lub podwójnym). Każdy element tablicy będący parametrem reprezentuje kod ASCII danego znaku. Mogą to być cyfry i znaki '+' i '-'.

- Funkcja STRING_TO_INT: konwersja łańcucha 6-znakowego, którego wartość mieści się w przedziale od -32768 do +32767. Pierwszy znak reprezentuje znak liczby, a pozostałe - jej wartość. Drugi znak odpowiada dziesiątkom tysięcy, ..., szósty - jednostkom. Wartość w łańcuchu musi być wyrównana do prawej strony.
- Funkcja STRING_TO_DINT: konwersja łańcucha 12-znakowego, którego wartość mieści się w przedziale od -2147483648 do +2147483647. Pierwszy znak reprezentuje znak liczby a pozostałe - jej wartość. Drugim znakiem jest '0', trzeci znak odpowiada tysiącom milionów, ..., dwunasty - jednostkom. Wartość w łańcuchu musi być wyrównana do prawej strony.

Struktura


```
LD            TRUE
[%MW13 := STRING_TO_INT (%MB20:7)]
```

Przykład: %MW13 := STRING_TO_INT (%MB20:7) gdzie

%MB	20	21	22	23	24	25	26
	'_'	'0'	'2'	'3'	'4'	'7'	Ø

==> wynik %MW13 = -2347 w kodzie dziesiętnym

Składnia

Operator: Wynik := STRING_TO_INT (łańcuch)

Argumenty

Typ	Wynik	Łańcuch
Słowa indeksowalne	%MW	
Słowa nieindeksowalne	%QW,%SW,%NW.	
Tablice 6-bajtowe + znak końca łańcucha		%MB:7,%KB:7, Wart. bezp.

Bit %S18 przyjmuje wartość 1 wtedy, kiedy wartość z łańcucha znaków nie mieści się w przedziale -32768 do +32767 lub gdy jeden z 6 znaków jest nieprawidłowy.

Operator: Wynik := STRING_TO_DINT (łańcuch)

Argumenty

Typ	Wynik	Łańcuch
Indeks. słowa podwójne		%MD
Nieindeks. słowa podwójne	%QD,%SD	
Tablice 12-bajtowe + znak końca łańcucha		%MB:13,%KB:13, Wart. bezp.

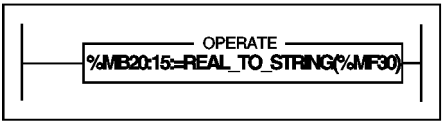
Bit %S18 przyjmuje wartość 1 wtedy, kiedy wartość z łańcucha znaków nie mieści się w przedziale -2147483648 do +2147483647 lub gdy jeden z 12 znaków jest nieprawidłowy.

2.8-7 Konwersja Wartość zmiennoprzecinkowa ---> Kod ASCII

Jest to funkcja umożliwiająca konwersję rzeczywistej wartości numerycznej, zapisanej w słowie zmiennoprzecinkowym, na łańcuch znaków ASCII. Wynik zapisywany jest w tablicy składającej się z 14 bajtów + znak końca łańcucha.
Przeliczane na kod ASCII są wszystkie cyfry wartości numerycznej i znaki '+', '-', '.', 'e' oraz 'E (zapisywane są w tablicy wynikowej).
Pierwszym znakiem jest znak liczby, znak dziesiętny (.) jest trzecim znakiem a exponent 'e' jest znakiem jedenastym, natomiast znak exponentu - dwunastym.

Struktura

Język *Ladder*



Język *List*

```
LD              TRUE
[ %MB20:15 := REAL_TO_STRING (%MF30)]
```

Język *ST*

```
%MB20:15 := REAL_TO_STRING (%MF30) ;
```

Przykład: %MB20:15 := REAL_TO_STRING (%MF30)
gdzie %MF30 = - 3.234718 e26 ==> wynik

%MB	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
	'-'	'3'	'.'	'2'	'3'	'4'	'7'	'1'	'8'	'0'	'e'	'+'	'2'	'6'	Ø

Składnia

Operator

```
Wynik := REAL_TO_STRING (wartość)
```

Argumenty

Typ	Wynik	Wartość
Tablice 14-bajtowe + znak końca łańcucha	%MB:15	
Słowa indeksowalne		%MF, %KF
Słowa nieindeksowalne		Wartość bezp., Wyr. num.

Bit %S18 przyjmuje wartość 1, gdy będąca parametrem wartość zmiennoprzecinkowa jest spoza przedziału od -3.402824e+38 do -1.175494e-38 lub +1.175494e-38 do +3.402824e+38. W takim przypadku, wynik jest nieprawidłowy.

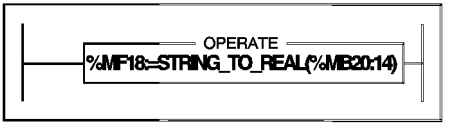
2.8-8 Konwersja Kod ASCII --> Wartość zmiennoprzecinkowa

Funkcja ta umożliwia konwersję łańcucha znaków reprezentującego rzeczywistą wartość numeryczną na wartość zmiennoprzecinkową (wynik jest zapisywany w słowie zmiennoprzecinkowym).
Każdy element tablicy będący parametrem reprezentuje kod ASCII jednego znaku. Mogą to być cyfry oraz znaki '+', '-', '.', 'e' i 'E'. Nie stosuje się znaku końca łańcucha. Oznacza to, że wszystkich 14 bajtów tablicy musi być poprawnych.

Pierwszym znakiem jest znak liczby, kropka dziesiętna (.) umieszczana jest na trzecim miejscu, znak 'e' - na jedenastym a znak exponentu - na dwunastym.
Dla przykładu: wartość 3.12 musi być zapisana w formie '+3.1200000e+00'.

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[ %MF18 := STRING_TO_REAL (%MB20:14)]
```

Język *ST*

```
%MF18 := STRING_TO_REAL (%MB20:14) ;
```

Przykład: %MF18 := STRING_TO_REAL (%MB20:14)

gdzie

%MB	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
	'-'	'3'	'.'	'2'	'3'	'4'	'7'	'1'	'8'	'0'	'e'	'+'	'2'	'6'	Ø

==> wynik %MF18 = - 3.234718e26

Składnia

Operator

```
Wynik := STRING_TO_REAL (łańcuch)
```

Argumenty

Typ	Wynik	Łańcuch
Słowa indeksowalne	%MF	
Tablice 14-bajtowe		%MB:14, %KB:14 Wartość bezpośrednia

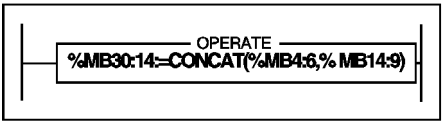
Bit %S18 przyjmuje wartość 1 wtedy, kiedy wartość opisana łańcuchem znaków nie mieści się od -3.402824e+38 do -1.175494e-38 lub +1.175494e-38 do +3.402824e+38 lub gdy jeden z 14 znaków jest niewłaściwy.

2.8-9 Łączenie dwóch łańcuchów

Funkcja ta umożliwia łączenie dwóch łańcuchów znaków zdefiniowanych jako parametry. Wynik zapisywany jest w tablicy bajtów zawierającej łańcuch znaków.

Struktura

Język *Ladder*



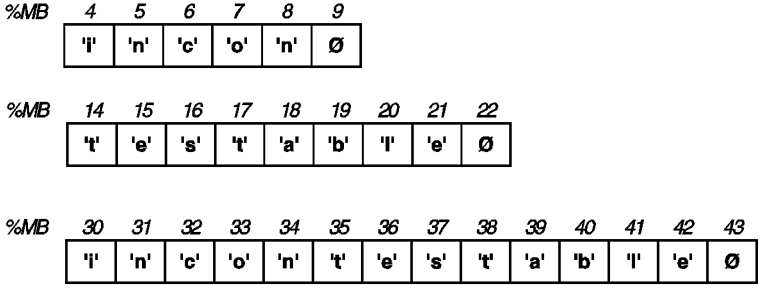
Język *List*

```
LD          TRUE
[ %MB30:14 := CONCAT (%MB4:6, %MB14:9)]
```

Język *ST*

```
%MB30:14 := CONCAT (%MB4:6, %MB14:9) ;
```

Przykład: %MB30:14 := CONCAT (%MB4:6, %MB14:9)



Składnia

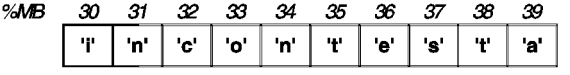
Operator

Wynik :=CONCAT (łańcuch1, łańcuch2)

Argumenty

Typ	Wynik	Łańcuch 1 i 2
Tablice bajtów	%MB:L	%MB:L,%KB:L, Wart. bezp.

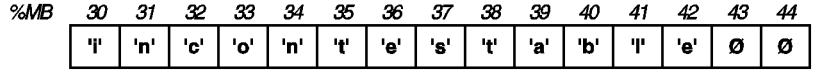
- Jeżeli tablica wynikowa jest za krótka, to wynik jest obcinany, a bit %S15 przyjmuje wartość 1. %MB30:10 := CONCAT (%MB4:6, %MB14:9)



==> %S15 stan 1

- Jeżeli tablica wynikowa jest za długa, to łańcuch uzupełniają znaki końcaø.

%MB30:15 := CONCAT (%MB4:6, %MB14:9)



2.8-10 Usuwanie ciągu znaków z łańcucha

Jest to funkcja umożliwiająca usuwanie zadanej liczby znaków (długość *L*), z zadanego rzędu (pozycja pierwszego znaku do usunięcia) łańcucha traktowanego jako parametr. Wynikiem jest tablica bajtów zawierająca łańcuch znaków.

Struktura

Język *Ladder*

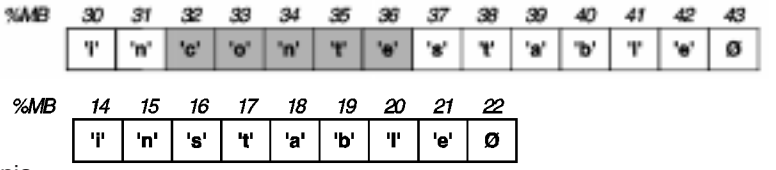
Język *List*

```
LD          TRUE
[%MB14: 9 := DELETE (%MB30:14, %MW2,
%MW4)]
```

Język *ST*

```
%MB14:9 := DELETE (%MB30:14, %MW2, %MW4) ;
```

Przykład: %MB14: 9 := DELETE (%MB30:14, %MW2, %MW4)
gdzie %MW2 = 5 (5 znaków do usunięcia) %MW4 = 3 (pozycja=3)



Składnia

Operator

Wynik :=DELETE (łańcuch, długość, Poz)

Argumenty

Typ	Wynik	Łańcuch	Długość Poz (pozycja)
Tablice bajtów	%MB:L	%MB:L,%KB:L Wartość bezp.	
Słowa indeksowalne			%MW, %KW, %Xi.T
Słowa nieindeks.			%IW,%QW,%SW,%NW Wartość bezpośrednia Wyrażenie numeryczne

Uwagi:

W zależności od użytych indeksów obiektów PL7, niektóre parametry mogą na siebie zachodzić. Mogą to być:

- tablica zawierająca łańcuch źródłowy,
- tablica zawierająca łańcuch docelowy,
- słowo definiujące długość łańcucha przeznaczonego do usunięcia,
- słowo definiujące pozycję pierwszego znaku łańcucha przeznaczonego do usunięcia.

Podanie ujemnej długości lub pozycji traktowane jest jak 0. Wartością początkową parametru opisującego pozycję jest 1 odpowiadająca pozycji pierwszego znaku łańcucha.

Gdy tablica wynikowa jest zbyt długa, to łańcuch uzupełniany jest znakami końca .

Bit systemowy %S15 przyjmuje wartość 1 w następujących przypadkach:

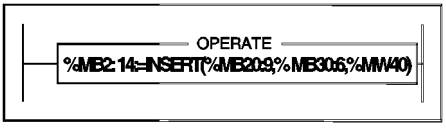
- Długość łańcucha do usunięcia wynosi zero - tablica wynikowa jest kopią źródłowej,
- Zadana pozycja jest większa niż długość łańcucha lub pozycja pierwszego, znalezionego znaku końca łańcucha jest mniejsza lub równa pozycji pierwszego znaku łańcucha przeznaczonego do usunięcia. Wynikiem jest wtedy pusty łańcuch.
- Pozycja równa zeru. Tablica wynikowa zawiera pusty łańcuch znaków.
- Tablica wynikowa jest zbyt krótka. Łańcuch został obcięty.

2.8-11 Wstawianie łańcucha znaków

Jest to funkcja pozwalająca na wstawienie łańcucha znaków zdefiniowanego jako parametr 2 (łańcuch2) do łańcucha zdefiniowanego jako parametr 1 (łańcuch1).
Łańcuch znaków jest wstawiany do pierwszego łańcucha za znakiem znajdującym się na wskazanej pozycji (Parametr *Poz*).
Wynikiem operacji jest nowy łańcuch znaków zapisywany w tablicy bajtów.

Struktura

Język *Ladder*



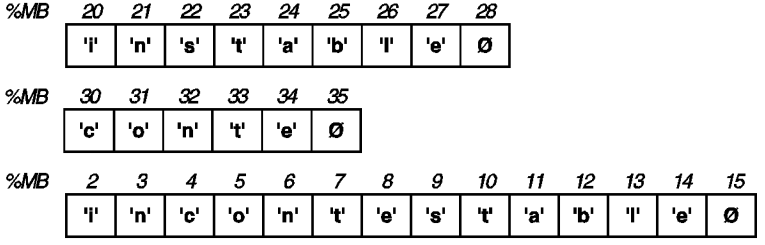
Język *List*

```
LD          TRUE
[%MB2:14 := INSERT (%MB20:9, %MB30:6, %MW40)]
```

Język *ST*

```
%MB2:14 := INSERT (%MB20:9, %MB30:6, %MW40) ;
```

Przykład: %MB2:14 := INSERT (%MB20:9, %MB30:6, %MW40)
gdzie %MW40 := pozycja 2



Składnia

Operator

Wynik :=INSERT (łańcuch1, łańcuch2, Poz)

Argumenty

Typ	Wynik	Łańcuch 1 i 2	Poz (pozycja)
Tablice bajtów	%MB:L Immed. value	%MB:L,%KB:L	
Słowa indeksowalne			%MW, %KW, %Xi.T
Słowa nieindeksowalne			%IW,%QW,%SW,%NW Wartość bezpośrednia Wyrażenia numeryczne

Uwagi:

Najmniejszą wartością parametru określającego pozycję jest 1, co odpowiada pierwszej pozycji w łańcuchu znaków.

Nie można wstawić nowego łańcucha na początek łańcucha źródłowego. Do tego służy funkcja CONCAT.

Jeżeli tablica jest zbyt długa, to należy ją uzupełnić znakami końca łańcucha.

Bit systemowy %S15 przyjmuje wartość 1 w następujących przypadkach:

- Wartość parametru opisującego pozycję jest ujemna lub równa zeru. W takim przypadku, parametr jest traktowany jako równy 0, a tablica wynikowa zawiera pusty łańcuch znaków (składający się z samych znaków końca łańcucha).
- Parametr opisujący pozycję jest większy lub równy długości łańcucha źródłowego. Tablica wynikowa zawiera pusty łańcuch znaków (składający się z samych znaków końca łańcucha).
- Jeżeli tablica jest zbyt krótka, to łańcuch znaków jest obcinany.

2.8-12 Wymiana ciągu znaków w łańcuchu znaków

Jest to funkcja, która umożliwia wymianę części łańcucha źródłowego (łańcuch1) na łańcuch znaków zdefiniowany w tablicy zamiany (łańcuch2). Podmianę łańcucha definiuje się za pomocą dwóch parametrów: pozycji (*Poz*) oraz długości. Parametr opisujący długość dotyczy łańcucha znaków, który ma być usunięty z łańcucha źródłowego, a nie łańcucha, który ma być wpisany do łańcucha źródłowego.

Struktura

Język *Ladder*

Język *List*

```
LD  TRUE
[ %MB2:13 := REPLACE (%MB20:12,
                      %MB30:9, %MW40,
%MW41)]
```

Język *ST*

```
%MB2:13 := REPLACE (%MB20:12, %MB30:9, %MW40, %MW41) ;
```

Przykład: %MB2:13 := REPLACE (%MB20:12, %MB30:9, %MW40, %MW41)
gdzie %MW40 = 3 (długość=3) i %MW41 = 9 (pozycja=9)

	%MB	20	21	22	23	24	25	26	27	28	29	30	31
Łańcuch 1		'm'	'l'	's'	'e'	'_'	'e'	'n'	'_'	'r'	'u'	'n'	Ø

	%MB	30	31	32	33	34	35	36	37	38
Łańcuch 2		's'	't'	'o'	'p'	Ø	'r'	'u'	'n'	Ø

%MB	2	3	4	5	6	7	8	9	10	11	12	13	14
	'm'	'l'	's'	'e'	'_'	'e'	'n'	'_'	's'	't'	'o'	'p'	Ø

Składnia

Operator

```
Wynik := REPLACE (łańcuch1, łańcuch2, długość, Poz)
```

Argumenty

Typ	Wynik	Łańcuch 1 i 2	Długość Poz (pozycja)
Tablica bajtów	%MB:L	%MB:L,%KB:L Wartość bezp.	
Słowa indeksowalne			%MW, %KW,%Xi.T
Słowa nieindeks.			%IW,%QW,%SW,%NW Wartość bezpośrednia Wyrażenie numeryczne

Uwagi:

Najmniejszą wartością parametru określającego pozycję jest 1, co odpowiada pierwszej pozycji w łańcuchu znaków.

Jeżeli tablica jest zbyt długa, to jest uzupełniana znakami końca łańcucha.

Bit systemowy %S15 przyjmuje wartość 1 w następujących przypadkach:

- Wartość parametru opisującego pozycję jest ujemna lub równa zeru. W takim przypadku, parametr jest traktowany jako równy 0, a tablica wynikowa zawiera pusty łańcuch znaków (składający się z samych znaków końca łańcucha).
- Parametr opisujący pozycję jest większy lub równy długości łańcucha źródłowego. Tablica wynikowa zawiera pusty łańcuch znaków (składający się z samych znaków końca łańcucha).
- Jeżeli tablica jest zbyt krótka, to łańcuch znaków jest obcinany.
- Jeżeli pozycja pierwszego znaku końca łańcucha jest mniejsza lub równa pozycji pierwszego znaku, który ma być wymieniony, to tablica wynikowa jest kopią tablicy źródłowej do znaku końca łańcucha uzupełnioną znakami końca łańcucha.

2.8-13 Wydzielanie ciągu znaków z łańcucha znaków

Ta funkcja umożliwia wydzielenie zadanej liczby znaków z łańcucha źródłowego. Pozycję pierwszego znaku łańcucha, który ma być wydzielony opisuje parametr (Pos), natomiast liczbę znaków wydzielanego łańcucha opisuje parametr 'Długość'. Wydzielony łańcuch zapisywany jest w tablicy bajtów (Wynik).

Struktura

Język *Ladder*

Język *List*

```
LD  TRUE
[%MB14: 7 := MID (%MB30:13, %MW2, %MW4)]
```

Język *ST*

```
%MB14:7 := MID (%MB30:13, %MW2, %MW4) ;
```

Przykład: %MB14: 7 := MID (%MB30:13, %MW2, %MW4)
gdzie %MW2 = 4 (długość), %MW4 = 9 (pozycja)

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*
	'm'	'i'	's'	'e'	'_'	'e'	'n'	'_'	's'	't'	'o'	'p'	Ø

==> wynik

%MB	*14*	*15*	*16*	*17*	*18*	*19*	*20*
	's'	't'	'o'	'p'	Ø	Ø	Ø

Składnia

Operator

Wynik :=MID (łańcuch, długość, Poz)

Argumenty

Typ	Wynik	Łańcuch	Długość Poz (pozycja)
Tablica bajtów	%MB:L Wartość bezp.	%MB:L,%KB:L	
Słowa indeksowalne			%MW,%KW,%Xi.T
Słowa nieindeks.			%IW,%QW,%SW,%NW Wartość bezpośrednia Wyrażenie numeryczne

Uwagi:

Najmniejszą wartością parametru określającego pozycję jest 1, co odpowiada pierwszej pozycji w łańcuchu znaków.

Jeżeli tablica jest zbyt długa, to jest uzupełniana znakami końca łańcucha.

Jeżeli zadana długość (parametr) jest większa niż rozmiar łańcucha źródłowego, to tablica wynikowa zawiera cały łańcuch źródłowy.

Jeżeli przed osiągnięciem liczby znaków zdefiniowanych do wydzielenia z łańcucha zostanie wykryty znak końca łańcucha lub ostatni element tablicy, to operacja zatrzymuje się w tym punkcie.

Bit systemowy %S15 przyjmuje wartość 1 w następujących przypadkach:

- Długość łańcucha znaków, który ma być wydzielony z łańcucha źródłowego jest ujemna lub równa zeru. W takim przypadku jest ona interpretowana jako 0, a tablica wynikowa zawiera pusty łańcuch znaków (składający się z samych znaków końca łańcucha).
- Parametr opisujący pozycję jest większy lub równy długości tablicy lub jest większy lub równy pozycji pierwszego znaku końca łańcucha. Tablica wynikowa zawiera pusty łańcuch znaków (składający się z samych znaków końca łańcucha).
- Jeżeli tablica jest zbyt krótka, to łańcuch znaków jest obcinany.

2.8-14 Wydzielanie znaków z łańcucha

Jest to funkcja umożliwiająca wydzielenie określonej liczby znaków poczynając od lewej (LEFT) lub prawej (RIGHT) strony łańcucha źródłowego (parametr *łańcuch*). Liczbę wydzielanych znaków określa parametr 'Długość'.
Wydzielone znaki zapisywane są w tablicy bajtów (Wynik).

Struktura

Język *Ladder*

Język *List*

```
LD  TRUE
[%MB10: 10 := LEFT (%MB30:13, %MW2)]
```

Język *ST*

```
%MB10:10 := LEFT (%MB30:13, MW2) ;
```

Przykład: %MB10: 10 := LEFT (%MB30:13, %MW2)
gdzie %MW2 = 8 (długość)

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*
	'm'	'i'	's'	'e'	'_'	'e'	'n'	'_'	's'	't'	'o'	'p'	Ø

==> wynik

%MB	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*
	'm'	'i'	's'	'e'	'_'	'e'	'n'	'_'	Ø	Ø

Składnia

Operator

Wynik :=LEFT (łańcuch, długość)

Wynik :=RIGHT (łańcuch, długość)

Argumenty

Typ	Wynik	Łańcuch	Długość
Tablica bajtów	%MB:L	%MB:L,%KB:L Wartość bezp.	
Słowa indeksowalne			%MW,%KW,%Xi.T
Słowa nieindeksowalne			%IW,%QW,%SW,%NW Wartość bezpośrednia Wyrażenie numeryczne

Uwagi:

Jeżeli tablica jest zbyt długa, to łańcuch wynikowy jest uzupełniany znakami końca łańcucha.

Jeżeli zadana długość (parametr) jest większa niż rozmiar łańcucha źródłowego, to tablica wynikowa zawiera cały łańcuch źródłowy.

Bit systemowy %S15 przyjmuje wartość 1 w następujących przypadkach:

- Długość łańcucha znaków, który ma być wydzielony z łańcucha źródłowego jest ujemna lub równa zeru. W takim przypadku tablica wynikowa zawiera pusty łańcuch znaków (składający się z samych znaków końca łańcucha).
- Jeżeli tablica jest zbyt krótka, to łańcuch znaków jest obcinany.

2.8-15 Porównywanie łańcuchów znaków

Ta funkcja służy do porównywania dwóch łańcuchów znaków. W wyniku uzyskuje się słowo zawierające pozycję pierwszego znaku różniącego te łańcuchy.
Jeżeli porównywane łańcuchy są takie same, to wynikiem jest -1.

Struktura

Język *Ladder*

Język *List*

```
LD  TRUE
[%MW2 := EQUAL_STR (%MB18:14, %MB50:14)]
```

Język *ST*

```
%MW2 := EQUAL_STR (%MB18:14, %MB50:14) ;
```

Przykład: %MW2 := EQUAL_STR (%MB18:14, %MB50:14) gdzie

%MB	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'p'	'w'	'x'	'y'	'z'

%MB	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	'a'	'b'	'c'	'd'	'?'	'f'	'g'	'h'	Ø	'v'	'w'	'x'	'y'	'z'

==> MW2 := 5

Składnia

Operator

```
Wynik :=EQUAL_STR (łańcuch1, łańcuch2)
```

Argumenty

Typ	Wynik	Łańcuch 1 i 2
Słowa indeksowalne	%MW	
Słowa nieindeksowalne	%QW,%SW,%NW.	
Tablice bajtów		%MB:L,%KB:L Wartość bezpośrednia

Uwaga:
Ujemna wartość długości lub pozycji interpretowana jest jako 0.

Podczas operacji porównywania wychwytywane są różnice pomiędzy dużymi i małymi literami.

2.8-16 Wyszukiwania zadanego ciągu znaków

Ta funkcja służy do wyszukiwania ciągu znaków zdefiniowanych w parametrze 2 w łańcuchu znaków zdefiniowanym jako parametr 1.
W wyniku uzyskuje się słowo zawierające pozycję pierwszego znaku wyszukiwanego łańcucha.
Jeżeli wyszukiwanie kończy się fiaskiem, to wynikiem jest -1.

Struktura

Język *Ladder*

Język *List*

```
LD  TRUE
[ %MW2 := FIND (%MB18:14, %MB50:4)]
```

Język *ST*

```
%MW2 := FIND (%MB18:14, %MB50:4) ;
```

Przykład: %MW2 := FIND (%MB18:14, %MB50:4) gdzie

%MB	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	Ø	'w'	'x'	'y'	'z'

%MB	50	51	52	53
	'f'	'g'	'h'	Ø

==> MW2 := 6 oznacza, że pierwszy znak poszukiwanego łańcucha znaków znajduje się na pozycji 6 łańcucha źródłowego.

Składnia

Operator

```
Wynik :=FIND (łańcuch1, łańcuch2)
```

Argumenty

Typ	Wynik	Łańcuch 1 i 2
Słowa indeksowalne	%MW	
Słowa nieindeksowalne	%QW,%SW,%NW.	
Tablice bajtów		%MB:L,%KB:L Wartość bezpośrednia

Uwaga:

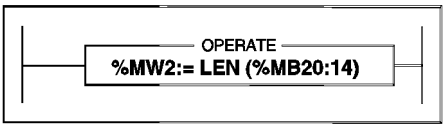
Ujemna wartość długości lub pozycji jest interpretowana jako 0.

2.8-17 Długość łańcucha znaków

Jest to funkcja umożliwiająca obliczenie długości zadanego łańcucha znaków (parametr), czyli liczby znaków znajdujących się przed znakiem końca łańcucha.

Struktura

Język *Ladder*



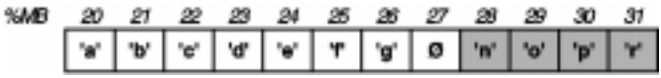
Język *List*

```
LD  TRUE
[%MW2 := LEN (%MB20:14)]
```

Język *ST*

```
%MW2 := LEN (%MB20:14) ;
```

Przykład: %MW2 := LEN (%MB20:14) gdzie



==> %MW2 = 7

Składnia

Operator

```
Wynik := LEN (łańcuch)
```

Argumenty

Typ	Wynik	Łańcuch
Słowa indeksowalne	%MW	
Słowa nieindeksowalne	%QW,%SW,%NW.	
Tablice bajtów		%MB:L,%KB:L Wartość bezpośrednia

Uwaga:

Jeżeli nie zostanie odnaleziony znak zakończenia łańcucha, to wynikiem jest rozmiar tablicy (patrz rozdział 2.8-1).

2.9 Instukcje związane z czasem: Data, Godzina, Czas trwania

2.9-1 Format parametru

Parametry Data *Date*, Pora dnia (godzina) *Time of day* oraz Czas trwania *Duration* mają format zgodny z normą IEC 1131-3.

- Format czasu trwania (typ TIME)

  Ten format służy do wyrażania czasu trwania z dokładnością do dziesiątych części sekundy i odpowiada formatowi standardowemu TIME.

  Wartości wyświetlane są w następującym formacie: ssssssss.d
  czyli na przykład: 3674.3
  oznacza 1 godzinę, 1 minutę, 14 sekund i 3 dziesiąte sekundy.

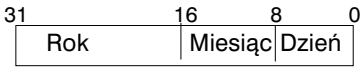
  Wartość jest zapisywana w 32 bitach (słowo podwójnej precyzji), co daje ograniczenie na poziomie [0, 4294967295] dziesiątych części sekundy, co odpowiada mniej więcej 13 latom i 7 miesiącom.

- Format daty (typ DATE)

  Ten format służy do zapisywania roku, miesiąca i dnia. Odpowiada on formatowi standardowemu DATE.

  Wartość wyświetlana jest w następujący sposób: rrrr-mm-dd
  na przykład: 1984-06-25

  Wartość jest zapisywana w BCD w 32 bitach (słowo podwójne) w 3 polach:



Rok : 4 cyfry
Miesiąc : 2 cyfry
Dzień : 2 cyfry

Przykład:

format szesnastkowy

19h	84h	06h	25h

= 1984-06-25

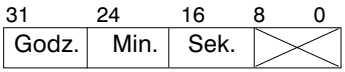
Dozwolone są tylko wartości z przedziału [1990-01-01, 2099-12-31].

• Format pory dnia (typ TOD)

Ten format wykorzystuje się do opisywania pory dnia (godzina, minuty, sekundy). Odpowiada on formatowi standardowemu TIME_OF_DAY.

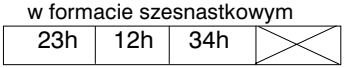
Wartość wyświetlana jest następująco: gg:mm:ss
czyli na przykład: 23:12:34

Wartość kodowana jest w BCD w 32 bitach (słowo podwójne) w 3 polach:



Godzina : 2 cyfry (słowo ważniejsze)
Minuty : 2 cyfry (słowo ważniejsze)
Sekundy : 2 cyfry (słowo mniej ważne)

Przykład:



= 23:12:34

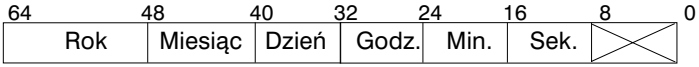
Dozwolone są tylko wartości z przedziału [00:00:00, 23:59:59].

• Format łączny dla daty i pory dnia (typ DT)

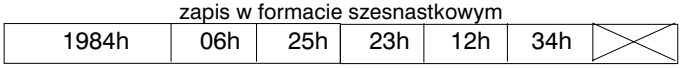
Ten format służy do opisywania daty z podaniem pory dnia (rok, miesiąc, dzień, godzina, minuty, sekundy). Odpowiada to formatowi DATE_AND_TIME.

Wartość jest wyświetlana następująco: rrrr-mm-dd-gg:mm:ss
czyli na przykład: 1984-06-25-23:12:34

Wartość jest zapisywana w kodzie BCD, w 64 bitach (tablica 4 słów):



Przykład:



Dozwolone wartości: [1990-01-01-00:00:00, 2099-12-31- 23:59:59].

• Format: Godzina, Minuty, Sekundy (typ HMS)
Ten format, stosowany wyłącznie dla funkcji TRANS_TIME, służy do kodowania pory dnia (czyli dokładnej godziny).

Wartość jest wyświetlana następująco: gg:mm:ss
czyli na przykład: 23:12:34

Wartość jest zapisana w BCD w 32 bitach (słowo podwójne) w 3 polach:

Godziny : 4 cyfry (słowo ważniejsze)
Minuty : 2 cyfry (słowo mniej ważne)
Sekundy : 2 cyfry (słowo mniej ważne)

Przykład:

zapis w kodzie szesnastkowym

23h	12h	34h

= 23:12:34

2.9-2 Rola słów i bitów systemowych - podsumowanie

Bit systemowy %S17 przyjmuje wartość 1 w następujących okolicznościach:
• Wynik operacji nie mieści się w dopuszczalnym zakresie.
• Parametr wejściowy nie może być zinterpretowany i jest niezgodny z żądanym formatem (DAT, DT lub TOD).
• Operacje na porze dnia (format TOD) prowadzą do zmiany daty.
• Następuje kolizja z zegarem czasu rzeczywistego.

Bit systemowy %S15 przyjmuje wartość 1, gdy łańcuch zapisywany w tablicy przekracza jej długość.

Słowa systemowe:
• %SD18 : bezwzględny licznik czasu, wykorzystywany do obliczania okresu czasu (wartość zwiększana przez system co 1/10 sekundy).
• %SW49 do %SW53 mogą być również wykorzystane do wyświetlania daty (patrz rozdział 3.2-2, część B).

2.9-3 Zegar czasu rzeczywistego

Ta funkcja służy do zarządzania akcjami o określonym lub obliczanym czasie trwania i dacie. Nadaje ona parametrowi wyjściowemu OUT wartość 1 jeżeli data, w momencie wywołania funkcji, mieści się w okresie zdefiniowanym w parametrach wejściowych.

Składnia

Operator

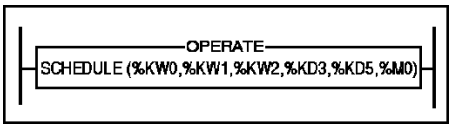
```
SCHEDULE (DBEG, DEND, WEEK, HBEG, HEND, OUT)
```

Charakterystyka parametrów

Wyjście	OUT	Bit zawierający wynik porównania dokonywanego przez zegar czasu rzeczywistego: ma wartość 1 przez okres zdefiniowany w parametrach.
Data początkowa	DBEG	Słowo zawierające datę początkową zadanego okresu (miesiąc-dzień) zakodowanego w BCD (limit: 01-01 do 12-31).
Data końcowa	DEND	Słowo zawierające datę końcową zadanego okresu (miesiąc-dzień) zakodowanego w BCD (limit: 01-01 do 12-31).
Dzień tygodnia	WEEK	Słowo zawierające informację o dniach tygodnia, w których obowiązują warunki zdefiniowane w parametrach DBEG i DEND. 7 mniej znaczących bitów słowa reprezentuje 7 dni tygodnia: bit 6 = Poniedziałek, bit 5 = Wtorek,..., bit 0 = Niedziela.
Godzina początkowa	HBEG	Słowo podwójne zawierające informację o godzinie początkowej zadanego okresu (godziny- minuty- sekundy) w BCD zapisane w formacie pory dnia (typ: TOD). Limit: 00:00:00, 23:59:59.
Godzina końcowa	HEND	Słowo podwójne zawierające informację o godzinie końcowej zadanego okresu (godziny- minuty- sekundy) w BCD zapisane w formacie pory dnia (typ: TOD). Limit: 00:00:00, 23:59:59.

Struktura

Język *Ladder*



Język *List*

```
LD TRUE
[SCHEDULE (%KW0,%KW1,%KW2,%KD3,
 %KD5,%Q0.0)]
```

Język *ST*

```
SCHEDULE    ( 16#0501,                  (* data poczatkowa : 1 Maja*)
              16#1031,                  (* data koncowa : 31 Pazdziernika*)
              2#0000000001111100,       (* Od Poniedzialku do Piatku*)
              16#08300000,              (* poczatek : 08:30*)
              16#18000000,              (* koniec : 18:00*)
              %Q0.0) ;                  (* wynik w slowie : %Q0.0*)
```

Argumenty

SCHEDULE (DBEG, DEND, WEEK, HBEG, HEND, OUT)

Typ	DBEG,DEND,WEEK	HBEG,HEND	OUT
Słowa indeksowalne	%MW,%KW,%Xi.T		
Słowa nieindeksowalne	%IW,%QW,%SW,%NW Wartość bezpośrednia Wyrażenie numeryczne		
Indeks. słowa podwójne		%MD,%KD	
Nieindeksowalne słowa podwójne		%ID,%QD Wartość bezp. Wyr. numeryczne	
Bity			%I,%Q, %M, %S, %BLK,%•:Xk, %X

Uwagi:

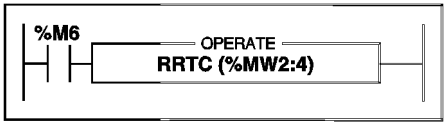
- Parametry DBEG i DEND określają zadany okres w kontekście dni roku. Ten okres może rozciągać się na dwa lata kalendarzowe, np. od 10 października do 7 kwietnia. Dzień 29 lutego może być wykorzystany przy definiowaniu tego okresu, ale będzie uwzględniany tylko w roku przestępnym.
- Parametry HBEG i HEND definiują zadany okres w czasie dnia. Okres ten może obejmować dwa dni, np. od 22:00 do 06:10:20.
- Jeżeli jedna z dat DBEG lub DEND lub jeden z czasów HBEG lub HEND będzie niewłaściwa, tj. nie będzie mieściła się w zdefiniowanym zakresie, to na wyjściu OUT pojawi się 0, a bit %S17 przyjmie wartość 1.
- Jeżeli programowany sterownik nie jest wyposażony w wewnętrzny zegar czasu rzeczywistego (jak np. TSX37-10), to na wyjściu będzie 0, a bit systemowy %S17 będzie miał stan 1.
- Jeżeli duża dokładność nie jest konieczna, to można sterować wywołaniem funkcji SCHEDULE za pomocą bitu systemowego %S6 lub %S7 (odciąża to procesor sterownika).

2.9-4 Odczytywanie daty systemowej

Funkcja ta powoduje odczytanie daty systemowej (zegar czasu rzeczywistego) i zapisanie jej w formacie DT we wskazanym obiekcie (parametr).

Struktura

Język *Ladder*



Język *List*

```
LD      %M6
[RRTC (%MW2:4)]
```

Język *ST*

```
IF %M6 THEN
    RRTC (%MW2:4) ;
END_IF ;
```

Przykład: RRTC (%MW2:4)
Wynik jest zapisywany w tablicy słów wewnętrznych o długości 4 słów: %MW2 do %MW5.

Składnia
Operator

RRTC(data)

Argument

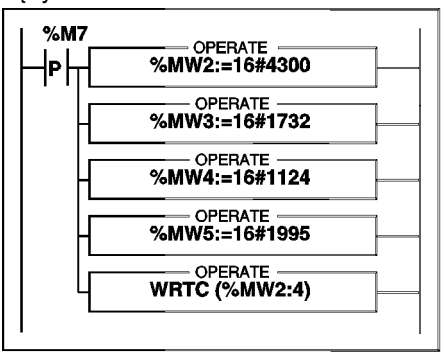
Typ	Data
Tablica - 4 słowa w formacie DT	%MW:4

2.9-5 Aktualizowanie daty systemowej

Funkcja ta umożliwia uaktualnienie daty systemowej (zegar czasu rzeczywistego) i powoduje zapisanie jej w formacie DT we wskazanym obiekcie (parametr).

Struktura

Język *Ladder*



Język *List*

```
LDR     %M7
[%MW2:= 16#4300]
[%MW3:= 16#1732]
[%MW4:= 16#1124]
[%MW5:= 16#1995]
[WRTC (%MW2:4)]
```

Język *ST*

```
IF RE %M7 THEN
    %MW2 := 16#4300 ;
    %MW3 := 16#1732 ;
    %MW4 := 16#1124 ;
    %MW5 := 16#1995 ;
    WRTC (%MW2:4) ;
END_IF ;
```

Przykład: w tablicy słów wewnętrznych %MW2:4 o długości 4 słów zapisywana jest nowa data, po czym jest ona wysyłana do systemu za pomocą funkcji WRTC.

Składnia

Operator WRTC(data)

Argument

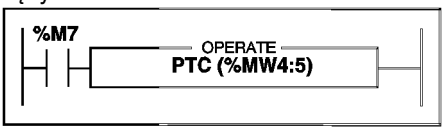
Typ	Data
Tablica o długości 4 słów w formacie DT	%MW:4, %KW:4

2.9-6 Odczytanie daty i kodu zatrzymania sterownika

Jest to funkcja umożliwiająca odczytanie daty ostatniego zatrzymania sterownika i kodu zawierającego informację o przyczynie zatrzymania (w piątym słowie, równoważnym słowu %SW58). Patrz rozdział 3.2-2, część B.

Struktura

Język *Ladder*



Język *List*

```
LD      %M7
[PTC (%MW4:5)]
```

Język *ST*

```
IF %M7 THEN
    PTC (%MW4:5) ;
END_IF ;
```

Przykład: PTC (%MW4:5)
Wynik jest zapisywany w tablicy słów wewnętrznych (długość 5 słów): %MW4 ÷ %MW8.

Składnia

Operator PTC (data)

Argument

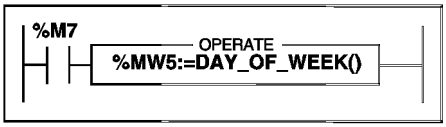
Typ	Data
Tablica o długości 5 słów w formacie DT	%MW:5

2.9-7 Odczytywanie dnia tygodnia

Funkcja ta powoduje wyświetlenie dnia tygodnia w formie cyfry od 1 do 7 zapisywanej w zadanym słowie (1 = Poniedziałek, 2 = Wtorek, 3 = Środa, 4 = Czwartek, 5 = Piątek, 6 = Sobota, 7 = Niedziela).

Struktura

Język *Ladder*



Język *List*

```
LD      %M7
[%MW5 := DAY_OF_WEEK()]
```

Język *ST*

```
IF %M7 THEN
    %MW5 := DAY_OF_WEEK () ;
END_IF ;
```

Przykład: %MW5 := DAY_OF_WEEK() %MW5 := 4 co odpowiada Czwartkowi

Składnia

Operator

```
Wynik :=DAY_OF_WEEK()
```

Argument

Typ	Wynik
Słowa indeksowalne	%MW
Słowa nieindeksowalne	%QW, %SW, %NW

Uwaga

Jeżeli, ze względu na błąd dostępu do zegara czasu rzeczywistego, funkcja nie może ustalić wyniku, to wystawia 0, a bit systemowy %S17 przyjmuje wartość 1.

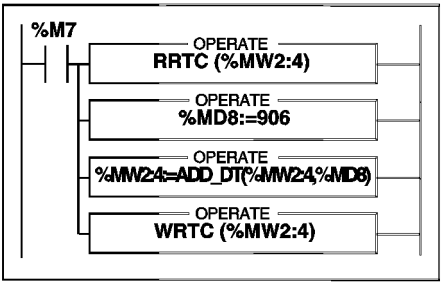
2.9-8 Dodawanie (odejmowanie) okresu czasu do daty

Powoduje dodanie lub usunięcie dodatkowego czasu *In2* (wyrażonego w dziesiątych częściach sekundy) do daty *In1*. W wyniku uzyskuje się nowy czas zapisywany w tablicy składającej się z 4 słów.

ADD_DT () = dodanie okresu czasu SUB_DT () = odjęcie okresu czasu

Struktura

Język *Ladder*



Język *List*

```
LD          %M7
[RRTC (%MW2:4)]
[%MD8 := 906]
[%MW2:4:= ADD_DT(%MW2:4, %MD8)]
[WRTC (%MW2:4)]
```

Język *ST*

```
IF %M7 THEN
    RRTC (%MW2:4) ;
    %MD8 := 906 ;
    %MW2:4 := ADD_DT (%MW2:4, %MD8) ;
    WRTC (%MW2:4) ;
END_IF ;
```

Przykład: %MW2:4 := ADD_DT(%MW2:4, %MD8)
%MW2:4 := data źródłowa
%MD8 := 906 (906 dziesiątych części sekundy zaokrąglone do 1min. 31s)
%MW2:4 := nowa data

Składnia

Operatory

Wynik :=ADD_DT (In1, In2)
Wynik :=SUB_DT (In1, In2)

Argumenty

Typ	Wynik	In1 (data źródłowa)	In2 (okres czasu)
Tablica o długości 4 słów w formacie DT	%MB:4	%MW4:4, %KW:4	
Indeks. słowa podwójne			%MD,%KD
Nieindeks. słowa podwójne			%ID,%QD Wartość bezp. Wyrażenie num.

Uwagi:

- Parametr definiujący dodawany okres czasu (wyrażony w 1/10 sekundy) jest zaokrąglany w górę lub w dół (z dokładnością do 1 sekundy).
  - ssssssss.0 do sssssssss.4 jest zaokrąglany do ssssssss.0
  - ssssssss.5 to sssssssss.9 jest zaokrąglany do ssssssss.0 + 1.0
- Należy zastosować odpowiednie środki jeżeli aplikacja ma uwzględniać lata przestępne.
- Jeżeli wynik operacji wykracza poza zadane ograniczenia, to bit systemowy %S17 przyjmuje wartość 1, a wynik przyjmuje wartość minimalną (dla SUB_DT) lub zostaje zablokowany na wartości maksymalnej (dla ADD_DT).
- Jeżeli "data źródłowa" nie może być zinterpretowana ponieważ nie jest zgodna z formatem DT (DATE_AND_TIME), to bit systemowy %S17 przyjmuje wartość 1, a w wyniku uzyskuje się wartość 0001-01-01-000:00:00.

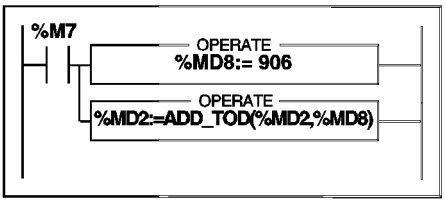
2.9-9 Dodawanie okresu czasu do pory dnia

Funkcje te pozwalają na dodanie (lub odjęcie) zadanego okresu czasu do pory dnia. W wyniku uzyskuje się nowy czas, który jest zapisywany w słowie podwójnym.

ADD_TOD () = dodanie okresu czasu SUB_TOD () = odjęcie okresu czasu

Struktura

Język *Ladder*



Język *List*

```
LD          %M7
[%MD8 := 906]
[%MD2 := ADD_TOD (%MD2, %MD8)]
```

Język *ST*

```
IF %M7 THEN
    %MD8 := 906 ;
    %MD2 := ADD_TOD (%MD2, %MD8) ;
END_IF ;
```

Przykład: %MD2 := ADD_TOD (%MD2, %MD8)
%MD2 := czas źródłowy (np. 12:30:00)
%MD8 := 906 (906 dziesiątych części sekundy zaokrąglone do 1min. 31s)
%MD2 := nowy czas (np. 12:31:31)

Składnia

Operatory

Wynik :=ADD_TOD (In1, In2)
Wynik :=SUB_TOD (In1, In2)

Argumenty

Typ	Wynik	In1 (czas źródłowy) i In2 (okres czasu)
Indeks. słowa podwójne	%MD	%MD,%KD
Nieindeksowalne słowa podwójne	%QD	%ID,%QD Wartość bezpośrednia, Wyr. numeryczne

Wynik oraz In1 są w formacie TOD, natomiast In2 jest w formacie TIME.

Uwagi:

- Parametr definiujący czas dodawany (wyrażony w 1/10 sekundy) jest zaokrąglanyw górę lub w dół (z dokładnością do 1 sekundy).
  - ssssssss.0 do ssssssss.4 jest zaokrąglany do ssssssss.0
  - ssssssss.5 do ssssssss.9 jest zaokrąglany do ssssssss.0 + 1.0
- Jeżeli wynik operacji wykracza poza zadane ograniczenie, to zmienia się dzień. W takim przypadku bit systemowy %S17 przyjmuje wartość 1, a wartość wynikowa może być traktowana jako reszta po godzinie 24:00:00.
- Jeżeli pora dnia (parametr *In1*) nie może być prawidłowo odczytana we formacie TOD, to bit systemowy %S17 przyjmuje wartość 1, a w wyniku uzyskuje się godzinę 00:00:00.

2.9-10 Różnica między dwiema datami (bez czasu)

Ta funkcja umożliwia obliczania ilości dni różniących od siebie dwie daty. Wynik podawany jest jako wartość bezwzględna i jest zapisywana w słowie podwójnym.

Struktura

Język *Ladder*

Język *List*

```
LD          %M7
[%MD10 := DELTA_D (%MD2, %MD4)]
```

Język *ST*

```
IF %M7 THEN
    %MD10 := DELTA_D (%MD2, %MD4) ;
END_IF ;
```

Przykład: %MD10 := DELTA_D (%MD2, %MD4)
%MD2 := data numer1 (np. 1994-05-01)
%MD4 := data numer2 (np. 1994-04-05)
==> %MD10 = 22464000 (==> różnica = 26 dni)

SKładnia

Operator

```
Wynik :=DELTA_D(Data1,Data2)
```

Argumenty

Typ	Wynik	Data 1 i 2
Indeks. słowa podwójne	%MD	%MD,%KD
Nieindeksowalne słowa podwójne	%QD	%ID,%QD Wartość bezpośrednia, Wyr. num.

Wynik ma format TIME, natomiast Data 1 i 2 mają format DATE.
W formacie TIME wartość jest definiowana z dokładnością do 1/10 sekundy. Natomiast wartość w formacie DATE jest definiowana z dokładnością do 1 dnia. Obliczona więc różnica czasu będzie wielokrotnością 864000 (= 1dzień = 24 h x 60 min x 60 s x 10 dziesiątych części sekundy).

Uwaga

- Jeżeli wynik przekracza wartość maksymalną dla okresu czasu (format TIME), to następuje przepełnienie. Wynikiem jest wtedy 0, a bit %S18 ma stan 1.
- Jeżeli jeden z parametrów wejściowych nie może być prawidłowo zinterpretowany lub jest niezgodny z formatem DATE, to bit systemowy %S17 przyjmuje wartość 1, a wynikiem jest 0.

2.9-11 Różnica pomiędzy dwiema datami (z czasem)

Jest to funkcja umożliwiająca obliczenie różnicy między dwiema datami z uwzględnieniem pory dnia. Wynikiem jest wartośc bezwzględna zapisywana w słowie podwójnym.

Struktura

Język *Ladder*

Język *List*

```
LD TRUE
[%MD10 := DELTA_DT (%MW2:4,
%MW6:4)]
```

Język *ST*

```
%MD10 := DELTA_DT (%MW2:4, %MW6:4) ;
```

Przykład: %MD10 := DELTA_DT (%MW2:4, %MW6:4)
%MW2:4 := data numer1 (np. 1994-05-01-12:00:00)
%MW6:4 := date numer2 (np. 1994-05-01-12:01:30)
==> %MD10 = 900 (==> różnica = 1 minuta i 30 sekund)

Składnia

Operator

```
Wynik :=DELTA_DT(Data1,Data2)
```

Argumenty

Typ	Wynik	Data 1 i 2
Indeks. słowa podwójne	%MD,	
Nieindeksowalne słowa podwójne	%QD	
Tablica o długości 4 słów w formacie DT		%MW:4, %KW:4

Wynik ma format TIME natomiast Data 1 i 2 - format DT.

Wartość zapisywana jest w formacie TIME z dokładnością do 1/10 sekundy. Natomiast wartości w formacie DT definiuje się z dokładnością do 1 sekundy. Stąd też, obliczona różnica czasu będzie wielokrotnością 10.

Uwaga

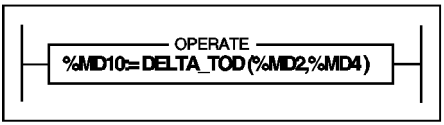
- Jeżeli wynik przekracza wartość maksymalną dla okresu czasu (format TIME), to następuje przepełnienie. Wynikiem jest wtedy 0, a bit %S18 ma stan 1.
- Jeżeli jeden z parametrów wejściowych nie może być prawidłowo zinterpretowany lub jest niezgodny z formatem DATE, to bit systemowy %S17 przyjmuje wartość 1, a wynikiem jest 0.

2.9-12 Różnica pomiędzy dwoma czasami

Ta funkcja umożliwia obliczanie różnicy pomiędzy dwoma czasami określającymi porę dnia. Wynik jest wartością bezwzględną zapisywaną w słowie podwójnym.

Struktura

Język *Ladder*



Język *List*

```
LD TRUE
[%MD10 := DELTA_TOD (%MD2, %MD4)]
```

Język *ST*

```
%MD10 := DELTA_TOD (%MD2, %MD4) ;
```

Przykład: %MD10 := DELTA_TOD (%MD2, %MD4)
%MD2 := czas1 (np. 02:30:00)
%MD4 := czas2 (np. 02:40:00)
==> %MD10 = 6600 (==> różnica = 11 minut)

Składnia

Operator

```
Wynik :=DELTA_TOD(Czas1,Czas2)
```

Argumenty

Typ	Wynik	Czas 1 i 2
Indeks. słowa podwójne	%MD	%MD,%KD
Nieindeksowalne słowa podwójne	%QD	%ID,%QD Wartość bezp., Wyr. numeryczne

Wynik ma format TIME natomiast Czas 1 i 2 mają format TOD.

Wartość zapisywana jest w formacie TIME z dokładnością do 1/10 sekundy. Natomiast wartości w formacie TOD definiuje się z dokładnością do 1 sekundy. Stąd też obliczona różnica czasu będzie wielokrotnością 10.

Uwaga

Jeżeli jeden z parametrów wejściowych nie może być prawidłowo zinterpretowany i jest niezgodny z formatem TOD, to bit systemowy %S17 przyjmuje wartość 1, a wynikiem jest 0.

2.9-13 Konwersja daty na łańcuch znaków

Ta instrukcja powoduje przeliczenie daty (bez czasu) na łańcuch znaków o formacie RRRR-MM-DD (10 znaków). Łańcuch kończy znak końca łańcucha . Każdemu znakowi R,M,D odpowiada liczba.

Struktura

Język *Ladder*

Język *List*

```
LD              TRUE
[%MB2:11 := DATE_TO_STRING (%MD40)]
```

Język *ST*

```
%MB2:11 := DATE_TO_STRING (%MD40) ;
```

Przykład: %MB2:11 := DATE_TO_STRING (%MD40)
%MD40 := DATE (np. 1998-12-27)

==:

%MB	2	3	4	5	6	7	8	9	10	11	12
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'	'2'	'7'	Ø

Składnia

Operator

Wynik :=DATE_TO_STRING(Data)

Argumenty

Typ	Wynik	Data
Tablice o długości 11 bajtów	%MB:11	
Indeks. słowa podwójne		%MD,%KD
Nieindeksowalne słowa podwójne		%ID,%QD Wartość bezp., Wyrażenie num.

Uwagi: Jeżeli parametr wejściowy (data) nie może być właściwie zinterpretowana lub jest niezgodna z formatem DATE, to bit systemowy %S17 przyjmuje wartość 1, a funkcja zwraca następujący łańcuch: ' **** - ** - ** '. Jeżeli docelowy łańcuch jest za krótki, to następuje obcięcie, a bit systemowy %S15 przyjmuje wartość 1.

%MB2:8 := DATE_TO_STRING (%MD40)

==>

%MB	2	3	4	5	6	7	8	9
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'

==> %S15 = 1

Jeżeli łańcuch docelowy jest zbyt długi, to jest on uzupełniany znakami końca łańcucha .

%MB2:12 := DATE_TO_STRING (%MD40)

==>

%MB	2	3	4	5	6	7	8	9	10	11	12	13
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'	'2'	'7'	Ø	Ø

2.9-14 Konwersja pełnej daty na łańcuch znaków

Ta funkcja umożliwia konwersję daty (wraz z czasem) na łańcuch znaków o następującym formacie: RRRR-MM-DD-GG:MM:SS (19 znaków). Łańcuch kończy znak końca łańcucha . Każdemu znakowi R,M,D,G,M,S odpowiada liczba.

Struktura

Język *Ladder*

Język *List*

```
LD TRUE
[%MB2:20 := DT_TO_STRING (%MW50:4)]
```

Język *ST*

```
%MB2:20 := DT_TO_STRING (%MW50:4) ;
```

Przykład: %MB2:20 := DT_TO_STRING (%MW50:4)
%MW50:4 := Data i czas (typ DT) (np. 1998-12-27-23:14:37)

%MB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'	'2'	'7'	'-'	'2'	'3'	':'	'1'	'4'	':'	'3 '	'7'	Ø

Składnia

Operator

Wynik :=DT_TO_STRING(Data)

Argumenty

Typ	Wynik	Data
Tablica 20-bajtowa	%MB:20	
Tablica o dł. 4 słów w formacie DT		%MW:4, %KW:4

Uwaga: Jeżeli parametr wejściowy (data) nie może być poprawnie zinterpretowany lub jest niezgodny z formatem DT (DATE_AND_TIME), to bit systemowy %S17 przyjmuje wartość 1, a w wyniku otrzymuje się łańcuch znaków ' **** - ** -** - ** : ** : **'. Jeżeli łańcuch wynikowy jest za krótki, to następuje obcięcie jego treści, a bit systemowy %S15 przyjmuje wartość 1.

%MB2:8 := DT_TO_STRING (%MW50:4)

==>

%MB	2	3	4	5	6	7	8	9
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'

==> %S15 = 1

- Jeżeli łańcuch wynikowy jest zbyt długi, to jest on uzupełniany znakami końca łańcucha .

%MB2:21 := DT_TO_STRING (%MW50:4)

==>

%MB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'	'2'	'7'	'-'	'2'	'3'	':'	'1'	'4'	':'	'3 '	'7'	Ø	Ø

2.9-15 Konwersja okresu czasu na łańcuch znaków

Ta instrukcja powoduje przeliczenie okresu czasu (format TIME) na łańcuch znaków. Wynik może być wyświetlany z rozbiciem na godziny, minuty, sekundy i dziesiąte części sekundy (15 znaków): GGGGGG:MM:SS.D. Łańcuch znaków kończy znak końca łańcucha . Każdemu znakowi G,M,S,D odpowiada liczba.

Maksymalny okres czasu, który można poddać konwersji wynosi 119304 godziny, 38 minut, 49 sekund i 5 dziesiątych sekundy.

Struktura

Język *Ladder*

Język *List*

```
LD      TRUE
[%MB2:15 := TIME_TO_STRING (%MD40)]
```

Język *ST*

```
%MB2:15 := TIME_TO_STRING (%MD40) ;
```

Przykład: %MB2:15 := TIME_TO_STRING (%MD40)
gdzie %MD40 := 27556330.3 (format TIME)

%MB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	'0'	'0'	'7'	'6'	'5'	'4'	':'	'3'	'2'	':'	'1'	'0'	'.'	'3'	Ø

Składnia

Operator

```
Wynik :=TIME_TO_STRING(okres czasu)
```

Argumenty

Typ	Wynik	Okres czasu
Tablica 15-bajtowa	%MB:15	
Indeks. słowa podwójne		%MD,%KD
Nieindeksowalne słowa podwójne		%ID,%QD Wartość bezp., Wyr. numeryczne

Okres czasu: w formacie TIME.

Uwaga:
Jeżeli łańcuch wynikowy jest zbyt krótki, to jego treść jest obcinana, a bit %S15 ma stan 1.

%MB2:8 := TIME_TO_STRING (%MD40)

==>%MB

2	3	4	5	6	7	8	9
'0'	'0'	'7'	'6'	'5'	'4'	':'	'3'

==> %S15 = 1

Jeżeli łańcuch wynikowy jest zbyt długi to jest on uzupełniany znakami końca łańcucha .

%MB2:16 := TIME_TO_STRING (%MD40)

==>

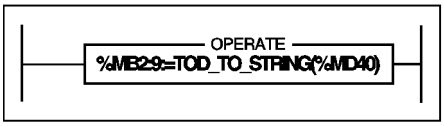
%MB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	'0'	'0'	'7'	'6'	'5'	'4'	':'	'3'	'2'	':'	'1'	'0'	'.'	'3'	Ø	Ø

2.9-16 Konwersja pory dnia na łańcuch znaków

Ta instrukcja umożliwia przeliczanie pory dnia (format TOD - TIME_OF_DAY) na łańcuch znaków o formacie GG:MM:SS składającego się z 8 znaków plus znak końca łańcucha . Każdemu znakowi G,M,S odpowiada liczba.

Struktura

Język *Ladder*



Język *List*

```
LD      TRUE
[%MB2:9 := TOD_TO_STRING (%MD40)]
```

Język *ST*

```
%MB2:9 := TOD_TO_STRING (%MD40) ;
```

Przykład: %MB2:9 := TOD_TO_STRING (%MD40)
gdzie %MD40 := 23:12:27 (format TOD)

==>

%MB	2	3	4	5	6	7	8	9	10
	'2'	'3'	':'	'1'	'2'	':'	'2'	'7'	Ø

Składnia

Operator

Wynik :=TOD_TO_STRING(czas)

Argumenty

Typ	Wynik	Pora dnia (czas)
Tablica 9-bajtowa	%MB:9	
Indeks. słowa podwójne		%MD,%KD
Nieindeksowalne słowa podwójne		%ID,%QD Wartość bezp., Wyrażenie num.

Czas: format TOD.

Uwaga:
Jeżeli łańcuch wynikowy jest zbyt krótki, to jego treść jest obcinana, a bit %S15 ma stan 1.

%MB2:8 := TOD_TO_STRING (%MD40) (gdzie %MD40 := 23:12:27)

==>%MB

2	3	4	5	6	7	8	9
'2'	'3'	':'	'1'	'2'	':'	'2'	'7'

==> %S15 = 1

Jeżeli łańcuch wynikowy jest zbyt długi, to jest on uzupełniany znakami końca łańcucha .

%MB2:10 := TOD_TO_STRING (%MD40) (gdzie %MD40 := 23:12:27)

==> %MB

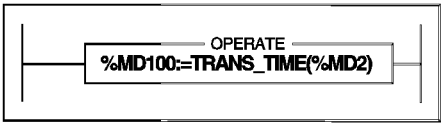
2	3	4	5	6	7	8	9	10	11
'2'	'3'	':'	'1'	'2'	':'	'2'	'7'	Ø	Ø

2.9-17 Konwersja okresu czasu na format GGGG:MM:SS

Ta instrukcja powoduje przeliczenie okresu czasu (format TIME) na format godziny-minuty-sekundy (GGGG:MM:SS). Ograniczenie: [0000:00:00, 9999:59:59].

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[%MD100 := TRANS_TIME (%MD2)]
```

Język *ST*

```
%MD100 := TRANS_TIME (%MD2) ;
```

Przykład: %MD100 := TRANS_TIME (%MD2)
gdzie %MD2 := 86324873 dziesiątych części sekundy

==> MD2

31	16	8	0
2 3 9 7	54	47	

wartość wyrażona w formacie szesnastkowym

Składnia

Operator

```
Wynik :=TRANS_TIME(Okres czasu)
```

Argumenty

Typ	Wynik	Okres czasu
Indeks. słowa podwójne	%MD	%MD,%KD
Nieindeksowalne słowa podwójne	%QD	%ID,%QD Wartość bezp., Wyrażenie num.

Wynik: format HMS.
Okres czasu: format TIME.

Uwagi:
Parametr "okres czasu" (wyrażany w 1/10 sekundy) jest, w celu umożliwienia dokonania konwersji, zaokrąglany do góry lub do dołu (z dokładnością do 1 sekundy).

- sssssssss.0 do sssssssss.4 jest zaokrąglany do sssssssss.0
- sssssssss.5 do sssssssss.9 jest zaokrąglany do sssssssss.0 + 1.0

Maksymalny okres czasu, który można poddać konwersji wynosi 10000 godzin. Oznacza to, że jeżeli wartość okresu czasu (TIME) jest większa lub równa 360000000, to nie można jej poddać konwersji. Bit systemowy %S15 przyjmuje wtedy wartość 1, a wynikiem jest 0000:00:00.

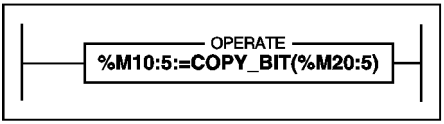
2.10 Operacje na tablicach bitowych

2.10-1 Kopiowanie tablicy bitowej do tablicy bitowej

Funkcja ta umożliwia kopiowanie, bit po bicie, tablicy bitowej do innej tablicy bitowej.

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[%M10:5 := COPY_BIT (%M20:5)]
```

Język *ST*

```
%M10:5 := COPY_BIT (%M20:5) ;
```

Składnia

Operator

```
Wynik :=COPY_BIT (Tab)
```

Argumenty

Typ	Wynik	Tablica (tab)
Tablica bitowa	%M:L, %Q:L, %I:L	%M:L, %Q:L, %I:L, %Xi:L

Uwagi:

- Tablice mogą być różnych rozmiarów. Wtedy tablica wynikowa zawiera wynik działania funkcji na obszarze odpowiadającym najmniejszemu rozmiarowi tablicy, a pozostała część nie jest modyfikowana.
- Należy zachować ostrożność by nie nastąpiło nadpisanie pomiędzy tablicą źródłową a wynikową.

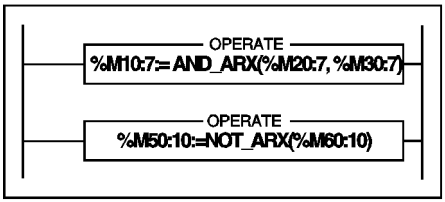
2.10-2 Operacje logiczne na tablicach bitowych

Istnieje możliwość wykonywania operacji logicznych na dwóch tablicach (bit po bicie), przy czym wynik zapisywany jest w tablicy trzeciej.

- AND_ARX : koniunkcja AND (bit po bicie),
- OR_ARX : alternatywa OR (bit po bicie),
- XOR_ARX : nierównoważność (bit po bicie),
- NOT_ARX : logiczne dopełnienie (bit po bicie) tablicy.

Struktura

Język *Ladder*



Język *List*

```
LD          TRUE
[%M10:7 := AND_ARX (%M20:7,
%M30:7)]

LD          TRUE
[%M50:10 := NOT_ARX (%M60:10)]
```

Język *ST*

```
%M10:7 := AND_ARX (%M20:7, %M30:7) ;
%M50:10 := NOT_ARX (%M60:10) ;
```

Składnia

Operator

```
Wynik := AND_ARX (Tab 1, Tab 2)
Wynik := OR_ARX (Tab 1, Tab 2)
Wynik := XOR_ARX (Tab 1, Tab 2)
Wynik := NOT_ARX (Tab 1)
```

Argumenty

Typ	Wynik	Tablica 1 i 2 (tab)
Tablica bitowa	%M:L, %Q:L, %I:L	%M:L, %Q:L, %I:L, %Xi:L

Uwagi:

- Tablice mogą być różnych rozmiarów. Wtedy tablica wynikowa zawiera wynik działania funkcji na obszarze odpowiadającym najmniejszemu rozmiarowi tablicy, a pozostała część nie jest modyfikowana.
- Należy zachować ostrożność by nie nastąpiło nadpisanie pomiędzy tablicą źródłową a wynikową.

2.10-3 Kopiowanie tablicy bitowej do tablicy słów

Ta funkcja pozwala kopiować bity tablicy bitowej (lub jej części) do tablicy słów (lub tablicy słów podwójnych).

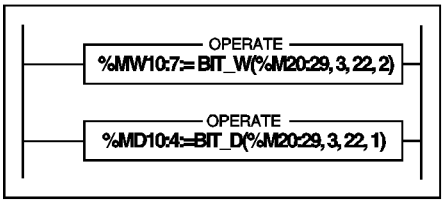
Operacja kopiowania tablicy bitów obejmuje określoną liczbę bitów (parametr *nbit*) licząc od podanego rzędu (parametr *brow*).

Kopiowane bity są zapisywane do tablicy słów (lub tablicy słów podwójnych) poczynając od rzędu (parametr *wrow* lub *drow*) rozpoczynającego się od najmniej znaczącego bitu każdego słowa.

- BIT_W : powoduje kopiowanie tablicy bitowej do tablicy słów.
- BIT_D : powoduje kopiowanie tablicy bitowej do tablicy słów podwójnych.

Struktura

Język *Ladder*



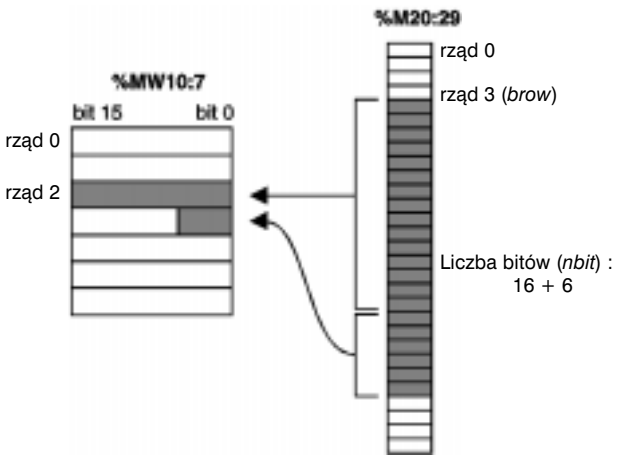
Język *List*

```
LD          TRUE
[%MW10:7 := BIT_W (%M20:29, 3, 22, 2)]

LD          TRUE
[%MD10:4 := BIT_D (%M20:29, 3, 22, 1)]
```

Język *ST*

```
%MW10:7 := BIT_W (%M20:29, 3, 22, 2) ;
%MD10:4 := BIT_D (%M20:29, 3, 22, 1)  ;
```

Przykład: %MW10:7 := BIT_W (%M20:29, 3, 22, 2) ;



Składnia

Operator

```
Wynik := BIT_W (Tab, brow, nbit, wrow)
Wynik := BIT_D (Tab, brow, nbit, drow)
```

Argumenty

Typ	Wynik	Tablica (tab)	brow - nbit wrow lub drow
Tablice słów	%MW:L		
Tablice słów podwójnych	%MD:L		
Tablice bitowe		%M:L, %Q:L, %I:L, %Xi.L	
Słowa indeksowalne			%MW, %KW, %Xi.T
Słowa nieindeksowalne			%IW, %QW, %SW, %NW, Wartość bezp. Wyr. numeryczne

Uwagi:

- Jeżeli zadana liczba bitów do skopiowania jest większa niż liczba bitów w tablicy, licząc od miejsca rozpoczęcia kopiowania (rząd *brow*) do końca tablicy, to kopiowanie jest wykonywane do ostatniego elementu tablicy.
- Jeżeli zadana liczba bitów do skopiowania przekracza liczbę bitów w słowach, stanowiących słowa pozostałe w tablicy wynikowej, to funkcja przerywa operację kopiowania po osiągnięciu ostatniego elementu w tablicy słów (albo tablicy słów podwójnych).
- Ujemna wartość parametru *brow*, *nbit*, *wrow* lub *drow* traktowana jest jak zero.

2.10-4 Kopiowanie elementów tablicy słów do tablicy bitowej

Ta funkcja powoduje kopiowanie wszystkich bitów (lub części bitów) tablicy słów (lub tablicy słów podwójnych) do tablicy bitowej.

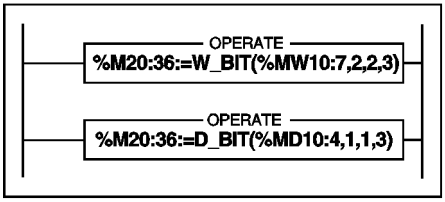
Operacja kopiowania z tablicy słów (lub tablicy słów podwójnych) obejmuje określoną liczbę słów (parametr *nwd*) licząc od podanego rzędu (parametr *wrow* lub *drow*).

Kopiowane bity są zapisywane do tablicy bitowej poczynając od najmniej znaczącego bitu każdego słowa.

- W_BIT : kopiowanie bitów z tablicy słów do tablicy bitowej.
- D_BIT : kopiowanie bitów z tablicy słów podwójnych do tablicy bitowej.

Struktura

Język *Ladder*



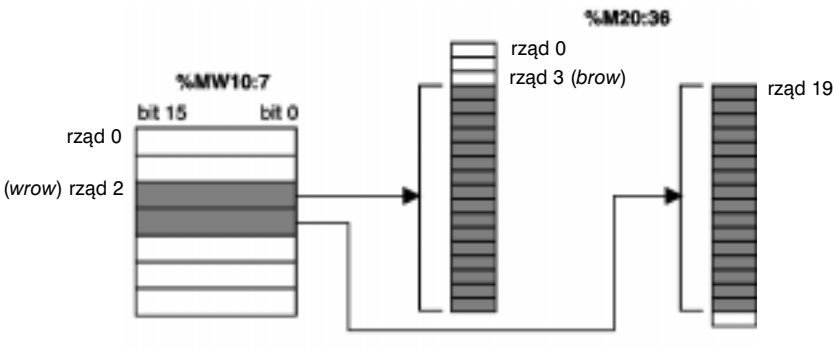
Język *List*

```
LD          TRUE
[%M20:36 := W_BIT (%MW10:7, 2, 2, 3)]

LD          TRUE
[%M20:36 := D_BIT (%MD10:4, 1, 1, 3)]
```

Język *ST*

```
%M20:36 := W_BIT (%MW10:7, 2, 2, 3) ;
%M20:36 := D_BIT (%MD10:4, 1, 1, 3) ;
```

Przykład: %M20:36 := W_BIT (%MW10:7, 2, 2, 3) ;



Składnia

Operator

```
Wynik := W_BIT (Tab, wrow, nwd, brow)
Wynik := D_BIT (Tab, drow, nwd, brow)
```

Argumenty

Typ	Wynik	Tablica (tab)	wrow lub drow nwd - brow
Tablice bitowe	%M:L,%Q:L,%I:L		
Tablice słów		%MW:L,%KW:L	
Tablice słów podwójnych		%MD:L,%KD:L	
Słowa indeksowalne			%MW, %KW, %Xi.T
Słowa nieindeksowalne			%IW, %QW, %SW, %NW, Wart. bezpośrednia Wyr. numeryczne

Uwagi:

- Jeżeli zadana liczba bitów do skopiowania jest większa niż liczba bitów w tablicy, licząc od miejsca rozpoczęcia kopiowania (rząd *wrow*) do końca tablicy, to kopiowanie jest wykonywane do ostatniego elementu tablicy.
- Jeśli zadana liczba bitów do skopiowania przekracza liczbę bitów pozostałych w tablicy wyniko-wej, to funkcja przerywa operację kopiowania po osiągnięciu ostatniego elementu w tablicy.
- Ujemna wartość parametru *brow*, *nbit*, *wrow* lub *drow* traktowana jest jak zero.

2.11 Funkcje "Orphee": przesuwanie, licznik

2.11-1 Operacje przesuwania na słowach z odzyskaniem przesuniętych bitów

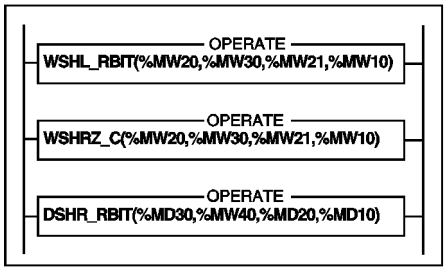
Opisane poniżej funkcje powodują przesunięcie w prawo lub w lewo, o określoną liczbę bitów (*nbit*), w słowie lub słowie podwójnym (a).

Słowo po dokonaniu operacji zapisywane jest jako (wynik), a bity, które zostały usunięte są zapamiętywane jako (reszta).

- WSHL_RBIT : przesunięcie w lewo, w słowie i odzyskanie usuniętych bitów.
- DSHL_RBIT : przesunięcie w lewo, w słowie podwójnym i odzyskanie bitów.
- WSHRZ_C : przesunięcie w prawo, w słowie z uzupełnieniem pustych miejsc przy pomocy 0 i odzyskanie usuniętych bitów.
- DSHRZ_C : przesunięcie w prawo, w słowie podwójnym z uzupełnieniem pustych miejsc przy pomocy 0 i odzyskanie usuniętych bitów.
- WSHR_RBIT : przesunięcie w prawo, w słowie z dołożeniem znaku i odzyskanie usuniętych bitów.
- DSHR_RBIT : przesunięcie w prawo, w słowie podwójnym z dołożeniem znaku i odzyskanie usuniętych bitów.

Struktura

Język *Ladder*



Język *List*

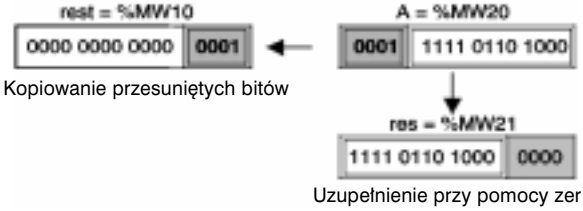
```
LD          TRUE
[WSHL_RBIT(%MW20,%MW30,%MW21,%MW10)]

LD          TRUE
[WSHRZ_C(%MW20,%MW30,%MW21,%MW10)]

LD          TRUE
[DSHR_RBIT(%MD30,%MW40,%MD20,%MD10)]
```

Język *ST*

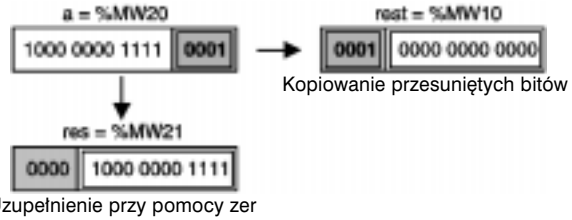
```
WSHL_RBIT (%MW20,%MW30,%MW21,%MW10) ;

WSHRZ_C (%MW20,%MW30,%MW21,%MW10) ;

DSHR_RBIT (%MD30,%MW40,%MD20,%MD10) ;
```

Przykład:

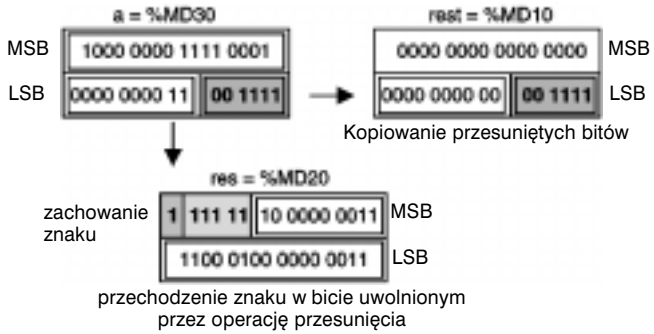
WSHL_RBIT(%MW20,%MW30,%MW21,%MW10) gdzie %MW30 = 4



WSHRZ_C(%MW20,%MW30,%MW21,%MW10) gdzie %MW30 = 4



DSHR_RBIT(%MD30,%MW40,%MD20,%MD10) gdzie %MW40 = 6



Składnia
Operator

```
WSHL_RBIT (a, nbit, wynik, reszta)
WSHRZ_C (a, nbit, wynik, reszta)
WSHR_RBIT (a, nbit, wynik, reszta)
```

Argumenty

Typ	a	nbit	Wynik Reszta
Słowa indeksowalne	%MW,%KW	%MW,%KW, %Xi.T	%MW
Słowa nieindeksowalne	%IW, %QW, %SW, %NW Wartość bezp. Wyr. numeryczne	%IW, %QW, %SW, %NW Wartość bezp. Wyr. numeryczne	%QW, SW, %NW

Składnia

Operator

```
DSHL_RBIT (a, nbit, wynik, reszta)
DSHRZ_C (a, nbit, wynik, reszta)
DSHR_RBIT (a, nbit, wynik, reszta)
```

Argumenty

Typ	a	nbit	Wynik Reszta
Indeks. słowa podwójne	%MD,%KD		%MD
Nieindeks. słowa podwójne	%ID,%QD,%SD Wartość bezp. Wyr. numeryczne		%QD,%SD
Słowa indeksowalne		%MW, %KW, %Xi.T	
Słowa nieindeksowalne		%IW, %QW, %SW, %NW Wartość bezp. Wyr. numeryczne	

Uwaga:

- Jeżeli zadana ilość bitów (parametr *nbit*) nie mieści się w zakresie od 1 do 16 dla operacji zmiany wykonywanej na słowach lub od 1 do 32 w przypadku słów podwójnych, to wartość wyjściowa (wynik) i pozostałość (reszta) są nieważne, a bit systemowy %S18 przyjmuje wartość 1.

2.11-2 Licznik dwukierunkowy z sygnalizacją przepełnienia

Jest to instrukcja ralizująca funkcję licznika typu dwukierunkowego (*up*/*down*) z sygnalizacją osiągnięcia pułapu licznika (ograniczenie dolne lub górne). Funkcję uaktywnia się podając na wejście (en) wartość 1.

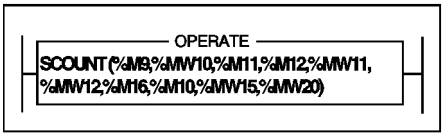
Do zliczania i odliczania służą dwa niezależne wejścia (cu i cd). Wyjście (Qmin) przyjmuje wartość 1, gdy licznik osiąga ograniczenie dolne (min), natomiast wyjście (Qmax) przyjmuje wartość 1, gdy licznik osiąga ograniczenie górne (max).

Wartość początkową licznika definiuje parametr (pv), natomiast bieżąca wartość licznika jest dostępna za pomocą parametru (cv).

Do zapamiętywania stanów wejść (cu i cd) wykorzystywane jest 16-bitowe słowo (mwd). Bit 0 służy do zapisywania wartości na wejściu cu, a bit 1 - wartości cd).

Struktura

Język *Ladder*



Język *List*

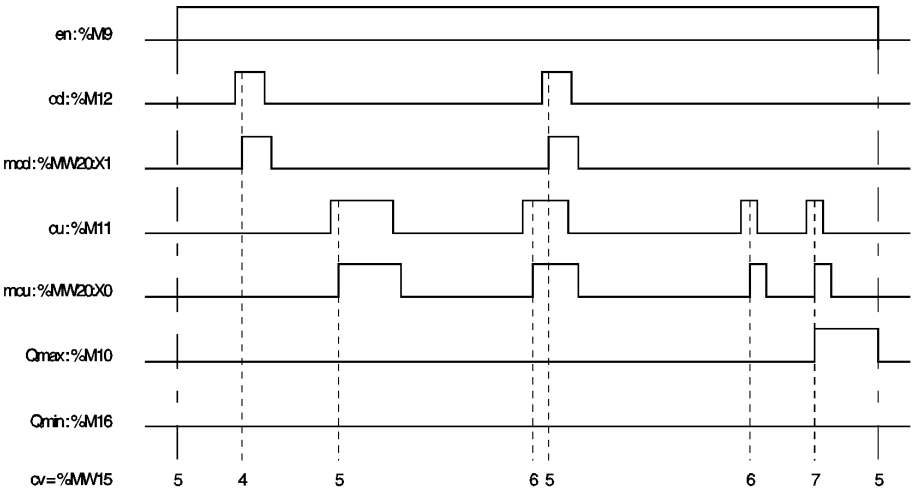
```
LD          TRUE
[SCOUNT(%M9,%MW10,%M11,%M12,%MW11,
%MW12,%M16,%M10,%MW15,%MW20)]
```

Język *ST*

```
SCOUNT(%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW20) ;
```

Przykład:

SCOUNT (%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW20)
gdzie %MW10 (pv) = 5, %MW11 (min) = 0, %MW12 (max) = 7



Składnia

Operator | SCOUNT (en, pv, cu, cd, min, max, Qmin, Qmax, cv, mwd)

Argumenty

Typ	en, cu, cd	Qmin, Qmax	pv, min, max	cv, mwd
Bity	%I,%Q,%M,%S, %BLK,%.:Xk	%I,%Q,%M		
Słowa indeksowalne			%MW,%KW, %Xi.T	%MW
Słowa nieindeks.			%IW, %QW, %SW, %NW, Wartość bezp. Wyr. numeryczne	%QW,%SW %NW

Uwagi:

- Jeżeli (en) = 0 , to funkcja jest nieaktywna, nie reaguje na żadne wywołanie, stąd:
  Qmin = Qmax = 0
  mcu = mcd = 0
  cv = pv
- Jeżeli max > min , to:
  cv ≥ max ---> Qmax = 1 i Qmin = 0
  min < cv < max ---> Qmax = Qmin = 0
  cv ≤ min ---> Qmax = 0 i Qmin = 1
- Jeżeli max < min , to:
  max ≤ cv - min ---> Qmax = 1 i Qmin = 0
  cv < max ---> Qmax = 0 i Qmin = 1
  cv > min ---> Qmax = 1 i Qmin = 0
- Jeżeli max = min , to:
  cv < min i max ---> Qmax = 0 i Qmin = 1
  cv ≥ min i max ---> Qmax = 1 i Qmin = 0
- Próba zmiany wartości parametru (pv), gdy funkcja jest aktywna (czyli na wejściu (en) jest 1), nie wpływa na wykonywaną operację.
- Ujemna wartość parametrów (pv) oraz (min) jest interpretowana jako zero.
- Wartość mniejsza od 1 dla parametru (max) jest interpretowana jako 1.

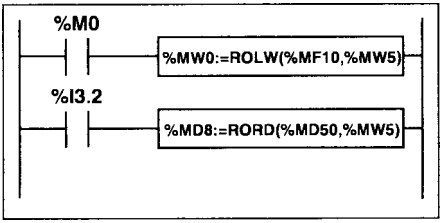
2.11-3 Przesunięcie okrężne

Ta funkcja umożliwia wykonywanie zmiany okrężnej w lewo lub prawo w słowie lub słowie podwójnym.

- ROLW : okrężne przesunięcie w lewo w słowie, o określoną liczbę,
- RORW : okrężne przesunięcie w prawo w słowie, o określoną liczbę,
- ROLD : okrężne przesunięcie w lewo w słowie podwójnym, o określoną liczbę,
- RORD : okrężne przesunięcie w prawo w słowie podwójnym, o określoną liczbę,

Struktura

Język *Ladder*



Język *List*

```
LD          %M0
[%MW0 := ROLW(%MW10,%MW5)]

LD          %I3.2
[%MD10 := RORD(%MD100,%MW5)]
```

Język *ST*

```
IF %M0 THEN
    %MW0 := ROLW(%MW10,%MW5) ;
END_IF ;
IF %I3.2 THEN
    %MD10 := RORD(%MD100,%MW5) ;
END_IF ;
```

Składnia

Operatory

• ROLW, RORW, ROLD, RORD

```
Arg1:=Operator(Arg2,n)
```

Argumenty funkcji ROLW, RORW,

Typ	Argument 1 (Arg1)	Argument 2 (Arg2)	Numer pozycji (n)
Słowa indeksowalne	%MW	%MW,%KW,%Xi.T	%MW,%KW,%Xi.T
Słowa nieindeksowalne		Wart.bezp,%IW,%QW, %SW,%NW,%BLK Wyr. numeryczne	Wart. bezp.,%IW,%QW, %SW,%NW,%BLK Wyr. numeryczne

Argumenty ROLD, RORD

Type	Operand 1 (Op1)	Operand 2 (Op2)	Position number (n)
Indeks. słowa podw.	%MD	%MD,%KD	%MW,%KW,%Xi.T
Nieindeks. słowa podw.	%QD,%SD,	Wart.bezp,%ID,%QD, %SD, Wyr. num.	Wart. bezp.,%IW,%QW, %SW,%NW,%BLK Wyr. numeryczne

Uwaga
Jeżeli liczba przesunięć jest statyczna, to zaleca się stosowanie standardowych funkcji ROL i ROR, ponieważ w takich warunkach przynoszą one najlepsze efekty.

2.12 Funkcje opóźniające *Time delay*

2.12-1 Wprowadzenie

W odróżnieniu od predefiniowanych bloków funkcyjnych, liczba stosowanych funkcji opóźniających nie jest ograniczona, przy czym można ich używać w kodach bloków DFB (własne bloki użytkownika).
Oprogramowanie oferuje 4 funkcje opóźniające:

- FTON : opóźnienie typu *on-delay*
- FTOF : opóźnienie typu *off-delay*
- FTP : generowanie impulsu o określonym czasie trwania
- FPULSOR : generowanie przebiegu prostokątnego

2.12-2 Funkcja opóźniająca FTON

Jest to funkcja realizująca opóźnienie typu *on-delay*. Opóźnienie to można programować.

Składnia

Operator

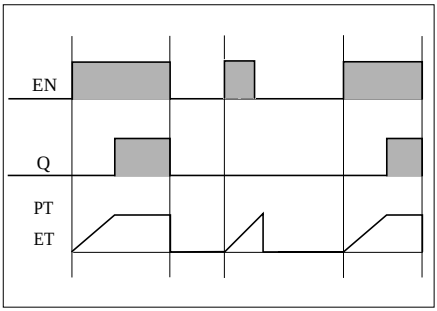
FTON (EN, PT, Q, ET, PRIV)

Charakterystyka

Wejście "Enable"	EN	Wyzwolenie opóźnienia (zbocze rosnące)
Wartość nastawiona	PT	Jest to słowo wejściowe definiujące czas trwania (w setnych częściach sekundy) opóźnienia. Wartość maksymalna wynosi 5 min i 27 s, z dokładnością do 10 ms. (1)
Wyjście "Timer"	Q	Po odliczeniu opóźnienia na wyjściu wystawiana jest 1
Wartość bieżąca	ET	Słowo wyjściowe, którego wartość, podczas odliczania opóźnienia, rośnie od 0 do PT.
Zmienna obliczeniowa	PRIV	Słowo podwójne przeznaczone do przechowywania stanów wewnętrznych. Jest to zmienna zarezerwowana wyłącznie dla tych operacji.

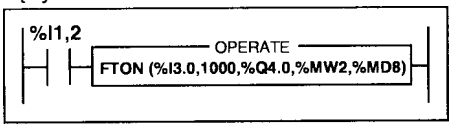
(1) Zmiana wartości tego słowa jest uwzględniana w trakcie odliczania opóźnienia.

Uruchomienie zegara następuje po pojawieniu się zbocza rosnącego na wejściu EN: wartość bieżąca ET rośnie od 0 do PT (w setnych częściach sekundy). Bit wyjścia Q zmienia stan na 1, gdy wartość bieżąca zrównuje się z wartością nastawioną PT i utrzymuje ten stan tak długo, jak długo na wejściu EN jest 1.
Gdy stan EN zmienia się na 0, zegar zatrzymuje się (nawet jeśli nie skończył liczenia): ET przyjmuje wartość 0.



Struktura

Język *Ladder*



Język *List*

```
LD %I1.2
[FTON (%I3.0,1000,%Q4.0,%MW2,%MD8)]
```

Język *ST*

```
IF %I1.2 THEN
    FTON (%I3.0,1000,%Q4.0,%MW2,%MD8) ;
END_IF ;
```

Argumenty

FTON (EN, PT, Q, ET, PRIV)

Typ	EN	PT	Q	ET	PRIV
Słowa indeks.		%MW,%KW,%Xi.T		%MW	
Słowa nieindeksowalne		%IW,%QW,%SW Wart. bezp., Wyr. num.,%NW		%IW,%QW	
Indeks. słowa podwójne					%MD
Bity	%I,%Q, %M, %S %BLK,%•:Xk,%X		%I,%Q, %M, %S,%•:Xk,%X		

2.12-3 Funkcja opóźniająca FTOF

Jest to funkcja realizująca opóźnienie typu *off-delay*. Opóźnienie to można programować.

Składnia

Operator

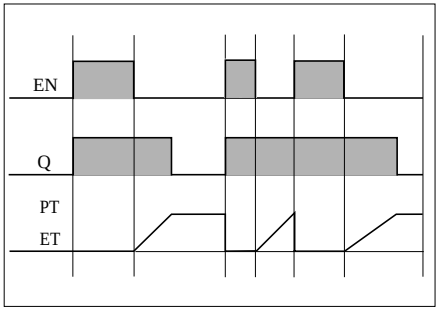
```
FTOF (EN, PT, Q, ET, PRIV)
```

Charakterystyka

Wejście "Enable"	EN	Wyzwolenie opóźnienia (zbocze opadające)
Wartość nastawiona	PT	Jest to słowo wejściowe definiujące czas trwania (w setnych częściach sekundy) opóźnienia. Wartość maksymalna wynosi 5 min i 27 s, z dokładnością do 10 ms. (1)
Wyjście "Timer"	Q	Przy pojawieniu się 1 na wejściu EN, na wyjściu tym pojawia się 1, natomiast po odliczeniu opóźnienia - wystawiane jest 0.
Wartość bieżąca	ET	Słowo wyjściowe, którego wartość, podczas odliczania opóźnienia, rośnie od 0 do PT.
Zmienna obliczeniowa	PRIV	Słowo podwójne przeznaczone do przechowywania stanów wewnętrznych. Jest to zmienna zarezerwowana wyłącznie dla tych operacji.

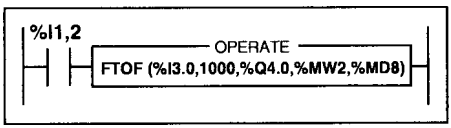
(1) Zmiana wartości tego słowa jest uwzględniana w trakcie odliczania opóźnienia.

Pojawienie się zbocza rosnącego na wejściu EN, powoduje skasowanie wartości bieżącej ET do 0 (nawet, gdy zegar wciąż liczy). Pojawienie się zbocza opadającego na wejściu EN powoduje uruchomienie zegara (zaczyna liczyć).
Wartość bieżąca zaczyna rosnąć od 0 do PT (w setnych częściach sekundy).
Bit wyjściowy Q przyjmuje wartość 1, gdy pojawia się zbocze rosnące na wejściu EN i zmienia stan na 0 w momencie osiągnięcia wartości nastawionej PT.



Structure

Język *Ladder*



Język *ST*

```
IF %I1.2 THEN
 FTOF (%I3.0,1000,%Q4.0,%MW2,%MD8) ;
END_IF ;
```

Argumenty - patrz funkcja FTON (poprzedni rozdział).

2.12-4 Funkcja FTP - generowanie impulsu

Funkcja ta umożliwia wygenerowanie impulsu o ściśle określonym czasie trwania. Czas trwania impulsu może być programowany.

Składnia

Operator

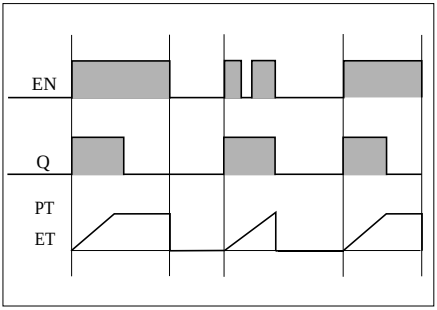
FTP (EN, PT, Q, ET, PRIV)

Charakterystyka

Wejście "Enable"	EN	Wyzwolenie opóźnienia (zbocze rosnące)
Wartość nastawiona	PT	Jest to słowo wejściowe definiujące czas trwania (w setnych częściach sekundy) opóźnienia. Wartość maksymalna wynosi 5 min i 27 s, z dokładnością do 10 ms. (1)
Wyjście "Timer"	Q	Po odliczeniu opóźnienia, na wyjściu wystawiana jest 1
Wartość bieżąca	ET	Słowo wyjściowe, którego wartość, podczas odliczania opóźnienia, rośnie od 0 do PT.
Zmienna obliczeniowa	PRIV	Słowo podwójne przeznaczone do przechowywania stanów wewnętrznych. Jest to zmienna zarezerwowana wyłącznie dla tych operacji.

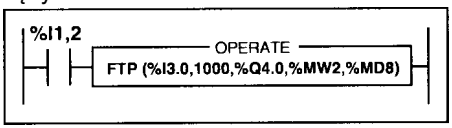
(1) Zmiana wartości tego słowa jest uwzględniana w trakcie odliczania opóźnienia.

Pojawienie się zbocza rosnącego na wejściu EN powoduje uruchomienie odliczania (o ile zegar nie pracuje). Bieżąca wartość ET rośnie od 0 do PT (wartość w setnych częściach sekundy). Bit wyjściowy Q zmienia stan na 1 w momencie uruchomienia odliczania, a gdy wartość bieżąca osiąga poziom wartości nastawionej PT - zmienia stan na 0.
Gdy na wejściu EN i na wyjściu Q jest 0, to PT również przyjmuje wartość 0.
Ten przerzutnik monostabilny nie może być reaktywowany.



Struktura

Język *Ladder*



Język *ST*

```
IF %I1.2 THEN
 FTP (%I3.0,1000,%Q4.0,%MW2,%MD8) ;
END_IF ;
```

Argumenty - patrz funkcja FTON (poprzedni rozdział).

2.12-5 Funkcja FPULSOR - generowanie przebiegu prostokątnego

Ta funkcja powoduje wygenerowanie okresowego sygnału prostokątnego. Szerokość impuslu definiuje się programowo (zmiana stanów 1 - 0) za pomocą 2 liczników:

- TON : zegar opóźniający typu *on-delay* (dla impulsu 1).
- TOFF : zegar opóźniający typu *off-delay* (dla impulse 0).

Składnia

Operator

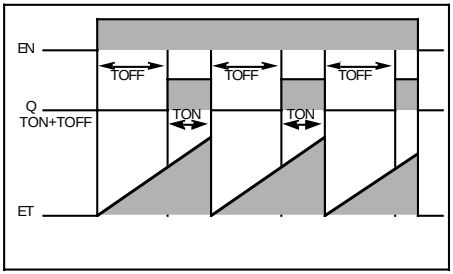
FPULSOR (EN, TON, TOFF, Q, ET, PRIV)

Charakterystyka

Wejście "Enable"	EN	Wyzwolenie sygnału prostokątnego (zbocze rosnące)
Wartość nastawiona (impuls 1)	TON	Jest to słowo wejściowe definiujące czas trwania (w setnych częściach sekundy) impulsu (stan 1). Wartość maksymalna wynosi 5 min i 27 s, z dokładnością do 10 ms. (1)
Wartość nastawiona (impuls 0)	TOFF	Jest to słowo wejściowe definiujące czas trwania (w setnych częściach sekundy) impulsu (stan 0). Wartość maksymalna wynosi 5 min i 27 s, z dokładnością do 10 ms. (1)
Wyjście sygnału	Q	Wyjście przebiegu prostokątnego przyjmujące wartość 0 w czasie TOFF, i 1 w czasieTON.
Wartość bieżąca	ET	Słowo wyjściowe, którego wartość, podczas odliczania opóźnienia, rośnie od 0 do TON+TOFF.
Zmienna obliczeniowa	PRIV	Słowo podwójne przeznaczone do przechowywania stanów wewnętrznych. Jest to zmienna zarezerwowana wyłącznie dla tych operacji.

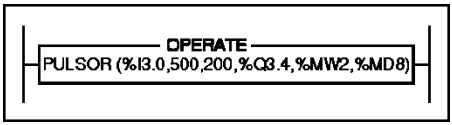
(1) Dowolne zmiany dokonywane na tym słowie są uwzględniane podczas odliczania opóźnienia. Czas TOFF+TON może mieć maksymalnie wartość równą 5 min i 27 s.

Pojawienie się zbocza rosnącego na wejściu EN powoduje wygenerowanie przebiegu prostokątnego (o ile przebieg ten nie jest już generowany). Wartość bieżąca ET rośnie wtedy od 0 do TON+TOFF (w setnych częściach sekundy). Bit wyjściowy Q ma, podczas odliczania opóźnienia TOFF, stan 0; stan zmienia się na 1 (i utrzymuje się 1) podczas odliczania opóźnienia TON, po czym znowu zmienia się na 0 podczas odliczania czasu TOFF i tak w kółko dopóki na wejściu EN nie pojawi się 0.



Struktura

Język *Ladder*



Język *List*

```
LD True
[FPULSOR (%I3.0,500,200,%Q4.0,%MW2,
%MD8)]
```

Język *ST*

```
IF %I1.2 THEN
    FPULSOR (%I3.0,500,200,%Q4.0,%MW2,%MD8) ;
END_IF ;
```

Argumenty

FPULSOR (EN, TON, TOFF, Q, ET, PRIV)

Typ	EN	TON,TOFF	Q	ET	PRIV
Słowa indeks.	%MW,%KW,%Xi.T		%MW		
Słowa nieindeks.		%IW,%QW,%SW Wart. bezp. Wyr. num.,%NW		%IW,%QW	
Indeks. słowa podwójne					%MD
Bity	%I,%Q, %M, %S %BLK,%•:Xk,%X		%I,%Q, %M, %S,%•:Xk,%X		

3.1 Bity systemowe

3.1-1 Lista bitów

Bit	Funkcja	Wartość początk.	Kontrola (1)
%S0	1 = zimny start (zanik zasilania z utratą danych)	0	S lub U->S
%S1	1 = gorący start (zanik zasilania bez utraty danych)	0	S lub U->S
%S4,%S5, %S6,%S7	Podstawa czasu 10ms, 100ms, 1s, 1mn	-	S
%S8 (2)	Testowanie podłączeń (może być użyte dla nie skonfigurowanego TSX 37)	1	U
%S9	1 = wymuszenie stanów wyjść w trybie zastępczym	0	U
%S10	0 = błąd I/O	1	S
%S11	1 = zadziałanie układu monitorującego (*watchdog*)	0	S
%S13	1 = pierwsze "przejście" po ustawieniu trybu RUN	-	S
%S15	1 = błąd łańcucha znaków	0	S->U
%S16	0 = błąd I/O w zadaniu	1	S->U
%S17	1 = przepełnienie	0	S->U
%S18	1 = przepełnienie lub błąd arytmetyczny	0	S->U
%S19	1 = przekroczenie okresu zadania	0	S->U
%S20	1 = przekroczenie indeksu	0	S->U
%S21	1 = inicjacja *Grafcet-u*	0	S lub U->S
%S22	1 = kasowanie *Grafcet-u*	0	U->S
%S23	1 = przygotowanie *Grafcet-u* i zamrożenie	0	U->S
%S24 (3)	1 = kasowanie makra do 0 zgodnie z %SW22 to 25	0	S
%S26	1 = przepełnienie tablicy (kroki/bramki)	0	S
%S30	1 = uaktywnienie zadania głównego MASTER	1	U
%S31	1 = uaktywnienie zadania szybkiego FAST	1	U
%S38	1 = odblokowanie kontroli zdarzeń	1	U
%S39	1 = nasycenie procesu przetwarzania zdarzeń	0	S->U
%S40 ÷ %S47	1 = błąd I/O dla slotu TSX 57 (3)	1	S
%S49	1 = kasowanie błędnych stanów wyjść logicznych	0	U
%S50	1 = ustawianie zegar czasu rzeczywistego	0	U

(1) (2) i (3) - patrz następna strona.

Bit	Funkcja	Wartość początk.	Kontrola (1)
%S51	1 = brak czasu z zegara czasu rzeczywistego	0	S lub U->S
%S59	1 = umożliwienie zmiany bieżącej daty	0	U
%S66 (2)	1 = kontrolka baterii zawsze wyłączona	0	U
%S67	0 = zasilanie bateryjne zasobnika pamięci	-	S
%S68	0 = praca baterii podtrzymującej (procesor)	-	S
%S69 (2)	1 = odblok. pamięci słów (WORD) wyświetlacza	0	U
%S70	1 = aktualizacja danych szyny As-i lub łącza TSX Nano	0	S->U
%S73 (2)	1 = przełączenie na tryb chroniony szyny AS-i	0	U->S
%S74 (2)	1 = zapisanie konfiguracji AS-i	0	U->S
%S80	1 = kasowanie komunikatu licznika	0	S->U
%S90	1 = aktualizacja słów wspólnych	0	S->U
%S94 (3)	1 = zapisanie wartości korygujących dla bloku DFB	0	U->S
%S95 (3)	1 = odtworzenie wartości korygujących bloku DFB	0	U->S
%S96 (2)	0 = brak kopii zapasowej programu 1 = kopia zapasowa programu jest dostępna	0	S->U
%S97 (2)	0 = brak kopii %MW 1 = jest kopia %MW	-	S
%S98 (2)	1 = zastąpienie modułu przycisku TSX SAZ 10 wejściem dyskretnym	0	U
%S99 (2)	1 = zastąpienie centralnego modułu przycisków wejściem dyskretnym	0	U
%S100	1 = odtworzenie wartości korygujących bloku DFB	-	S
%S118	0 = błąd I/O FIPIO	1	S
%S119	0 = wewnętrzny błąd I/O	1	S

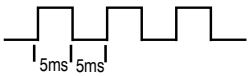
(1) S = kontrolowane przez system, U =kontrolowane przez użytkownika, U->S = wartość 1 nadawana przez użytkownika, kasowanie do 0 przez system, S->U = wartość 1 nadawana przez system, kasowanie do 0 przez użytkownika.

(2) Tylko dla TSX 37.

(3) Tylko dla TSX/PMX/PCX 57.

3.1-2 Szczegółowy opis bitów systemowych

Sterowniki TSX 37 i TSX 57 mają bity systemowe %Si, które sygnalizują stan sterownika lub uprawniają użytkownika do interwencji w jego działanie.
Te bity mogą być testowane w programie użytkowym, w celu wykrywania sytuacji wymagających specjalnej reakcji. Niektórym z nich, program musi przywracać stan początkowy lub normalny. Jednakże należy pamiętać, że bitów systemowych, którym system przywrócił stan początkowy lub normalny, nie wolno kasowanać za pomocą programu czy terminala.

Bity system.	**Funkcja**	**Opis**
%S0	Zimny start *Cold start*	Normalnie stan 0. Nadanie wartości 1 może być wywołane: • zanikiem zasilania z utratą danych (uszkodzenie baterii), • programowo przez użytkownika, • za pomocą terminala, • wymianą zasobnika pamięci, • naciśnięciem klawisza RESET, Bit przyjmuje wartość 1 w trakcie pierwszego, kompletnego "przejścia" programu. Jest kasowane do 0 przed następnym "przejściem". Działanie: patrz część A, rozdział 1.4.
%S1	Gorący start *Warm restart*	Normalnie stan 0. Nadanie wartości 1 może być wywołane: • zanikiem zasilania z zachowaniem danych, • programowo przez użytkownika, • za pomocą terminala, Bit jest kasowany przez system do 0 na końcu pierwszego, kompletnego "przejścia" programu, przed uaktualnieniem stanów wyjść. Działanie: patrz część A, rozdział 1.4.
%S4 **%S5** **%S6** **%S7**	Podstawa czasu *Time base* 10ms 100ms 1s 1min	Zmianą stanów tych bitów zarządza wewnętrzny zegar. Nie są one synchronizowane z pracą sterownika. Przykład: %S4 
%S8	Podłączenia *Wiring*	Normalnie - 1, ten bit służy do testowania podłączeń, gdy sterownik TSX 37 nie jest skonfigurowany. • Stan 1 powoduje wymuszenie 0 na wyjściach. • Stan 0 uprawnia do dokonywania zmian za pomocą terminala.
%S9	Tryb zastępczy wyjść na wszystkich magistralach	Normalnie - 0. Zmiana na 1 - programowo lub terminal. • Stan 1 - wymuszenie na wyjściach trybu zastępczego (magistrala X, FIPIO, AS-i, itp.). • Stan 0 - wyjścia są normalnie aktualizowane.
%S10	Błąd I/O	Normalnie - 1. Zmienia stan na 0, gdy pojawia się błąd I/O modułu lokalnego lub zdalnego FIPIO (błąd konfiguracji, błąd wymiany, błąd sprzętowy). Bit %S10 powraca do stanu 1, gdy znika przyczyna błędu.

Bity system.	Funkcja	Opis
%S11	Układ śledzący *Watchdog*	Normalnie - 0. System nadaje 1, gdy czas wykonywania zadania przekracza maksymalny czas wykonywania, zdefiniowany dla zadania podczas konfiguracji. Zadziałanie zabezpieczenia powoduje zatrzymanie sterownika (przejście do trybu STOP) a aplikacja przerywa pracę sygnalizując błąd (miga kontrolka ERR).
%S13	1-e "przejście" *First scan*	Normalnie - 0. System utrzymuje wartość 1 przez czas trwania pierwszego "przejścia" po uruchomieniu sterownika (tryb RUN).
%S15	Błąd łańcucha znaków	Normalnie - 0. Przyjmuje wartość 1, gdy obszar docelowy dla transferu łańcucha znaków jest zbyt mały. Bit ten musi być skasowany przez użytkownika.
%S16	Błąd zadania I/O	Normalnie - 1. System nadaje mu wartość 0, gdy pojawia się błąd w lokalnym lub zdalnym (FIPIO) module I/O. Bit ten musi być skasowany do 1 przez użytkownika. Zadanie I/O ma swój własny bit %S16.
%S17	Bit wyjściowy dla operacji arytmetycznych lub przesunięcia	Normalnie - 0. System nadaje mu wartość 1: • podczas operacji przesuwania (zawiera stan ostatniego bitu), • w przypadku przepełnienia dla bezznakowej operacji arytmetycznej (na datach). Ten bit musi być skasowany do 0 przez użytkownika.
%S18	Błąd lub przepełnienie operacji arytmetycznej	Normalnie - 0. Przyjmuje wartość 1, w przypadku przepełnienia operacji 16-bitowej, czyli gdy: • wynik jest większy niż + 32767 lub mniejszy niż - 32768 dla operacji na obiektach pojedynczej precyzji, • wynik jest większy niż + 2 147 483 647 lub < - 2 147 483 648 dla operacji na obiektach podwójnej precyzji, • wynik jest większy niż +3.402824E+38 lub < -3.402824E+38 dla operacji zmiennoprzecinkowych (wersja programu > 1.0), • następuje przekroczenie pojemności kodu DCB, • występuje dzielenie przez 0, • argument pierwiastka kwadratowego jest ujemny, • następuje odwołanie do nieistniejącego kroku bębna, • podejmowana jest próba zapisania w zapełnionym rejestrze lub odczytania pustego rejestru. Bit ten musi być testowany przez program po każdej operacji, która niesie zagrożenie wystąpienia przepełnienia. W razie jego wystąpienia użytkownik powinien skasować go do 0.
%S19	Przekroczenie okresu zadania (zadanie okresowe)	Normalnie - 0. System nadaje mu wartość 1 w przypadku przekroczenia okresu zadania (czas "przejścia" zadania większy niż czas zdefiniowany przez użytkownika w konfiguracji lub zapisany w słowie %SW sprzężonym z zadaniem)

Bity system.	Funkcja	Opis
%S20	Przekroczenie indeksu	Normalnie - 0. Przyjmuje wartość 1, gdy adres obiektu indeksowanego jest mniejszy od 0 lub jest większy od liczby obiektów zadeklarowanej w konfiguracji. Bit ten musi być testowany przez program po każdej operacji, dla której istnieje ryzyko wystąpienia przepełnienia. W razie jego wystąpienia powinien być skasowany do 0 przez użytkownika.
%S21	Inicjacja *Grafcet-u*	Bit pozwala użytkownikowi na inicjację *Grafcet-u* (zaleca się nadawanie mu wartości 1 podczas przetwarzania wstępnego). Jest kasowany do 0 przez system po zakończeniu inicjacji (na końcu przetwarzania wstępnego, podczas oszacowywania nowego statusu *Grafcet-u*). Inicjacja powoduje dezaktywację wszystkich aktywnych kroków i uaktywnienie kroków inicjujących (*initial steps*). W przypadku zimnego startu system nadaje mu, podczas przetwarzania wstępnego, wartość 1.
%S22	Kasowanie *Grafcet-u*	Normalnie - 0. Wartość 1 może nadać temu bitowi tylko system podczas przetwarzania wstępnego. Stan 1 powoduje dezaktywację wszystkich kroków *Grafcet-u*. System kasuje go do 0 po zakończeniu przetwarzania wstępnego.
%S23	Zamrożenie *Grafcet-u*	Normalnie - 0. Nadanie bitowi wartości 1 powoduje zapamiętanie statusu *Grafcet-u*. Bez względu na spełnienie warunków przejścia diagram nie ulega zmianie. Zamrożenie trwa tak długo, jak długo bit %S23 ma stan 1. Stan bitu jest kontrolowany przez program. Wartość definiuje się tylko podczas przetwarzania wstępnego.
%S24	Kasowanie makrodefinicji	Normalnie - 0. Nadanie wartości 1 powoduje skasowanie do 0 wybranych makrodefinicji (tablica 4 słów %SW22÷%SW25). Po zatwierdzeniu system kasuje bit do 0, po zakończeniu przetwarzania wstępnego.
%S26	Przepełnienie tablicy (kroków/bramek)	Normalnie - 0. System nadaje wartość 1, w razie przekroczenia liczby uaktywnionych kroków lub bramek lub przy próbie wykonania niewłaściwie zdefiniowanego diagramu (np. połączenie z krokiem z innego diagramu). Powoduje to zatrzymanie sterownika (STOP). Skasowanie do 0 następuje po zainicjowaniu terminala.
%S30	Uaktywnienie zadania MAST	Normalnie - 1. Nadanie mu, przez użytkownika, wartości 0 powoduje dezaktywację zadania głównego MASTER.
%S31	Uaktywnienie zadania FAST	Normalnie - 1. Nadanie mu, przez użytkownika, wartości 0 powoduje dezaktywację zadania szybkiego FAST.
%S38	Oblokowanie zdarzeń	Normalnie - 1. Nadanie mu, przez użytkownika, wartości 0 powoduje zablokowanie reakcji na zdarzenia (*events*).
%S39	Przepełnienie przetwarzania zdarzeń	System nadaje bitowi %S39 wartość 1 sygnalizując, że z powodu przepełnienia stosu nie można przetwarzać zdarzeń. Bit ten jest kasowany do 0 przez użytkownika.

Bity system.	Funkcja	Opis
%S40 do %S47	Błędy I/O (wewn.) (1)	Bity %S40 do %S47 są przypisane do slotów 0 do 7. Normalnie mają one wartość 1. Jeżeli w danym slocie wystąpi błąd I/O, to odpowiedni bit przyjmuje wartość 0. Po zniknięciu błędu bit jest kasowany do stanu 1.
%S49	Reaktywacja wyjść (2)	Normalnie - 0. Nadanie mu, przez użytkownika, wartości 1 jest żądaniem reaktywacji (co 10s) wyjścia logicznego sygnalizującego przeciążenie lub zwarcie.
%S50	Aktualizacja daty i czasu za pom. słów %SW50÷53	Normalnie - 0. Wartość 1 lub 0 może mu nadawać program lub terminal. • Stan 0 umożliwia dostęp do daty i czasu (odczytanie słów) • Stan 1 umożliwia uaktualnienie daty i czasu poprzez zapisanie nowych wartości słów systemowych %SW50 to 53.
%S51	Brak czasu z zegara czasu rzeczywistego	Bitem tym zarządza system, wartość 1 sygnalizuje brak zegara czasu rzeczywistego lub niewłaściwe wartości słów systemowych. W takim przypadku należy nastawić zegar. Nastawienie zegara automatycznie zmienia stanu bitu na 0.
%S59	Aktualizacja daty i czasu za pomocą słowa %SW59	Normalnie - 0. Wartość 1 lub 0 może mu nadawać program lub terminal. • Stan 0 powoduje, że system nie kontroluje słowa %SW59. • Stan 1 powoduje, że system kontroluje zmianę stanu słowa %SW59 (zbocze rosnące lub opadające) i dostraja bieżący czas i datę.
%S66	Sygnalizacja stanu baterii	Normalnie - 0. Wartość 1 lub 0 może mu nadawać program lub terminal. Umożliwia on załączanie i wyłączanie sygnalizacji stanu baterii podtrzymującej (brak lub uszkodzenie): • stan 0: w razie braku lub uszkodzenia baterii kontrolka pali się, • stan 1: kontrolka baterii jest zawsze wyłączona. Podczas zimnego startu system kasuje bit %S66 do 0.
%S67	Stan baterii zasobnika	Bit ten umożliwia sprawdzenie stanu baterii podtrzymującej zasobnika pamięci RAM. • Stan 0: bateria pracuje prawidłowo, • Stan 1: brak baterii lub jej uszkodzenie.
%S68	Stan baterii procesora	Bit ten umożliwia sprawdzenie stanu baterii podtrzymującej pamięci RAM dla programu i danych. • Stan 0: bateria pracuje prawidłowo, • Stan 1: brak baterii lub jej uszkodzenie.

(1) Tylko dla sterowników TSX/PMX/PCX 57.

(2) Tylko dla sterowników TSX37.

Bity system	Funkcja	Opis
%S69	Wyświetlanie danych użytkowych na module sterownika	Normalnie - 0. Wartość 1 lub 0 nadaje mu program lub terminal. • Stan 0: na module sterownika jest wyświetlany stan I/O (kontrolka WRD jest wyłączona). • Stan 1 wyświetlane są dane użytkowe (WRD się świeci). (patrz słowa %SW67,68 i 69).
%S70	Aktualizacja danych na szynie As-i lub łączu TSX Nano	Na końcu każdego przeszukania sieci TSX Nano lub As-i system nadaje bitowi wartość 1. Po załączeniu bit ten informuje, że wszystkie dane zostały co najmniej raz odświeżone, w związku z czym są ważne. Bit jest kasowany do 0 przez użytkownika.
%S73	Przełączenie na tryb chroniony na szynie As-i	Normalnie - 0. Użytkownik nadaje bitowi wartość 1 w celu przełączenia magistrali As-i na pracę w trybie chronionym. Jednakże najpierw bit %S74 powinien mieć stan 1. Bit ten nie ma zastosowania w sterowniku, służy tylko do testowania połączeń.
%S74	Zapamiętanie bieżącej konfiguracji magistrali AS-i	Normalnie - 0. Nadanie mu, przez użytkownika wartości 1 powoduje zapisanie do pamięci bieżącej konfiguracji magistrali As-i. Bit ten nie ma zastosowania w sterowniku, służy tylko do testowania połączeń.
%S80	Kasowanie licznika komunikatów	Normalnie - 0. Użytkownik może mu nadać wartość 1, co powoduje skasowanie liczników komunikatów %SW80 do %SW86.
%S90	Odświeżenie słów wspólnych	Normalnie - 0. Jeżeli słowa wspólne pochodzą od innej stacji sieci, to bit ten przyjmuje wartość 1. Program lub terminal może zmieniać wartość bitu na 0 w celu sprawdzenia cyklu wymiany słów wspólnych.
%S94	Zapisanie korekcji DFB	Normalnie - 0. Użytkownik zmienia stan na 1 w celu zapisania wartości korygujących (*adjustment values*) dla bloku DFB.
%S95	Odczytanie korekcji DFB	Normalnie - 0. Użytkownik zmienia stan na 1 w celu wczytania wartości korygujących do bloku DFB.
%S96	Informacja o kopii programu	0 -> brak kopii zapasowej programu, 1 -> kopia programu jest dostępna. Stan bitu może być odczytywany w dowolnym momencie (przez program w trybie regulacji *adjustment mode*), a w szczególności po zimnym lub gorącym starcie. Jego stan informuje o możliwości wczytania do sterownika kopii zapasowej programu (*backup application*) zapisanej za pomocą PL7 w wewnętrznej pamięci *Flash EPROM*.
%S97	Informacja o kopii %MW	0 -> brak kopii słów %MW, 1 -> kopia słów %MW jest dostępna. Stan można czytać w dowolnym momencie (program w trybie regulacji), w szczególności po zimnym lub gorącym starcie.
%S98	Funkcja modułu TSX SAZ 10	Normalnie - 0. Bitem zarządza użytkownik: 0 -> przycisk na module TSX SAZ 10 jest aktywny, 1 -> przycisk na TSX SAZ 10 zastąpiony wejściem dyskretnym (patrz %SW98).

Bity system.	Funkcja	Opis
%S99	Przycisk na module sterownika (blok sygnalizacyjny)	Normalnie - 0. Bitem zarządza użytkownik: 0 -> przycisk na module sterownika aktywny, 1 -> przycisk na module zastąpiony przez wejście dyskretne (patrz %SW99).
%S100	Protokół portu terminala	W zależności od stanu zworki INL/DPT na porcie terminala system nadaje bitowi wartość 0 lub 1. • Jeżeli nie ma zworki (%S100=0), to stosowany jest protokół UNI-TELWAY. • Jeżeli jest zworka (%S100=1), to używany jest protokół zdefiniowany w konfiguracji.
%SW118	Ogólny błąd I/O FIPIO	Normalnie - 1. Bit zmienia wartość na 0, gdy jakieś urządzenie podłączone do szyny FIPIO sygnalizuje błąd I/O. Po zniknięciu błędu bit jest kasowany przez system do 1.
%SW119	Ogólny błąd I/O wewnętrzny	Normalnie - 1. Bit zmienia wartość na 0, gdy jakieś urządzenie wewnętrzne sygnalizuje błąd I/O. Po zniknięciu błędu bit jest kasowany przez system do 1.

3.2 Słowa systemowe

3.2-1 Lista słów

Słowo	Funkcja	Kontrola
%SW0	Wartość okresu zadania głównego MAST (zadanie okresowe)	U
%SW1	Wartość okresu zadania szybkiego FAST	U
%SW8	Sterowanie odczytem stanów wejść dla każdego zadania	U
%SW9	Sterowanie aktualizacją stanów wyjść dla każdego zadania	U
%SW10	Pierwsze "przejście" po zimnym starcie	S
%SW11	Wartość czasu dla układu śledzącego (*watchdog*)	S
%SW12	Adres portu terminala w sieci UNI-TELWAY	S
%SW13	Główny adres stacji	S
%SW17	Status błędu dla operacji zmiennoprzecinkowych	S i U
%SD18	Zegar czasu bezwzględnego	S i U
%SW20	Liczba kroków aktywnych, do uaktywnienia i do dezaktywacji	S
%SW21	Liczba bramek otwartych, do otwarcia i do zamknięcia	S
%SW22 ÷ %SW25 (2)	Tablica 4 słów używana do zaznaczania makr przeznaczonych do skasowania do 0 gdy bit %S24 przyjmie wartość 1	U
%SW30	Czas wykonywania ostatniego zadania głównego	S
%SW31	Maksymalny czas trwania "przejścia" dla zadania głównego	S
%SW32	Minimalny czas trwania "przejścia" dla zadania głównego	S
%SW33	Czas wykonywania ostatniego zadania szybkiego	S
%SW34	Maksymalny czas trwania "przejścia" dla zadania szybkiego	S
%SW35	Minimalny czas trwania "przejścia" dla zadania szybkiego	S
%SW48	Liczba przetworzonych zdarzeń	S i U
%SW49 (1) **%SW50 (1)** **%SW51 (1)** **%SW52 (1)** **%SW53 (1)**	Funkcje zegara czasu rzeczywistego: słowa zawierające bieżącą datę i czas (w kodzie BCD) %SW49 = dzień tygodnia (nazwa dnia) %SW50 = sekundy %SW51 = godziny i minuty %SW52 = miesiąc i dzień %SW53 = rok	Si U
%SW54 (1) **%SW55 (1)** **%SW56 (1)** **%SW57 (1)**	Funkcje zegara czasu rzeczywistego: słowa zawierające czas i datę ostatniej awarii zasilania lub zatrzymania sterownika (BCD) %SW54 = sekundy i kod błędu %SW55 = godzina i minuta %SW56 = miesiąc i dzień %SW57 = rok	S
%SW58	Identyfikacja przyczyny ostatniego stopu i dzień tygodnia (nazwa)	S
%SW59	Poprawianie bieżącej daty i czasu	U
%SW67 **%SW68** **%SW69**	Tryb wyświetlania: %SW67: odczytywanie stanów przycisków %SW68: indeksy - bieżący i max "wyświetlanych obiektów" %SW69: numer pierwszego wyświetlanego obiektu	S i U
%SW80 **%SW81** **%SW82** **%SW83** **%SW84** **%SW85** **%SW86**	L-ba komunikatów przesłanych przez system do portu terminala L-ba komunikatów odebranych przez system od portu terminala L-ba komunikatów przesłanych przez system do karty PCMCIA L-ba komunikatów odebranych przez system od karty PCMCIA Liczba telegramów wysłanych przez system Liczba telegramów odebranych przez system Liczba telegramów odrzuconych przez system	S i U

(1) S = zarządzany przez system, U = zarządzany przez użytkownika

(2) Tylko dla TSX 57.

(3) Tylko dla TSX 37-21/22 i TSX 57.

Słowo	Funkcja	Kontrola
%SW96 (2)	Kontrola/diagnostyka zapisywania i odczytywania programu aplikacji i %MW	S i U
%SW97 (2)	Liczba słów %MW do zapisania	U
%SW98 (2)	Fizyczny adres (moduł/kanał) wejścia dyskretnego zastępującego przycisk na module TSX SAZ 10	U
%SW99 (2)	Fizyczny adres (moduł/kanał) wejścia dyskretnego zastępującego przycisk na module sterownika (blok sygnalizacyjny)	U
%SW108	Liczba wymuszonych bitów	S
%SW109	Liczba kanałów analogowych, na których wymuszono 0	S
%SW116	Błąd zdalnych I/O w sieci FIPIO	S
%SW124	Rodzaj ostatniego, wyszukanego procesora	S
%SW125	Typ błędu blokującego działanie procesora	S
%SW126 %SW127	Adres instrukcji błędu blokującego	S
%SW128÷ %SW143	Błąd podłączenia w sieci FIPIO	S
%SW155	Liczba przeprowadzonych operacji wymiany jawnej	S

(1) S = kontrolowane przez system, U = kontrolowane przez użytkownika,
(2) Tylko dla TSX 37.

3.2-2 Szczegółowy opis słów systemowych

Słowa systemowe	Funkcja	Opis
%SW0	Okres zadania głównego	Umożliwia zmianę okresu zadania głównego *Mast* zdefiniowanego podczas konfiguracji (program lub terminal). Czas trwania okresu wyrażony jest w ms (1..255ms). Dla zadania okresowego %SW0=0. Podczas zimnego startu słowo przyjmuje wartości określone w konfiguracji.
%SW1	Okres zadania szybkiego	Umożliwia zmianę okresu zadania szybkiego *Fast* zdefiniowanego podczas konfiguracji (program lub terminal). Czas trwania okresu wyrażony jest w ms (1..255ms). Dla zadania okresowego %SW0=0. Podczas zimnego startu słowo przyjmuje wartości określone w konfiguracji.
%SW8	Sterowanie odczytem wejść	Normalnie - 0. Wartość 1 lub 0 definiuje się przy pomocy programu lub terminala. Pozwala na pominięcie fazy odczytywania stanów wejść: %SW8:X0 1= pominięcie odczytu dla zadania głównego, %SW8:X1 1= pominięcie odczytu dla zadania szybkiego.
%SW9	Sterowanie aktualizacją wyjść	Normalnie - 0. Wartość 1 lub 0 definiuje się przy pomocy programu lub terminala. Pozwala na pominięcie fazy aktualizacji stanów wyjść: %SW8:X0 1= pominięcie fazy dla zadania głównego, %SW8:X1 1= pominięcie fazy dla zadania szybkiego.
%SW10	Pierwsze "przejście" zimnym starcie	Wartość 0 słowa dla bieżącego zadania oznacza, że wykonywane jest pierwsze "przejście" po zimnym starcie. %SW10:X0 : jest przypisane do zadania głównego *Mast*, %SW10:X1 : jest przypisane do zadania szybkiego *Fast*.
%SW11	Czas dla układu śledzącego	Odczytanie czasu ustawionego podczas konfiguracji dla układu śledzącego, wyrażonego w ms (10...500ms).
%SW12	Adres portu terminala sieci UNI-TELWAY	Adres portu terminala UNI-TELWAY (tryb *Slave*) zdefiniowany w konfiguracji i załadowany do tego słowa podczas zimnego startu.
%SW13	Główny adres stacji	Dla sieci głównej zawiera : • numer stacji (bajt mniej znaczący) od 0 do 127, • numer sieci (bajt bardziej znaczący) od 0 do 63 (pozycja przełącznika DIP karty PCMCIA).

Słowa systemowe	Funkcja	Opis
%SW17	Status błędu dla operacji zmiennoprzec.	Wystąpienie błędu podczas opercji zmiennoprzecinkowej, powoduje zmainę stanu bit %S18 na 1 i zapisanie kodu błędu w słowie %SW17: %SW17:X0 = niewłaściwa operacja / wynik nie jest liczbą, %SW17:X1 = niestandard. argument / wynik poprawny, %SW17:X2 = dzielenie przez 0 / wynik $\pm\infty$, %SW17:X3 = wartość za duża / wynik $\pm\infty$, %SW17:X4 = wartość za mała / wynik ± 0, %SW17:X5 = wynik jest nieprecyzyjny. Słowo to jest kasowane do 0 przez system podczas zimne-go startu i przez program w celu powtórnego użycia.
%SD18	Zegar czasu bezwzględnego	To słowo podwójne jest wykorzystywane do obliczania okre-su czasu. Jego wartość jest zwiększana co 1/10 sekundy przez system (nawet, gdy sterownik jest zatrzymany). Może ono być czytane i zapisywane przez program i terminal.
%SW20	Poziom aktywności *Grafcet-u*	Słowo to zawiera (dla bieżącego "przejścia") liczbę kroków aktywnych, do uaktywnienia i do dezaktywacji. Jest uaktu-alniane przez system przy każdej zmianie diagramu.
%SW21	Tablica odblokowanych przejść	To słowo zawiera (dla bieżącego "przejścia") liczbę bramek otwartych, do otwarcia i do zamknięcia. Jest uaktualniane przez system przy każdej zmianie diagramu.
%SW22 do %SW25	Tablica makr do skasowania	Każdemu elementowi tablicy %SW22:X0÷XM0...%SW25:X16÷XM63 przyporządkowana jest makrodefinicja. Makra, których bity w tablicy mają stan 0 zostaną skasowane, gdy bit %S24 przyjmie wartość 1.
%SW30	Czas "przejścia" zadania głównego MAST (1)	Zawiera czas trwania ostatniego "przejścia" zadania głów-nego MASTER (w ms).
%SW31	Maksymalny czas "przejścia" zadania głównego MAST (1)	Zawiera czas trwania najdłuższego "przejścia" zadania głównego od ostatniego zimnego startu (w ms).

(1) Ten czas jest równy czasowi upływającemu od momentu rozpoczęcia (odczytywanie stanów wejść) do zakończenia (aktualizacja stanów wyjść) cyklu "przejścia". Czas ten obejmuje przetwarzanie wyzwalane zdarzeniami oraz wykonywanie zadania szybkiego, jak również przetwarzanie żądań pochodzących z terminala.

Słowa systemowe	Funkcja	Opis
%SW32	Minimalny czas "przejścia" zadania głównego MAST (1)	Zawiera czas trwania najkrótszego "przejścia" zadania głównego od ostatniego zimnego startu (w ms).
%SW33	Czas "przejścia" zadania szybkiego FAST (1)	Zawiera czas trwania ostatniego "przejścia" zadania szybkiego FAST (w ms).
%SW34	Maksymalny czas "przejścia" zadania szybkiego FAST (1)	Zawiera czas trwania najdłuższego "przejścia" zadania szybkiego od ostatniego zimnego startu (w ms).
%SW35	Minimalny czas "przejścia" zadania szybkiego FAST (1)	Zawiera czas trwania najkrótszego "przejścia" zadania szybkiego od ostatniego zimnego startu (w ms).
%SW48	Liczba zdarzeń	Zawiera liczbę zdarzeń przetworzonych od ostatniego zimnego startu (w ms). Słowo to może być zapisywane przez program lub przez terminal.
%SW49 **%SW50** **%SW51** **%SW52** **%SW53**	Funkcja zegara czasu rzeczywistego (2)	Słowa systemowe zawierające bieżącą datę i czas w BCD: %SW49 : dzień tygodnia (1 - Poniedziałek do 7 - Niedziela). %SW50 : Sekundy (SS00) %SW51 : Godziny i Minuty (GGMM) %SW52 : Miesiąc i Dzień (MMDD) %SW53 : Rok (RRRR) Jeżeli bit %S50 ma wartość 0, to te słowa są kontrolowane przez system. Jeżeli bit %S50 ma wartość 1, to te słowa mogą być zapisywane przez program lub terminal.
%SW54 **%SW55** **%SW56** **%SW57** **%SW58**	Funkcja zegara czasu rzeczywistego (2)	Słowa systemowe zawierające datę i czas ostatniej awarii zasilania lub zatrzymania sterownika (w BCD): %SW54 : Sekundy (00SS), %SW55 : Godziny i Minuty (HHMM), %SW56 : Miesiąc i Dzień (MMDD), %SW57 : Rok (RRRR). %SW58 : bajt ważniejszy zawiera nazwę dnia tygodnia (1 oznacza Poniedziałek, a 7 - Niedzielę).

(1) Ten czas jest równy czasowi upływającemu od momentu rozpoczęcia (odczytywanie stanów wejść) do zakończenia (aktualizacja stanów wyjść) cyklu "przejścia". Czas ten obejmuje przetwarzanie wyzwalane zdarzeniami oraz wykonywanie zadania szybkiego, jak również przetwarzanie żądań pochodzących z terminala.

(2) Tylko dla sterowników TSX 37-21/22 i TSX 57.

Słowa systemowe	Funkcja	Opis
%SW58	Kod ostatniego zatrzymania	Bajt mniej znaczący zawiera kod przyczyny ostatniego stopu. 1= zmiana z trybu RUN na STOP na żądanie z terminala, 2= błąd programowy (przekroczenie dopuszczalnego czasu), 4= zanik zasilania, 5= błąd sprzętowy, 6= reakcja na instrukcję HALT.
%SW59	Korekta bieżącej daty	Zawiera dwa 8-bitowe zestawy parametrów bieżącej daty. Operacja nastawiania zegara jest zawsze wykonywana na zboczu rosnącym bitu. Bit %S59 uprawnia to słowo. **Zwiększenie** / **Zmniejszenie** / **Parametr** bit 0 / bit 8 / Dzień tygodnia bit 1 / bit 9 / Sekundy bit 2 / bit 10 / Minuty bit 3 / bit 11 / Godziny bit 4 / bit 12 / Dzień bit 5 / bit 13 / Miesiąc bit 6 / bit 14 / Rok bit 7 / bit 15 / Wiek
%SW66 (1)	Sterowanie wyświetlaczem 7-segment.	Zawiera kod Hexadecymalny/BCD wartości, którą użytkownik chce wyświetlić na dodatkowym 7-segmentowym wyświetlaczu. Pojawia się ona, gdy %S66 przyjmuje wartość 1.
%SW67 **%SW68** **%SW69**	Zarządzanie trybem "WORD"	Gdy %S69=1, to słowa te odblokowują wyświetlacz (płyta czołowa sterownika) umożliwiając pracę w trybie WORD: • %SW67 : sterowanie i status trybu WORD, • %SW68 : indeks maksymalny i bieżący, • %SW69 : nr pierwszego obiektu wyświetlanego na module. Więcej informacji na temat tego trybu pracy można znaleźć w części F, w rozdziale 1.5 (instrukcji instalacji).
%SW80	Zarządzanie komunikatami i telegramami	L-ba komunikatów wysłanych przez system do portu terminala.
%SW81		L-ba komunikatów odebranych przez system od portu terminala.
%SW82		L-ba komunikatów wysłanych przez system do karty PCMCIA.
%SW83		L-ba komunikatów wysłanych przez system od karty PCMCIA.
%SW84		Liczba telegramów wysłanych przez system.
%SW85		Liczba telegramów odebranych przez system.
%SW86		Liczba telegramów odrzuconych przez system.

(1) W aktualnej wersji nie jest wykorzystane.

Słowa systemowe	Funkcja	Opis
%SW96	Sterowanie/ diagnostyka zapisywania i odzysku	Funkcja zapisywania i odzyskiwania programu oraz %MW: bit 0 : żądanie zapisania. Ten bit jest aktywny na zboczu rosnącym. Jest kasowany do 0 przez system, po uwzględnieniu zbocza rosnącego. bit 1 : gdy ten bit ma wartość 1, to oznacza, że operacja zapisywania została zakończona. Jest kasowany do 0 przez system, po uwzględnieniu zbocza rosnącego. bit 2 : raport na temat operacji zapisu: 0 -> brak błędów podczas operacji, 1 -> błędy podczas operacji. bity 3 do 5 : zarezerwowane. bit 6 : informacja na temat kopii zapasowej programu (tak samo jak %S96). bit 7 : informacja o kopii zapasowej %MW (tak samo jak %S97). bity 8 do 15 : ten bajt jest znaczący tylko wtedy, kiedy bit raportu ma wartość 1 (bit 2 = 1, błąd podczas zapisywania). 1 -> liczba słów %MW przeznaczonych do zapisania jest większa od liczby zadeklarowanej, 2 -> liczba słów %MW przeznaczonych do zapisania jest większa od 1000 lub mniejsza od 0, 3 -> liczba słów %MW przeznaczonych do odzyskania jest większa od liczby zadeklarowanej, 4 -> aplikacja w pamięci wewnętrznej RAM jest większa niż 15 Ksłów (należy pamiętać o tym, że %MW są zawsze zapisywane, gdy zapisywany jest program, do wewnętrznej pamięci typu *Flash EPROM*), 5 -> operacja jest zabroniona podczas pracy w trybie RUN, 6 -> zasobnik dla kopii zapasowej jest zainstalowany, 7 -> błąd zapisu do pamięci *Flash EPROM*.
%SW97	Liczba %MW do zapisania	Umożliwia definiowanie liczby słów %MW przeznaczonych do zapisania. Jeżeli słowo to mieści się w przedziale do1 do 1000, to pierwszych 1000 słów %MW jest zapisywanych w wewnętrznej pamięci *Flash EPROM*. Kiedy to słowo ma wartość 0, to przenoszony do wewnętrznej pamięci *Flash EPROM* jest tylko program znajdujący się w wewnętrznej pamięci RAM. **Wszystkie zapisane słowa %MW są wtedy kasowane.** Jeżeli wewnętrzna pamięć *Flash EPROM* nie zawiera kopii zapasowej słów %MW, to podczas zimnego startu słowo to inicjowane jest z wartością -1. W innym przypadku słowo to jest inicjowane z wartością odpowiadającą liczbie zapisanych słów.
%SW98	Adres wejścia dyskretnego	Gdy bit %S98 = 1, to słowo to zawiera fizyczny adres (moduł/ kanał) wejścia dyskretnego zastępującego przycisk modułu TSX SAZ 10: Ważniejszy | Mniej ważny Numer modułu | Numer kanału

Słowa systemowe	Funkcja	Opis
%SW99	Adres wejścia dyskretnego	Gdy bit %S99 = 1, to słowo to zawiera fizyczny adres (moduł/kanał) wejścia dyskretnego zastępującego przycisk z modułu sterownika (znajdujący się w części sygnalizacyjnej modułu): Ważniejszy | Mniej ważny Numer modułu | Numer kanału
%SW108	Liczba bitów wymuszonych	Zawiera liczbę bitów o wymuszonych stanach w aplikacji. Normalnie - 0. Stan słowa jest uaktualniany przez system w pamięci.
%SW109	Licznik wymuszenia kanałów analogowych	Zawiera liczbę wymuszonych kanałów analogowych.
%SW116	Błąd FIPIO w zadaniu	Normalnie - 0. Każdy bit tego słowa reprezentuje status wymiany FIPIO dla zadania, w którym jest on testowany. **Słowo to jest kasowane do 0 przez użytkownika.** **%SW 116 :** x0 =1 błąd wymiany jawnej (zmienna nie jest wymieniana w sieci), x1 =1 przekroczenie czasu przeznaczonego dla wymiany jawnej (brak odpowiedzi po upływie dopuszczalnego czasu), x2 =1 max liczba jednoczesnych wymian jawnych, x3 =1 niewłaściwy status MPS (brak zawartości zmiennej), x4 =1 długość odebranej zmiennej jest większa od zadeklarowanej, x5 = zarezerwowany (stan 0), x6 =1 niewłaściwy kod PDU (zmienna musi zostać zignorowana), musi nastąpić inicjacja elementu x7 =1 przekroczenie czasu: czas przeznaczony dla elementu do wyprodukowania zmiennej został przekroczony, co oznacza brak w sieci zadeklarowanego urządzenia, x8 =1 błąd kanału x9 = zarezerwowany (stan 0), x10 to x14 = zarezerwowane (stan 0), x15 =1 błąd globalny (lub alternatywnie bity 3, 4, 6, 7, 8).
%SW124	Rodzaj błędu procesora	System zapisuje w tym słowie informację o ostatnim błędzie w działaniu procesora (zimny start nie zmienia tych kodów): 16#30 : błąd systemu 16#60 to 64 : przepełnienie stosu 16#90 : przerwanie pracy systemu: nie wykryto znaku IT 16#53 : przekroczenie czasu podczas wymiany I/O.

Słowa systemowe	Funkcja	Opis
%SW125	Rodzaj błędu blokującego	System zapisuje w tym słowie informację o rodzaju ostatniego błędu blokującego: 16#DB0 : przekroczenie nastaw układu śledzącego, 16#2258 : wykonanie instrukcji HALT, 16#DEF8 : wykonanie instrukcji JMP (skok) w stosunku do nie zdefiniowanej etykiety, 16#2XXX : wykonanie instrukcji CALL (wywołanie) nie zdefiniowanej procedury, 16#0XXX : wykonanie nieznanej funkcji, 16#DEFE : diagram *Grafcet* bez zdefiniowanego łącznika źródłowego lub docelowego, 16#DEFF : brak obiektów zmiennoprzecinkowych, 16#DEF0 : dzielenie przez 0, (1-->%S18) 16#DEF1 : błąd podczas transferu łańcucha znaków (1-->%S15) 16#DEF2 : przekroczenie pojemności, (1-->%S18) 16#DEF3 : przekroczenie indeksu (1-->%S20)
%SW126 **%SW127**	Adres instrukcji powodującej błąd	Adres instrukcji wywołującej błąd powodujący zablokowanie działania aplikacji (błąd blokujący). %SW126 zawiera adres względny %SW127 zawiera adres podstawowy
%SW128 do **%SW143**	Błąd podłączenia do sieci FIPIO	Każdy bit z grupy słów reprezentuje urządzenie podłączone do sieci FIPIO. Normalnie - 1. Jeżeli jeden z bitów przyjmuje wartość 0, to oznacza to błąd podłączenia dla danego urządzenia %SW128 reprezentuje adresy od 0 do 15 %SW128:X0 --> @0, %SW128:X1-->@1,....., %SW128:X15-->@15, %SW129 reprezentuje adresy od 16 do 31 %SW129:X0 --> @16, %SW129:X1-->@17,....., %SW129:X15-->@31, ... %SW143 reprezentuje adresy od 240 do 255 %SW143:X0 -->@240, %SW143:X1-->@241,....., %SW143:X15-->@255, System kasuje bit do 1, kiedy przyczyna błędu ustaje. Uwaga: @ = punkt podłączenia
%SW155	Liczba operacji wymiany jawnej	Liczba wykonanych operacji wymiany jawnej.

4.1 Wyszczególnienie

Wartości bezpośrednie *Immediate values*

Obiekty	PL7-2/3	PL7 Micro/Junior
Całkowite, dziesiętne	1234	1234
Całkowite, dwójkowe	L'10011110'	2#10011110
Całkowite, szesnastkowe	H'ABCD'	16#ABCD
Zmiennoprzecinkowe	-1.32e12 (PL7-3)	-1.32e12
Łańcuch znaków	M'aAbBcB'	'aAbBcC'

Etykiety *Labels*

Etykiety	Li i = 0 do 999	%Li i = 0 do 999

Bity

Obiekty	PL7-2/3	PL7 Micro/Junior
Bit wejścia modułu wewn. Indeksowany bit wejścia modułu wew. Bit wejścia zdalnego Indeksowany bit wejścia zdalnego	Ixy,i Ixy,i (Wj) (PL7-3) RIx,y,i (PL7-3) RIx,y,i (Wj) (PL7-3)	%Ixy.i %Ixy.i[%MWj] %I\<ścieżka>\<mod>.<kanał>
Bit wyjścia modułu wewn. Indeksowany bit wyjścia modułu wew. Bit wyjścia zdalnego Indeksowany bit wyjścia zdalnego	Oxy,i Oxy,i (Wj) (PL7-3) ROx,y,i (PL7-3) ROx,y,i (Wj) (PL7-3)	%Qxy.i %Qxy.i[%MWj] %Q\<ścieżka>\<mod>.<kanał>
Bit wewnętrznego błędu I/O • bit błędu modułu • bit błędu kanału	Ixy,S / Oxy,S	%Ixy.MOD.ERR %Ixy.i.ERR
Bit błędu I/O zdalnego • bit błędu modułu • bit błędu kanału • bit kanału wyjściowego • kasowanie bitu kanału wyjściowego	(PL7-3) RDx,y,i / ERRORx,y,i TRIPx,y,i RSTx,y,i	 %I\<ścieżka>\<mod>.MOD.ERR %I\<ścieżka>\<mod>.<kanał>.ERR
Bit wewnętrzny	Bi	%Mi
Indeksowany bit wewnętrzny	Bi(Wj) (PL7-3)	%Mi[%MWj]
Bit systemowy	SYi	%Si
Bit kroku	Xi	%Xi
Bit makrodefinicji	XMj (PL7-3)	%XMj
Bit kroku *i* makrodefinicji *j*	Xj,i (PL7-3)	%Xj.i
Bit kroku wejściowego makra *j*	Xj,I (PL7-3)	%Xj.IN
Bit kroku wyjściowego makra *j*	Xj,O (PL7-3)	%Xj.OUT
Bit *j* słowa wewnętrznego *i*	Wi,j	%MWi:Xj
Bit *j* indeksowanego słowa wewn. *i*	Wi(Wk),j(PL7-3)	%MWi[%MWk]:Xj

Bit *j* słowa stałego *i*	CWi,j	%KWi:Xj
Bit *j* indeks. słowa stałego *i*	CWi(Wk),j (PL7-3)	%KWi[%MWk]:Xj
Bit *j* rejestru *i*	I/OWxy,i,j	%IW/%QWxy.i:Xj
Bit *k* słowa wspólnego *j* stacji *i*	COMi,j,k COMXi,j,k (X = B, C, D)	%NWi.j:Xk %NXWi.j:Xk
Bit *j* słowa systemowego *i*	SWi,j	%SWi:Xj

Słowa

Obiekty	**PL7-2/3**	**PL7 Micro/Junior**
Słowo wewnętrzne pojedyncze	Wi	%MWi
Pojedyncze, indeks. słowo wewnętrzne	Wi(Wj) (PL7-3)	%MWi[%MWj]
Słowo wewnętrzne podwójne	DWi (PL7-3)	%MDi
Podwójne, indeks. słowo wewnętrzne	DWi(Wj) (PL7-3)	%MDi[%MWj]
Rzeczywiste słowo wewnętrzne		%MFi
Rzeczywiste, indeks. słowo wewnętrzne		%MFi[%MWj]
Słowo stałe pojedyncze	CWi	%KWi
Pojedyncze, indeks. słowo stałe	CWi(Wj)	%KWi[%MWj]
Podwójne słowo stałe	CDWi (PL7-3)	%KDi
Podwójne, indeks. słowo stałe	CDWi(Wj) (PL7-3)	%KDi[%MWj]
Rzeczywiste słowo stałe		%KFi
Rzeczywiste, indeks. słowo stałe		%KFi[%MWj]
Pojedyncze słowo wejściowe rejestru	IWxy,i	%IWxy.i
Podwójne słowo wejściowe rejestru		%IDxy.i
Pojedyncze słowo wyjściowe rejestru	OWxy,i	%QWxy.i
Podwójne słowo wyjściowe rejestru		%QDxy.i
Słowo wejściowe rejestru zdalnego	RIWx,y,i (PL7-3)	%IW\<ścieżka>\<mod>.<kanał>
Słowo wyjściowe rejestru zdalnego	ROWx,y,i (PL7-3)	%QW\<ścieżka>\<mod>.<kanał>
Słowo systemowe	SWi	%SWi
Słowo wspólne *j* stacji *i*	COMi,j COMXi,j (gdzie X = B, C, D)	%NW{i}j %NW{[r.]i}j r = nr sieci
Słowo statusu zdalnego modułu dyskretnego	STATUSAx,y,i (PL7-3) STATUSBx,y,i (PL7-3)	
Słowo statusu zdalnego kanału dyskretnego	STSx,y,i(PL7-3)	%IW\<ścieżka>\<mod>.<kanał>.ERR
Czas aktywności kroków *Grafcet-u*	Xi,V	%Xi.T
Czas aktywności kroku *i* makra *j*	Xj,i,V (PL7-3)	%Xj.i.T

Czas aktywności kroku wejściowego makra *j*	Xj,I,V (PL7-3)	%Xj.IN.T
Czas aktywności kroku wyjściowego makra *j*	Xj,O,V (PL7-3)	%Xj.OUT.T

Bloki funkcyjne

Obiekty	**PL7-2/3**	**PL7 Micro/Junior**
Zegar *Timer*	Ti	%Ti
• wartość nastawiona (słowo)	Ti,P	%Ti.P
• wartość bieżąca (słowo)	Ti,V	%Ti.V
• sygnalizacja pracy zegara (bit)	Ti,R	%Ti.R
• sygnalizacja końca pracy zegara (bit)	Ti,D	%Ti.D
Przerzutnik monostabilny *Monostable*	Mi	%MNi
• wartość nastawiona (słowo)	Mi,P	%MNi.P
• wartość bieżąca (słowo)	Mi,V	%MNi.V
• sygnalizacja pracy bloku (bit)	Mi,R	%MNi.R
Licznik dwukierunkowy *Up/down counter*	Ci	%Ci
• wartość nastawiona (słowo)	Ci,P	%Ci.P
• wartość bieżąca (słowo)	Ci,V	%Ci.V
• przekroczenie limitu górnego (bit)	Ci,E	%Ci.E
• osiągnięcie wartości nastawionej (bit)	Ci,D	%Ci.D
• przekroczenie limitu dolnego (bit)	Ci,F	%Ci.F
Rejestr *Register*	Ri	%Ri
• wejście (słowo)	Ri,I	%Ri.I
• wyjście (słowo)	Ri,O	%Ri.O
• rejest pełny (bit)	Ri,F	%Ri.F
• rejestr pusty (bit)	Ri,E	%Ri.E
Tekst	TXTi	brak bloku tekst.
Bęben *Drum controller*	Di (PL7-2)	%DRi
• numer aktywnego kroku (słowo)	Di,S	%DRi.S
• czas aktywności kroku bieżącego (słowo)	Di,V	%DRi.V
• 16 bitów komend (słowo)	Di,Wj	%DRi.Wj
• przetwarzanie ostatniego kroku (bit)	Di,F	%DRi.F
Szybki licznik *Fast Counter* / *Timer*	FC (PL7-2)	-
• wartość nastawiona (słowo)	FC,P	-
• wartość bieżąca (słowo)	FC,V	-
• zewnętrzne kasowanie (bit)	FC,E	-
• osiągnięcie wartości nastawionej (bit)	FC,D	-
• liczenie w toku (bit)	FC,F	-
Zegar czasu rzeczywistego *Real-time clock*	H (PL7-2)	-
• typ "WEEK" (tydzień) lub "YEAR" (rok) wybór dnia MTWTFSS (słowo)	VD	-
• początek czasu aktywności (słowo)	BGN	-
• koniec czasu aktywności (słowo)	END	-
• wartość bieżąca < limit (bit)	<	-
• wartość bieżąca = limit (bit)	=	-
• wartość bieżąca > limit (bit)	>	-

Tablice bitów i słów

Obiekty	PL7-2/3	PL7 Micro/Junior
Strumienie bitów		
• wewnętrzny łańcuch bitów	Bi[L]	%Mi:L
• łańcuch bitów wejściowych	Ixy,i[L] (PL7-3)	%Ixy.i:L
• łańcuch bitów wyjściowych	Oxy.i[L] (PL7-3)	%Qxy.i:L
• łańcuch bitów kroków *Grafcet*	Xi[L] (PL7-3)	%Xi:L
• łańcuch bitów makrodefinicji	XMi[L] (PL7-3)	
Łańcuchy znaków		%MBi:L (1) (gdzie *i* parzyste)
Tablice słów		
• tablica słów wewnętrznych	Wi[L]	%MWi:L
• tablica słów wewnętrznych indeks.	Wi(Wj)[L]	%MWi[%MWj]:L
• tablica podwójnych słów wewnętrznych	DWi[L] (PL7-3)	%MDi:L
• tablica podwójnych, indeks. słów wew.	DWi(Wj)[L] (PL7-3)	%MDi[%MWj]:L
• tablica słów stałych	CWi[L]	%KWi:L
• tablica słów stałych indeksowanych	CWi(Wj)[L]	%KWi[%MWj]:L
• tablica podwójnych słów stałych	CDWi[L] (PL7-3)	%KDi:L
• tablica podwójnych słów stałych indeks.	CDWi(Wj)[L] (PL7-3)	%KDi[%MWj]:L
• tablica obiektów rzeczywistych		%MFi:L
• tablica obiektów rzeczywistych indeks.		%MFi[%MWj]:L
• tablica rzeczywistych stałych		%KFi:L
• tablica rzeczywistych stałych indeks.		%KFi[%MWj]:L
• tablica zdalnych elementów wejściowych	RIx,y,i[L] (PL7-3)	
• tablica zdalnych elementów wyjściowych	ROx,y,i[L] (PL7-3)	
• tablica zdalnych elementów wej. indeks.	RIx,y,i(Wj)[L] (PL7-3)	
• tablica zdalnych elementów wyj. indeks.	ROx,y,i(Wj)[L] (PL7-3)	

Dodatkowe bloki funkcyjne

Obiekty	PL7-3	PL7 Micro/Junior
Dodatkowy blok funkcyjny	<OFB>i	
Element OFB	<OFB>i, <element>	
Indeksowany element OFB	<OFB>i, <element>(Wj)	
Element tablicy OFB	<OFB>i, <element>[L]	
Indeksowany element tablicy OFB	<OFB>i, <element>(Wj)[L]	

Instrukcje

Obiekty	PL7-2	PL7-3	PL7 Micro/Junior
Operacje na bitach - instukcje			
• Zaprzeczenie logiczne		NOT	NOT
• Koniunkcja AND	AND	•	AND
• Alternatywa OR	OR	+	OR
• Nierównoważność XOR	XOR		XOR
• Zbocze rosnące		RE	RE
• Zbocze opadające		FE	FE
• Nadanie wartości 1		SET	SET
• Skasowanie do 0		RESET	RESET
Operacje na słowach i słowach podwójnych - instrukcje			
• Dodawanie	+	+	+
• Odejmowanie	-	-	-
• Mnożenie	*	*	*
• Dzielenie	/	/	/
• Porównywanie	>, >=, <, <=, =, <>		>, >=, <, <=, =, <>
• Reszta z dzielenia	MOD	REM	REM
• Pierwiastek kwadratowy		SQRT	SQRT
• Wartość bezwzględna			ABS
• Koniunkcja AND	AND	AND	AND
• Alternatywa OR	OR	OR	OR
• Nierównoważność	XOR	XOR	XOR
• Logiczne dopełnienie	CPL	CPL	NOT
• Zwiększanie		INC	INC
• Zmniejszanie		DEC	DEC
• Logiczne przesunięcie w lewo		SHL	SHL
• Logiczne przesunięcie w prawo		SHR	SHR
• Okrężne przesunięcie w lewo	SLC	SLC	ROL
• Okrężne przesunięcie w prawo	SRC	SRC	ROR
Operacje zmiennoprzecinkowe (1)			
• Dodawanie		ADDF	+
• Odejmowanie		SUBF	-
• Mnożenie		MULF	*
• Dzielenie		DIVF	/
• Pierwiastek kwadratowy		SQRTF	SQRT
• Wartość bezwzględna			ABS
• Sprawdzenie równości		EQUF	=
• Porównywanie >		SUPF	>
• Porównywanie <		INFF	<
• Inne			>=, <=, <>

Instrukcje (kontynuacja)

Obiekty	PL7-2	PL7-3	PL7 Micro/Junior
Operacje na łańcuchach bajtów			
• Okrężne przesunięcie		SLCWORD	
Konwersje:			
• BCD - kod binarny	BCD	DTB	BCD_TO_INT
• Kod binarny - BCD	BIN	BTD	INT_TO_BCD
• ASCII - kod binarny	ATB	ATB	STRING_TO_INT lub STRING_TO_DINT
• Kod binarny - ASCII	BTA	BTA	INT_TO_STRING lub DINT_TO_STRING
• Kod Gray'a - kod binarny		GTB	GRAY_TO_INT
• Obiekt zmiennoprzecinkowy - całkowity	FTB	REAL_TO_INT lub	REAL_TO_DINT
• Obiekt całkowity - zmiennoprzecinkowy	FTF	INT_TO_REAL lub	DINT_TO_REAL
• BCD - obiekt zmiennoprzecinkowy		DTF	BCD_TO_REAL
• Obiekt zmiennoprzecinkowy - BCD		FTD	REAL_TO_BCD
• ASCII - obiekt zmiennoprzecinkowy		ATF	STRING_TO_REAL
• Obiekt zmiennoprzecinkowy - ASCII		FTA	REAL_TO_STRING
Operacje na tablicach			
• Operacje arytmetyczne		+, -, *, /, REM	+, -, *, /, REM
• Operacje logiczne		AND,OR,XO	RAND,OR,XOR,NOT
• Dodawanie słów w tablicy		+	SUM
• Wyszukiwanie 1-go słowa różniącego tablice		EQUAL	EQUAL
• Wyszukiwanie 1-go jednakowego słowa		SEARCH	FIND_EQU
Instrukcje programowe			
• Skok		JUMP Li	JUMP %Li
• Wywołanie procedury			CALL SRi SRi
• Powrót z procedury		RET	RETURN
• Zatrzymanie aplikacji		HALT	HALT
• Pętla warunkowa		IF/THEN/ELSE	IF/THEN/ELSE/END_IF
• Pętla iteracyjna		WHILE/DO	WHILE/DO/END_WHILE
Przerwania			
• Testowanie (odczytywanie stanów)		READINT	
• Maskowanie		MASKINT	MASKEVT
• Zdejmowanie maski		DMASKINT	UNMASKEVT
• Potwierdzenie		ACKINT	
• Generowanie znaku IT dla modułu		SETIT	
Operacje wymiany jawnej I/O			
• Odczytywanie stanów wejść dyskretnych		READBIT	
• Zapisywanie stanów wyjść dyskretnych		WRITEBIT	
• Odczytywanie rejestrów		READREG	
• Zapisywanie rejestrów		WRITEREG	
• Odczytywanie słów		READEXT	
• Zapisywanie słów		WRITEEXT	

Instrukcje (kontynuacja)

Obiekty	**PL7-3**	**PL7 Micro/Junior**
Operacje na blokach funkcyjnych		
• Nastawianie	PRESET Ti / Ci	PRESET %Ti / %Ci
• Uruchomienie	START Ti / Mi	START %Ti / %MNi •
Uaktywnienie zadań	START CTRLi	
• Kasowanie	RESET Ci / Ri / TXTi	RESET %Ci / %Ri
• Dezaktywacja zadań	RESET CTRLi	
• Zliczanie	UP Ci	UP %Ci
• Odliczanie	DOWN Ci	DOWN %Ci
• Zapisanie w rejestrze	PUT Ri	PUT %Ri
• Odczytanie z rejestru	GET Ri	GET %Ri
• Odebranie komunikatu	INPUT TXTi	
• Nadawanie komunikatu	OUTPUT TXTi	
• Nadanie/odbiór komunikatu	EXCHG TXTi	
• Wykonanie OFB	EXEC <OFBi>	
• Odczytywanie telegramów	READTLG	

Separatory

Obiekty	**PL7-2/3**	**PL7 Micro/Junior**
Przypisywanie	->	:=
Lewy margines dla indeksowania	(	[
Prawy margines dla indeksowania	)	]
Długość tablicy	[length]	:length

5.1 Słowa zarezerwowane

Wymienione poniżej słowa są zarezerwowane i nie mogą być używane jako symbole.

*_TO_* * = Litera
SRi
AUXi
EVTi
XMi
  i = liczba

ABS
ACCEPT
ACOS
ACTION
ACTIVATE_PULSE
ACTIVE_TIME
ADD
ADDRESS
ADD_DT
ADD_TOD
ADR
AND
ANDF
ANDN
ANDR
AND_ARX
ANY
ANY_BIT
ANY_DATE
ANY_INT
ANY_NUM
ANY_REAL
ARRAY
AR_D
AR_DINT
AR_F
AR_INT
AR_R
AR_W
AR_X
ASIN
ASK
ASK_MSG
ASK_VALUE
ASSIGN_KEYS
AT
ATAN
AUX
BCD_TO_INT
BIT_D
BIT_W
BLK
BLOCK
BODY
BOOL
BOTTOM
BTI
BTR
BY
BYTE
C
CAL
CALC
CALCN
CALL
CALL_COIL
CANCEL
CASE
CD
CHART
CH_M
CLK
CLOSE
CLOSED_CONTACT
COIL
COMMAND
COMMENTS
COMP4
COMPCH
CONCAT
CONF
CONFIGURATION
CONSTANT
CONTROL_LEDS
COPY_BIT
COS
CTD
CTU
CTUD
CU
D
DATE
DATE_AND_TIME
DAT_FMT
DAY_OF_WEEK
DA_TYPE
DEACTIVATE_PULSE
DEC
DELETE
DELTA_D
DELTA_DT
DELTA_TOD
DINT
DINT_TO_REAL
DINT_TO_STRING
DISPLAY_ALRM
DISPLAY_GRP
DISPLAY_MSG
DIV
DMOVE
DO
DOWN
DR
DRUM
DS
DSHL_RBIT
DSHRZ_C
DSHR_RBIT
DSORT_ARD
DSORT_ARW
DT
DTS
DWORD
D_BIT
E
EBOOL
ELSE
ELSIF
EMPTY
EMPTY_LINE
END
ENDC
ENDCN
END_ACTION
END_BLK
END_BLOCK
END_CASE
END_COMMENTS
END_CONFIGURATION
END_FOR
END_FUNCTION
END_FUNCTION_BLOCK
END_IF
END_MACRO_STEP
END_PAGE
END_PHRASE
END_PROG

END_PROGRAM
END_REPEAT
END_RESOURCE
END_RUNG
END_STEP
END_STRUCT
END_TRANSITION
END_TYPE
END_VAR
END_WHILE
EQ
EQUAL
EQUAL_ARR
ERR
EVT
EXCHG
EXCH_DATA
EXIT
EXP
EXPT
F
FALSE
FAST
FBD
FE
FIFO
FIND
FIND_EQ
FIND_EQD
FIND_EQDP
FIND_EQR
FIND_EQW
FIND_EQWP
FIND_GTD
FIND_GTR
FIND_GTW
FIND_LTD
FIND_LTR
FIND_LTW
FOR
FPULSOR
FROM
FTOF
FTON
FTP
FUNC
FUNCTION
FUNCTION_BLOCK
F_B
F_EDGE
F_TRIG
GE
GET
GET_MSG
GET_VALUE
GLOBAL_COMMENT
GR7
GRAY_TO_INT
GT
GTI
H
HALT
HALT_COIL
HASH_COIL
HW
H_COMPARE
H_LINK
I
IF
IL
IN
INC
INCJUMP
INDEX_CH
INFO
INITIAL_STEP
INIT_BUTTONS
INPUT
INPUT_CHAR
INSERT
INT
INTERVAL
INT_TO_BCD
INT_TO_REAL
INT_TO_STRING
ITB
ITS
JMP
JMPC
JMPCN
JUMP
JUMP_COIL
L
LAD
LANGAGE
LANGUAGE
LD
LDF
LDN
LDR
LE
LEFT
LEN
LENGTH_ARD
LENGTH_ARR
LENGTH_ARW
LENGTH_ARX
LIFO
LIMIT
LINT
LIST
LIT
LN
LOCATION
LOG
LREAL
LT
LW
LWORD
M
MACRO_STEP
MAIN
MASKEVT
MAST
MAX
MAX_ARD
MAX_ARR
MAX_ARW
MAX_PAGES
MAX_STEP
MCR
MCR_COIL
MCS
MCS_COIL
MID
MIN
MIN_ARD
MIN_ARR
MIN_ARW
MOD
MONO
MOVE
MPP
MPS
MRD
MS
MUL
MUX
M_CH
M_MACRO_STEP
N
N1
NAME
NB_ACTIVE_STEPS
NB_ACTIVE_TIME

NB_BLOCKS
NB_COMMON_WORDS
NB_CONSTANT_WORDS
NB_CPT
NB_DRUM
NB_INTERNAL_BITS
NB_INTERNAL_WORDS
NB_MACRO_STEPS
NB_MONO
NB_PAGES
NB_REG
NB_TIMER
NB_TM
NB_TRANSITIONS
NE
NIL
NO
NON_STORED
NOP
NOT
NOT_ARX
NOT_COIL
NOT_READABLE
NO_GR7
NO_PERIOD
N_CONTACT
O
OCCUR
OCCUR_ARD
OCCUR_ARR
OCCUR_ARW
OF
ON
OPEN
OPEN_CONTACT
OPERATE
OR
ORF
ORN
ORR
OR_ARX
OTHERS
OUT
OUTIN_CHAR
OUTPUT
OUT_BLK
P
P0
P1
PAGE
PAGE_COMMENT
PANEL_CMD
PERIOD
PHRASE
PHRASE_COMMENT
PID
PID_MMI
PLC
POST
PRESET
PRINT
PRINT_CHAR
PRIO0
PRIO1
PRIORITY
PRL
PROG
PROGRAM
PROG_LANGAGE
PROG_LANGUAGE
PT
PTC
PUT
PV
PWM
P_CONTACT
Q
QUERY
R
R1
RCV_TLG
RE
READ
READ_EVT_UTW
READ_ONLY
READ_PARAM
READ_STS
READ_VAR
READ_WRITE
REAL
REAL_TO_DINT
REAL_TO_INT
REAL_TO_STRING
REG
REM
REPEAT
REPLACE
REQ
RESET
RESET_COIL
RESOURCE
RESTORE_PARAM
RET
RETAIN
RETC
RETCN
RETURN
RET_COIL
RIGHT
ROL
ROLD
ROLW
ROL_ARD
ROL_ARR
ROL_ARW
ROL_DWORD
ROL_WORD
ROR
ROR_ARD
ROR_ARR
ROR_ARW
ROR_DWORD
ROR_WORD
RRTC
RS
RTB
RTC
RTS
RUNG
R_EDGE
R_TRIG
S
S1
SAVE
SAVE_PARAM
SCHEDULE
SD
SEARCH
SECTION
SEL
SEMA
SEND
SENDER
SEND_ALARM
SEND_MBX_ALARM
SEND_MBX_MSG
SEND_MSG
SEND_REQ
SEND_TLG
SERVO
SET
SET_COIL
SFC
SHIFT

SHL
SHOW_ALARM
SHOW_MSG
SHOW_PAGE
SHR
SHRZ
SIN
SINGLE
SINT
SL
SLCWORD
SMOVE
SOFT_CONFIGURATION
SORT
SORT_ARD
SORT_ARW
SQRT
SR
ST
STANDARD
START
STD
STEP
STI
STN
STOP
STR
STRING
STRING_TO_DINT
STRING_TO_INT
STRING_TO_REAL
STRUCT
SUB
SUB_DT
SUB_TOD
SUM
SU_TYPE
SWAP
S_T_AND_LINK
S_T_OR_LINK
T
TAN
TASK
TASKS
THEN
TIME
TIMER
TIME_OF_DAY
TM
TMAX
TMOVE
TO
TOD
TOF
TOFF
TON
TOP
TP
TRANSITION
TRANS_TIME
TRUE
TRUNC
TYPE
TYPES
T_S_AND_LINK
T_S_OR_LINK
U
UDINT
UINT
ULINT
UNMASKEVT
UNTIL
UP
USINT
USORT_ARD
USORT_ARW
UTIN_CHAR
VAR
VAR_ACCESS
VAR_EXTERNAL
VAR_GLOBAL
VAR_INPUT
VAR_IN_OUT
VAR_OUTPUT
VERSION
V_COMPARE
V_LINK
W
WHILE
WITH
WORD
WRITE
WRITE_CMD
WRITE_PARAM
WRITE_VAR
WRTC
WSHL_RBIT
WSHRZ_C
WSHR_RBIT
W_BIT
XM
XM_MONO
XM_MULTI
XOR
XORF
XORN
XORR
XOR_ARX
YES

6.1 Wprowadzenie

Norma IEC 1131-3 zatytułowana "PLCs - Part 3 : Programming languages" (*Sterowniki programowalne - Część 3: Języki programowania*) definiuje składnię oraz znaczenie elementów programowych służących do programowania sterowników.

Standard ten definiuje 2 języki tekstowe: IL (*Instruction List*) oraz ST (*Structured Text*), 2 języki graficzne: LD (*Ladder Diagram*) oraz FBD (*Function Block Diagram*) oraz mapę graficzną SFC (*Sequential Function Chart*), które służą do definiowania struktury wewnętrznej organizacji programowanej sekwencji.

Język PL7 umożliwia programowanie sterowników zgodnie z normą IEC: w jego skład wchodzą elementy języków opisanych w normie oraz funkcje rozszerzające jego możliwości, również spełniające warunki określone przez normę.

Norma IEC 1131-3 nie definiuje zasad rządzących interakcjami między elementami oprogramowania zastosowanymi przez producenta w celu spełnienia wymagań normy. Norma pozostawia w ten sposób większą dowolność w zakresie dostosowywania oprogramowania do wymagań użytkownika.

W tabeli poniżej, zestawiono elementy standardowe zastosowane w oprogramowaniu PL7, informację o zastosowaniach specjalnych oraz o wykrytych błędach.

6.1.1 Tabela zgodności

Ten system spełnia wymagania normy IEC 1131-3, pod warunkiem zachowania następujących obostrzeń dotyczących języka (charakterystyki):

Elementy wspólne

Nr tablicy	Nr charakterystyki	Opis
1	1	Wymagania dotyczące znaków: patrz paragraf 2.1.1 normy 1131-3
1	2	Małe znaki
1	3a	Znak: "numer" (#)

1	4a	Znak "dolara" ($)
1	5a	Kreska pionowa (|)
1	6a	Separatory indeksu dolnego: nawiasy "[]"
2	1	Duże litery i cyfry
2	2	Duże i małe litery, cyfry, podkreślenia
3	1	Komentarze
4	1	Literały całkowite (Uwaga 1)
4	2	Literały rzeczywiste (Uwaga 1)
4	3	Literały rzeczywiste z wykładnikiem
4	4	Literały dwójkowe (Uwaga 1)
4	6	Literały szesnastkowe (Uwaga 1)
4	7	Boolowskie Zero i Jeden
4	8	Boolowska PRAWDA i FAŁSZ
5	1	Parametry literowych łańcuchów znaków
6	2	$$ Znak "dolara"
6	3	$’ Pojedynczy cudzysłów
6	4	$L lub $l Przesunięcie o jedną linię LF
6	5	$N lub $n Nowa linia
6	6	$P lub $p Przesunięcie do następnej strony FF
6	7	$R lub $r Powrót karetki CR

6	8	$T lub $t Tabulator Tab
7	1a	Ciąg literałów z przedrostkiem t# (Uwaga 2)
10	1	Wartość logiczna BOOL -1 bit-
10	10	Wartość rzeczywista REAL -32 bity-
10	12	Czas TIME -32 bity- (Uwaga 3)
10	13	Data DATE -32 bity- (Uwaga 3)
10	14	Pora dnia TIME_OF_DAY -32 bity- (Uwaga 3)
10	15	Data i czas DATE_AND_TIME -64 bity- (Uwaga 3)
10	16	Łańcuch STRING
10	17	Bajt BYTE -8 bitów-
10	18	Słowo WORD -16 bitów-
10	19	Słowo podwójne DWORD -32 bity-
15	1	I przedrostek dla wejść
	2	Q przedrostek dla wyjść
	3	M przedrostek dla pamięci
15	4	X - przedrostek, pojedynczy bit
	5	Brak przedrostka - pojedynczy bit
	6	B - przedrostek, bajt (8 bitów)
	7	W - przedrostek, słowo (16 bitów)
	8	D - przedrostek, słowo podwójne (32 bity)

16	VAR VAR_INPUT VAR_OUTPUT VAR_IN_OUT VAR_EXTERNAL VAR_GLOBAL CONSTANT AT	Słowa kluczowe (Uwaga 4)
17	2	Deklaracja (adres) bezpośrednio reprezentowanych, zmiennych nieulotnych (Uwaga 4)
17	3	Deklaracja (symbol lub adres) slotów dla zmiennych symbolicznych (Uwaga 4)
17	5	Automatyczne przypisywanie pamięci zmiennym symbolicznym (zmienne bloków funkcyjnych) (Uwaga4)
18	2	Inicjacjacja (adres) bezpośrednio reprezentowanych, zmiennych nieulotnych (Uwaga 4)
18	3	Przypisywanie slotów i wartości początkowych zmiennym symbolicznym (symbol jako adres) (Uwaga 4)
18	5	Inicjacja zmiennych symbolicznych (zmienne bloków funkcyjnych) (Uwaga 4)
21	1	Funkcje PL7: ABS, EQUAL, ROL, ROR, SHL, SHR, SQRT, SUM
21	2	Ogólnie rzecz ujmując, funkcje języka PL7należą do tej kategorii
22	1	Funkcje związane z konwersją: DINT_TO_STRING, INT_TO_STRING, STRING_TO_DINT, STRING_TO_INT, DATE_TO_STRING, DT_TO_STRING, TIME_TO_STRING, TOD_TO_STRING, REAL_TO_STRING, STRING_TO_REAL, REAL_TO_INT, REAL_TO_DINT, INT_TO_REAL, DINT_TO_REAL (Uwaga 5)

22	2	Funkcja TRUNC: obcinanie do 0 typu REAL
22	3	Konwersja BCD_TO_INT (Uwaga 6)
22	4	Konwersja INT_TO_BCD (Uwaga 6)
23	1	Funkcja ABS: wartość bezwzględna
23	2	Funkcja SQRT: pierwiastek kwadratowy
23	3	Funkcja LN: logarytm naturalny
23	4	Funkcja LOG: logarytm dziesiętny
23	5	Funkcja EXP: funkcja z wykładnikiem naturalnym
23	6	Funkcja SIN: sinus (w radianach)
23	7	Funkcja COS: kosinus (w radianach)
23	8	Funkcja TAN: tangens (w radianach)
23	9	Funkcja ASIN: arc sin
23	10	Funkcja ACOS: arc cos
23	11	Funkcja ATAN: arc tan
25	1	Funkcja SHL: przesunięcie w lewo
25	2	Funkcja SHR: przesunięcie w prawo
25	3	Funkcja ROR: okrężne przesunięcie w prawo
25	4	Funkcja ROL: okrężne przesunięcie w lewo
29	1	Funkcja LEN: długość łańcucha
29	2	Funkcja LEFT: *n* znaków od lewej strony
29	3	Funkcja RIGHT: *n* znaków od prawej strony
29	4	Funkcja MID: *n* znaków od podanej pozycji
29	5	Funkcja CONCAT: łączenie (Uwaga 7)
29	6	Funkcja INSERT: wstawianie łańcucha do innego
29	7	Funkcja DELETE: kasowanie znaków

29	8	Funkcja REPLACE: zastępowanie znaków innymi
29	9	Funkcja FIND: szukanie ciągu znaków w łańcuchu
32	Input read Input write Out read Out write	Odczyt wejść (Uwaga 8) Zapis wejść Odczyt wyjść Zapis wyjść
33	1	Kwalifikacja do zapamiętania RETAIN zmiennych wewnętrznych bloków funkc. (Uwaga 9) (Uwaga 4)
33	2	Kwalifikacja do zapamiętania RETAIN wyjść bloków funkcyjnych (Uwaga 9) (Uwaga 4)
33	4a	Definiowanie I/O bloku funkcyjnego (język ST) (Uwaga 4)
37	1	Generowanie impulsu: TP (Uwaga 10)
37	2a	Opóźnienie typu *On-delay*: TON (Uwaga 10)
37	3a	Opóźnienie *Off-delay*: TOF (Uwaga 10)
38	diagramy czasowe	TP, TON, TOF
39	19	Użycie zmiennych bezpośrednio reprezentowanych (adres)
40	1	Krok, forma graficzna Uwaga: Numer kroku zastępujący identyfikator kroku
40	2	Krok, format teksowy stosowany tylko w formie źródłowej *Grafcet-u*
41	1	Warunek przejścia w języku ST
41	2	Warunek przejścia w języku LD
42	2l	Deklaracja akcji w języku LD
43	1	Blok akcji
43	2	Połączone bloki akcji

45	2	Kwalifikator akcji N (nie zapisany)
45	11	Kwalifikator akcji P1 (zbocze rosnące impulsu)
45	12	Kwalifikator akcji P0 (zbocze opadające impulsu)
46	1	Pojedyncza sekwencja krok/bramka
46	2c	Początek alternatywy: użytkownik musi tak definiować warunki żeby były wzajemnie się wykluczające
46	3	Koniec alternatywy
46	4	Początek koniunkcji, koniec koniunkcji
46	5c	Skok sekwencji na początek alternatywy
46	6c	Pętla: powrót do kroku poprzedniego
46	7	Strzałki Uwaga: Strzałki mogą być skierowane w dół lub w górę
48	40 41 42 43 45 46 57	Język *Grafcet* spełnia warunki minimalne zgodności z 1131-3 SFC Prezentacja graficzna
49	3	Konstrukcja RESOURCE...ON...END_RESOURCE
49	5a	Konstr. RESOURCE w zadaniu okresowym TASK
49	6a	Definicja programu PROGRAM wraz z przypisaniem PROGRAM-to-TASK
49	7	Definiowanie zmiennych bezpośrednio reprezentowanych w VAR_GLOBAL
50	5b	Uszeregowanie podzielne modelu wielozadaniowego

Uwaga 1:Podkreślanie znaków (_) pomiędzy cyframi literału nie jest dopuszczalne.

Uwaga 2: Te literały są tylko widoczne w aplikacji źródłowej, pokazują one czasy skonfigurowanych zadań.

Uwaga 3: Dane tego rodzaju nie są jeszcze wykorzystywane w sposób widoczny dla użytkownika. Ta tablica definiuje jednakże wykorzystanie pamięci dla ich wewnętrznej reprezentacji.

Uwaga 4: Te słowa kluczowe używane są tylko w kodach źródłowych generowanych przez programy do konwersji PL7 i PL7-2 oraz PL7-3.

Uwaga 5: Efekt ograniczonych możliwości konwersji:
DINT_TO_STRING: Jeżeli łańcuch docelowy mieści mniej niż 13 znaków, to wystepuje zjawisko obcięcia i zmiana wartości %S15.
INT_TO_STRING: Jeżeli łańcuch docelowy mieści mniej niż 7 znaków, to wystepuje zjawisko obcięcia i zmiana wartości %S15.
STRING_TO_DINT et STRING_TO_INT: Jeżeli łańcuch nie może byc poddany konwersji na wartość całkowitą, to wynik jest nieokreślony i zmienia się stan bitu %S18.
DATE_TO_STRING: Jeżeli łańcuch docelowy mieści mniej niż 11 znaków, to wystepuje zjawisko obcięcia i zmiana wartości %S15.
DT_TO_STRING: Jeżeli łańcuch docelowy mieści mniej niż 20 znaków, to występuje zjawisko obcięcia i zmiana wartości %S15.
TIME_TO_STRING: Jeżeli łańcuch docelowy mieści mniej niż 15 znaków, to występuje zjawisko obcięcia i zmiana wartości %S15.
TOD_TO_STRING: Jeżeli łańcuch docelowy mieści mniej niż 9 znaków, to występuje zjawisko obcięcia i zmiana wartości %S15.
REAL_TO_STRING: Jeżeli łańcuch docelowy mieści mniej niż 15 znaków, to występuje zjawisko obcięcia i zmiana wartości %S15.
STRING_TO_REAL: Jeżeli łańcuch nie może być poddany konwersji na wartość rzeczywistą, to wynikiem jest «1.#NAN» (16#FFC0_0000), a bit %S18 zmienia stan.
REAL_TO_INT: Jeżeli wartość rzeczywista po konwersji nie mieści się w zakresie od [-32768, +32767], to wynikiem jest -32768, a %S18 i %SW17:X0 zmieniają stan.
REAL_TO_DINT: Jeżeli wartość rzeczywista po konwersji nie mieści się w zakresie od [-2147483648, +2147483647], to wynikiem jest -2147483648, a %S18 i %SW17:X0 zmieniają stan.
INT_TO_REAL: Konwersja jest zawsze możliwa.
DINT_TO_REAL: Konwersja jest zawsze możliwa.

Uwaga 6: Ponieważ typ INT nie jest formalnie dopuszczony, choć jest stosowany, te funkcje umożliwiają zmianę formatu WORD.

Uwaga 7: Funkcja CONCAT jest ograniczona do łączenia 2 łańcuchów znaków.

Uwaga 8: Ten paragraf odnosi się do predefiniowanych bloków funkcyjnych PL7.

Uwaga 9: Kwalifikator RETAIN jest ukryty.

Uwaga 10: Zegary TP, TON, TOF pracują zgodnie z diagramem z tablicy 38 ale mają inny interfejs I/O w porównaniu z normą 1131-3.

Elementy języka IL (List)		
Nr tabeli	Nr ch-ki	Opis
51	Pola instrukcji	Etykieta, operator, argument, komentarz
52	1	LD
52	2	ST
52	3	S i R
52	4 6 7	AND OR XOR
52	18	JMP
52	20	RET
52	21	)
53	3	Wykorzystanie operatorów wejściowych dla uruchamiania bloków funkcyjnych w IL
54	11	IN (Uwaga 11)
54	12	IN (Uwaga 11)
54	13	IN (Uwaga 11)

Uwaga 11: Operator PT nie jest stosowany.

Elementy języka ST (Uwaga 12)		
Nr tabeli	Nr ch-ki	Opis
55	1	W nawiasie
55	2	Obliczanie funkcji
55	4	- Negacja
55	5	Logiczne dopełnienie NOT

55	6	* Mnożenie
	7	/ Dzielenie
55	9	+ Dodawanie
	10	- Odejmowanie
55	11	<, >, <=, >= Porównywanie
55	12	= Równość
55	13	<> Nierówność
55	15	Koniunkcja AND
55	16	Nierównoważność XOR
55	17	Alternatywa OR
56	1	:= Przypisanie
56	3	Powrót RETURN
56	4	Pętla IF "if... then... elsif... then... else... end_if"
56	6	Pętla FOR "for... to... do... end_for" (Uwaga 13)
56	7	Pętla WHILE "while... do... end_while"
56	8	Pętla REPEAT "repeat ... until... end_repeat"
56	9	Wyjście z pętli EXIT

Uwaga 12: Ten język jest stosowany w przypadku modułów ST. Podzbiór języka ST jest również stosowany w blokach operacyjnych OPERATE i porównywania COMPARE programowanych w językach *IL* oraz *LD*.

Uwaga 13: Zastosowanie pętli FOR z ukrytym krokiem równym 1.

Wspólne lementy graficzne

Nr tabeli	Nr ch-stytki	Opis
57	2	Graficzne linie pionowe
57	4	Graficzne linie poziome
57	6	Garficzne łączniki pionowe i poziome
57	8	Przecięcia linii bez połączeń
57	10	Węzły i załamania
57	12	Bloki z łącznikami graficznymi
58	2	Bezwarunkowy skok: język LD
58	4	Skok warunkowy: język LD
58	5	Powrót warunkowy: język LD
58	8	Powrót bezwarunkowy: język LD

Elementy języka LD

Nr tabeli	Nr ch-styki	Opis
59	1	Lewa linia bazowa (linia zasilająca)
59	2	Prawa linia bazowa (linia zasilająca)
60	1	Łącznik poziomy
60	2	Łącznik pionowy
61	1	Styk zwierny
61	3	Styk rozwierny
61	5	Bramka pozytywna
61	7	Bramka negatywna
62	1	Sprzężenie (cewka)proste
62	2	Sprzężenie odwrotne
62	3	Załączenie przerzutnika SET
62	4	Wyłączenie przerzutnika RESET

Parametry zależne od rodzaju zastosowania

Parametr	**Opis**
Procedura przetwarzania błędów	Błędy są zapisywane w słowach i bitach systemowych.
Znaki narodowe	ÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔ ÕÖØÙÚÛÜÝÞàáâãäåæçèéêëìí îïðñòóôõöøùúûüýþßÿ #, $, I
Maksymalna długość identyfikatorów	32
Maksymalna długość komentarza	222
Zakresy czasowe	(Uwaga 14)
Zakresy wartości zmiennych typu TIME	(Uwaga 14)
Dokładność odwzorowania sekund w zmiennych typu TIME_OF_DAY oraz DATE_AND_TIME	(Uwaga 15)
Maksymalna liczba indeksów siatki	1 (Uwaga 16)
Maksymalna liczba siatek	Zależnie od indeksowanego obszaru (Uwaga 16)
Dom. długość max. zmiennych STRING	Nie wykorzystane
Maksymalna, dopuszczalna długość zmiennych STRING	255
Maksymalna liczba poziomów	3
Mapowanie logiczne lub fizyczne	Mapowanie logiczne
Maksymalny zakres indeksów	Zależnie od indeksowanego obszaru (Uwaga 16)

Inicjacja wejść systemowych	Zmienne są inicjowane przez system: - z wartościami początkowymi o ile użytkownik je określił - z wartością 0, jeśli nie.
Wpływ rodzaju zastosowanej konwersji na dokładność	Patrz tabela 22, pozycja 1
Maksymalna liczba typów bloków i zastępczych bloków funkcjonalnych	Maksymalna liczba nie jest określona (ograniczenia wynikają z wielkości aplikacji)
Ograniczenia odnośnie rozmiarów programu	Maksymalny rozmiar kodu programu = 64 KB
Dokładność odmierzania czasu związanego z krokiem	100ms
Maksymalna liczba kroków na diagramie *Grafcet*	96 dla 3710 PLC 128 dla 3720 PLC 1024 dla 57xx V3.0 PLC
Maksymalna liczba bramek na diagramie i dla jednego kroku	1024 bramek na diagram 11 bramek na 1 krok
Mechanizm sterowania akcją	Kwalifikatory P0, P1 N1
Maksymalna liczba bloków akcji na krok	Można wykonać 3 akcje: przy aktywacji (P1), ciągła (N1) oraz przy dezaktywacji (P0)
Graficzna prezentacja statusu kroku	Krok aktywny wyświetlany jest w inwersji
Czas przejscia bramki (dezaktywacja kroków poprzedzających i uaktywnienie kroków następnych)	Czas przejścia zmienia się ale nigdy nie jest równy zeru.

Zasięg rozdzielenia ścieżki i jej zbiegu	Ograniczenie narzuca siatka
Lista sterowników, które można programować przy pomocy języka PL7	TSX 3710, 3720, TSX 5710, 5720, 5725, 5730, 5735, 5740, 57 45 PMX 5710, 5720, 5735, 5745, PCX 5710, 5735
Maksymalna liczba zadań	1 zadanie okresowe lub cykliczne 1 zadanie okresowe 8 zadań wyzwalanych dla 37 10 16 zadań wyzwalanych dla 37 20 32 zadania wyzwalane dla 57 10 64 zadania wyzwalane dla 57 20/30
Zakres przerwy między zadaniami	od 1 ms do 255 ms
Uszeregowanie podzielne, niepodzielne	Uszeregowanie podzielne
Maksymalna długość wyrażenia	Zmienna
Częściowe oszacowywanie wyrażeń logicznych	Nie
Maksymalna długość struktur sterujących w języku ST	Zmienna
Wartość zmiennej sterującej po wykonaniu pętli FOR	Wartość zmiennej sterującej równa się wartości limitu + 1 (ponieważ krok jest 1)
Reprezentacja graficzna/semi-graficzna	Reprezentacja graficzna
Restrykcje odnośnie topologii sieci	Sieć LD może zajmować maksymalnie 16 kolumn i 7 linii

Uwaga 14: Dane tego typu nie są jeszcze stosowane w sposób widoczny dla użytkownika. Ta tabela definiuje jednak zakresy ich wartości zgodnie z formatem IEC 1131-3.
TIME: od T#0 do T#429496729.5s
TIME_OF_DAY: od TOD#0:0:0 do TOD#23:59:59
DATE_AND_TIME: od DT#1990-01-01:0:0:0 do DT#2099-12-31:23:59:59
DATE: od D#1990-01-01 do D#2099-12-31

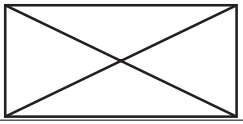
Uwaga 15: Zaokrąglanie odbywa się w następujący sposób: wartości od x.0 s do x.4 s są zaokrąglane do x s, a od x.5 s do x.9 s do wartości x+1 s.

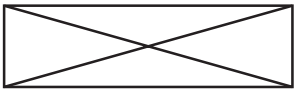
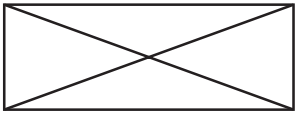
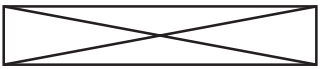
Uwaga 16: Istnieje możliwość indeksowania wszystkich typów danych reprezentujących bezpośrednio zmienne dodatnie lub ujemne w zakresie ograniczeń dotyczących ich maksymalnych wartości zdefiniowanych w konfiguracji.

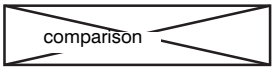
Błędy	**Opis**
Błędy konwersji	Sygnalizacja w czasie wykonywania za pomocą bitu system.- patrz tabela Wspólnych elementów: tabela 22, pozycja 1
Numeryczna wartość wyniku przekracza limit dla danych tego typu	Sygnalizacja w czasie wykonywania (bit systemowy %S18)
Brak podanej pozycji znaku	Sygnalizacja w czasie wykonywania (bit systemowy %S18)
Wynik przekracza dopuszczalną długość łańcucha	Sygnalizacja w czasie wykonywania (bit systemowy %S15)
Przekroczenie czasu podczas przejścia bramki	Sygn. podczas programowania
Program nie osiągnął linii końcowej	Sygnalizacja w czasie wykonywania (bit systemowy %S19)
Konflikty w planowaniu innych zadań	Sygn. podczas konfigurowania
Dzielenie przez 0 Niewłaściwy rodzaj danych dla operacji	Sygn. podczas programowania (jeśli to jest możliwe) lub w czasie wykonywania (bit systemowy %S18)
Pętla iteracyjna FOR lub WHILE nie osiągnęła końca (nie zamknęła się)	Błąd przekroczenia czasu (układ śledzący) - wskazanie procesora, którego to dotyczy

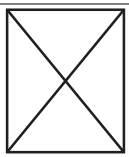
7.1 Słownik

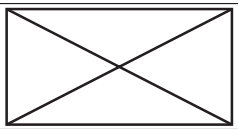
Instrukcje logiczne	LD	IL
Rejestr lub inicjacja stopnia labelki		`LD TRUE`

Testowanie (czytanie) bezpośrednie, odwrócone, zbocze rosnące, falling edge		`LD`
		`LDN`
		`LDR`
		`LDF`

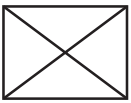
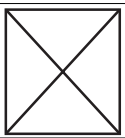
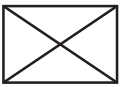
Koniunkcja AND		`AND ANDN ANDR ANDF` `AND( AND(N AND(RAND(F`
Alternatywa OR bezpośrednia, odwrócona, zbocze rosnące, zbocze opadające		`OR ORN ORR ORF` `OR( OR(N OR(R OR(F`
Negacja		`N`
Nierównoważność XOR (bezpośrednia, odwrócona, zbocze rosnące, opad.)		`XOR XORN` `XORR XORF`
Zapisywanie (bezpośrednie, odwrócone)		`ST` `STN`
Nadanie wartości 1 Nadanie wartości 0		`S` `R`
Blok operacyjny (zawartość - patrz następne strony)		`[akcja]`

Blok porównywania prosty (zawartość - patrz następne strony)		`LD [porównywannie]`
		`AND [porównywannie]`
		`AND( [porównywannie]`
		`OR [porównywannie]`
		`OR( [porównywannie]`
		`XOR [porównywannie]`

Blok porównywania złożony		

Funkcja *Memory PusH*
Funkcja *Memory ReaD*
Funkcja *Memory PoP*
		`MPS`
		`MRD`
		`MPP`

Instrukcje	ST
Przypisanie	:=
Alternatywa OR Koniunkcja AND Nierównoważność OR Negacja Zbocze: rosnące, opadające Nadanie: wartości 1, wartości 0	OR AND XOR NOT RE, FE SET, RESET

Function blocks	LD	IL
Zegar IEC		IN BLK..END_BLK struktura
Zegar PL7-3		
Licznik dwukierunkowy		R S CU CD BLK..END_BLK struktura
Przerzutnik monostabilny		S BLK..END_BLK struktura
Rejestr		R I O BLK..END_BLK struktura
Bęben		R U BLK..END_BLK struktura

Bloki funkcyjne	ST
Zegar IEC	START %TMi DOWN %TMi
Zegar PL7-3	PRESET %Ti START %Ti STOP %Ti
Licznik dwukierunkowy	RESET %Ci PRESET %Ci UP %Ci, DOWN %Ci
Przerzutnik monostabilny	START %MNi
Rejestr	RESET %Ri PUT %Ri GET %Ri
Bęben	RESET %DRi UP %DRi

Struktura sterująca	ST
Akcja warunkowa Iteracyjna akcja warunkowa Iteracyjna akcja warunkowa	IF...THEN... ELSIF...THEN... ELSE...END_IF; WHILE...DO...END_WHILE; REPEAT...UNTIL...END_REPEAT;
Akcja wielokrotnie powtarzana	FOR...DO...END_FOR;
Instrukcja wyjścia z pętli	EXIT
Operacje arytmetyczne na obiektach całkowitych (pojedynczych i podwójnych)	**LD/IL/ST**
Transfer lub inicjacja	:=
Porównywanie	= <> <= " >=
Dodawanie, odejmowanie, mnożenie, dzielenie, reszta	+ - * / REM
Koniunkcja, alternatywa, nierówn., negacja	AND OR XOR NOT
Wartość bezwzględna, pierwiasek kwadratowy	ABS, SQRT
Zwiększanie Zmniejszanie	INC DEC
Przesunięcie w lewo Przesunięcie w prawo Okrężne przesunięcie w lewo Okrężne przesunięcie w prawo	SHL SHR ROL ROR
Arytmetyczne operacje zmiennoprzecink.	**LD/IL/ST**
Transfer lub inicjacja	:=
Porównywanie	= <> <= " >=
Dodawanie, odejmowanie, mnożenie, dzielenie, część całkowita	+ - * / TRUNC
Wartość bezwzględna, pierwiasek kwadratowy Logarytm, funkcja wykładnicza Sinus, kosinus, tangens Arc sin, arc cos, arc tg Konwersja stopnie <--> radiany	ABS, SQRT LOG, LN, EXPT, EXP SIN, COS, TAN ASIN, ACOS, ATAN DEG_TO_RAD, RAD_TO_DEG

Konwersje numeryczne	LD/IL/ST
Konwersja BCD - liczby całkowite, pojedyncze	BCD_TO_INT
Konwersja GRAY - liczby całkowite, pojedyncze	GRAY_TO_INT
Konwersja liczby całkowite, pojedyncze - BCD	INT_TO_BCD
Konwersja liczby całkowite, pojed. - zmiennoprzecink.	INT_TO_REAL
Konwersja liczby całkowite, podw. - zmiennoprzecink.	DINT_TO_REAL
Konwersja liczby zmiennoprzecink. - całkowite, pojed.	REAL_TO_INT
Konwersja liczby zmiennoprzecink. - całkowite, podw.	REAL_TO_DINT
Konwersja 32-bit. BCD - 32-bit. całkowite	DBCD_TO_DINT
Konwersja 32-bit. całkowite - 32-bit. BCD	DINT_TO_DBCD
Konwersja 32-bit. BCD - 16-bit. całkowite	DBCD_TO_INT
Konwersja 16-bit. całkowite - 32-bit. BCD	INT_TO_DBCD
Wydzielania słowa mniej ważnego ze słowa podwójnego	LW
Wydzielania słowa ważniejszego ze słowa podwójnego	HW
Łączenie 2 słów pojedynczych	CONCATW

Tablice bitowe	LD/IL/ST
Transfer lub inicjacja	:=
Kopiowanie tablicy bitowej do innej tablicy bitowej	COPY_BIT
Koniunkcja między dwiema tablicami	AND_ARX
Alternatywa między dwiema tablicami	OR_ARX
Nierównoważność między dwiema tablicami	XOR_ARX
Negacja tablicy	NOT_ARX
Kopiowanie tablicy bitowej do tablicy słów	BIT_W
Kopiowanie tablicy bitowej do tablicy słów podwójnych	BIT_D
Kopiowanie tablicy słów do tablicy bitowej	W_BIT
Kopiowanie tablicy słów podwójnych do tablicy bitowej	D_BIT
Obliczanie długości tablicy	LENGTH_ARX
Operacje na tablicach	**LD/IL/ST**
Transfer lub inicjacja	:=
Operacje arytmetyczne między tablicami	+ - * / REM
Operacje logiczne między tablicami	AND OR XOR
Operacje arytmetyczne między tablicą a liczbą całkowitą	+ - * / REM
Operacje logiczne między tablicą a liczbą całkowitą	AND OR XOR
Dopełnienie elementów tablicy	NOT
Sumowanie elementów tablicy	SUM
Porównywanie dwu tablic	EQUAL
Szukanie 1-go elementu równego podanej wartości	FIND_EQW, FIND_EQD
Szukanie 1-go elementu równego wartości z danego rzędu	FIND_EQWP, FIND_EQDP
Szukanie 1-go elementu większego od podanej wartości	FIND_GTW, FIND_GTD
Szukanie 1-go elementu mniejszego od podanej wartości	FIND_LTW, FIND_LTD
Szukanie największej wartości w tablicy	MAX_ARW, MAX_ARD
Szukanie najmniejszej wartości w tablicy	MIN_ARW, MIN_ARD
Liczba wystąpień danej wartości w tablicy	OCCUR_ARW, OCCUR_ARD
Okrężne przesunięcie w lewo w tablicy	ROL_ARW, ROL_ARD
Okrężne przesunięcie w prawo w tablicy	ROR_ARW, ROL_ARW
Sortowanie tablicy (w porządku rosnącym lub malejącym)	SORT_ARW, SORT_ARD
Obliczanie długości tablicy	LENGTH_ARW, LENGTH_ARD
Operacje na tablicach zmiennoprzecinkowych	**LD/IL/ST**
Transfer lub inicjacja	:=
Sumowanie elementów tablicy	SUM_ARR
Porównywanie dwu tablic	EQUAL_ARR
Szukanie 1-go elementu równego podanej wartości	FIND_EQR
Szukanie 1-go elementu większego od podanej wartości	FIND_GTR
Szukanie 1-go elementu mniejszego od podanej wartości	FIND_LTR
Szukanie największej wartości w tablicy	MAX_ARR
Szukanie najmniejszej wartości w tablicy	MIN_ARR
Liczba wystąpień danej wartości w tablicy	OCCUR_ARR
Okrężne przesunięcie w lewo w tablicy	ROL_ARR
Okrężne przesunięcie w prawo w tablicy	ROR_ARR
Sortowanie tablicy (w porządku rosnącym lub malejącym)	SORT_ARR
Obliczanie długości tablicy	LENGHT_ARR

Operacje typu «Orphee»	LD/IL/ST
Przesunięcie w słowie, w lewo, z odzyskaniem bitów	WSHL_RBIT, DSHL_RBIT
Przesunięcie w słowie, w prawo z dodaniem znaku i odzyskaniem przesuniętych bitów	WSHR_RBIT, DSHR_RBIT
Przesunięcie w słowie, w prawo z uzupełnieniem pustych miejsc 0 i odzyskaniem przesuniętych bitów	WSHRZ_C, DSHRZ_C
Liczenie w dwóch kierunkach (w górę i w dół) z sygnalizacją przekroczenia limitu	SCOUNT
Okrężne przesunięcie w lewo	ROLW, ROLD
Okrężne przesunięcie w prawo	RORW, RORD
Wymiana jawna	**LD/IL/ST**
Czytanie parametrów kanału logicznego %M	READ_PARAM
Czytanie statusu %M kanału logicznego	READ_STS
Odzyskanie parametrów %M kanału logicznego	RESTORE_PARAM
Zachowanie parametrów %M kanału logicznego	SAVE_PARAM
Zapisanie sterowania %M kanału logicznego	WRITE_CMD
Zapisanie parametrów %M kanału logicznego	WRITE_PARAM
Operacje związane z zarządzaniem czasem	**LD/IL/ST**
Zegar czasu rzeczywistego	SCHEDULE
Porównywanie	= <> <= " >=
Transfer	:=
Odczytanie daty i kodu ostatniego "stopu" sterownika	PTC
Odczytanie daty systemowej	RRTC
Uaktualnienie daty systemowej	WRTC
Dodanie czasu do pełnej daty	ADD_DT
Dodanie czasu do pory dnia	ADD_TOD
Konwersja daty na łańcuch znaków	DATE_TO_STRING
Dzień tygodnia	DAY_OF_WEEK
Różnica między dwiema datami	DELTA_D
Różnica pomiędzy dwiema pełnymi datami	DELTA_DT
Różnica między dwiema porami dnia	DELTA_TOD
Konwersja pełnej daty na łańcuch znaków	DT_TO_STRING
Odjęcie czasu od pełnej daty	SUB_DT
Odjęcie czasu od pory dnia	SUB_TOD
Konwersja okresu czasu na łańcuch znaków	TIME_TO_STRING
Konwersja pory dnia na łańcuch znaków	TOD_TO_STRING
Przeliczenie okresu czasu na format godz.-min-sek	TRANS_TIME
Operacje opóźniające	**LD/IL/ST**
Opóźnienie typu *On-delay*	FTON
Opóźnienie typu *Off-delay*	FTOF
Generowanie impulsu o określonym czasie trwania	FTP
Generator przebiegu prostokątnego	FPULSOR

Operacje na łańcuchach znaków	LD/IL/ST
Porównywanie	= <> <= " >=
Transfer	:=
Konwersja podwójnej liczby całkowitej na łańcuch Konwersja pojedynczej liczby całkowitej na łańcuch Konwersja łańcucha na podwójną liczbę całkowitą Konwersja łańcucha na pojedynczą liczbę całkowitą Konwersja łańcucha na wartość zmiennoprzecinkową Konwersja wartości zmiennoprzecinkowej na łańcuch	DINT_TO_STRING INT_TO_STRING STRING_TO_DINT STRING_TO_INT STRING_TO_REAL REAL_TO_STRING
Łączenie dwóch łańcuchów Usunięcie ciągu znaków z łańcucha Znalezienie pierwszego innego znaku w łańcuchu Wyszukiwanie ciągu znaków w łańcuchu Wstawianie ciągu znaków do łańcucha Wydzielenie lewej części łańcucha Długość łańcucha Wydzielenie ciągu znaków z łańcucha Zastąpienie ciągu znaków w łańcuchu Wydzielenie prawej części łańcucha	CONCAT DELETE EQUAL_STR FIND INSERT LEFT LEN MID REPLACE RIGHT

Maskowanie zdarzeń	LD/IL/ST
Uaktywnienie / dezaktywacja zdarzeń Korekta czasu "przejścia" zadania	%Si pozycja %SWi pozycja
Globalne maskowanie zdarzeń Zdjęcie maski ze wszystkich zdarzeń	MASKEVT UNMASKEVT

Komunikacja	LD/IL/ST
Żądanie zatrzymania wykonywanej funkcji Wysyłanie i/lub odbieranie danych Żądanie czytania łańcucha znaków Żądanie przesłania i/lub czytania łańcucha znaków Przesłanie łańcucha znaków Odebranie telegramu Czytanie podstawowych obiektów językowych Przesyłanie / odbiór żądań UNI-TE Wysłanie telegramu Zapisywanie podstawowych obiektów językowych Okrężne przesunięcie w prawo o jeden bajt w tablicy Zamiana miejscami bajtów w tablicy słów Czytanie danych wspólnych Modbus+ Zapisywanie danych wspólnych Modbus+	CANCEL DATA_EXCH INPUT_CHAR OUT_IN_CHAR PRINT_CHAR RCV_TLG READ_VAR SEND_REQ SEND_TLG WRITE_VAR ROR1_ARB SWAP READ_GDATA WRITE_GDATA

8.1 Wprowadzenie

Niniejszy rozdział zawiera informacje umożliwiające obliczenie następujących parametrów dla programu napisanego dla sterowników TSX 37/57:

- czas wykonywania programu aplikacji,
- rozmiar pamięci dla programu aplikacji.

Czas wykonywania programu

Czas wykonywania programu oblicza się przy pomocy tabel zamieszczonych w niniejszym rozdziale poprzez sumowanie czasów przypadających na wykonywanie poszczególnych instrukcji.

Uwaga: Obliczony w ten sposób czas jest czasem maksymalnym. W rzeczywistości, blok operacyjny, czy też procedura zostaną wykonane tylko, gdy zostaną spełnione przyporządkowane im warunki (równanie logiczne, od którego wyniku zależy wykonanie procedury lub operacji). Może się tak przydarzyć, że faktyczny czas wykonania programu będzie dużo mniejszy od czasu obliczonego.

Obliczenie czasu trwania całego cyklu wymaga uwzględnienia parametrów specyficznych dla danego sterownika (czas trwania wykonywanych operacji pomocniczych, czas trwania wymiany I/O, itp.). Aby możliwe było obliczenie czasu całkowitego dla pracy sterownika niezbędne jest odczytanie tych parametrów z instrukcji obsługi odpowiedniego sterownika (rozdziała zawierający opis parametrów technicznych sterownika).

Rozmiar pamięci dla aplikacji

Rozmiar pamięci zajmowanej przez aplikację jest sumą następujących elementów:

Element	Sposób obliczania
• Program zamieszczone w rozdziałach	Należy zsumować wszystkie instrukcje (patrz tabele 8.2 and 8.3) i pomnożyć je przez współczynnik odpowiedni dla danego języka (patrz następna strona)
• Funkcje złożone	Patrz rozdział 8.4.4
• Skonfigurowane obiekty PL7	Patrz rozdział 8.4.2
• Skonfigurowany moduł I/O	Patrz rozdział 8.4.3

Na kolejnych stronach niniejszego rozdziału zamieszczono tabele zawierające informacje o rozmiarach obszarów pamięci odpowiadających danym kodom instrukcji. W celu określenia całkowitego rozmiaru zajętej pamięci dla instrukcji lub programu należy pomnożyć odczytane wartości przez współczynnik uwzględniający rozmiar informacji typowej dla zastosowanego języka (na przykład, informacje graficzne w przypadku języka *Ladder*).

- Język *Ladder*: Rozmiar całkowity = 1.7 x Rozmiar dla kodu

- Język *ST*: Rozmiar całkowity = 1.6 x Rozmiar dla kodu

- Język *IL*:
  - dla TSX37: Rozmiar całkowity = 1.4 x Rozmiar dla kodu
  - dla TSX57: Rozmiar całkowity = 1.6 x Rozmiar dla kodu

- Język *Grafcet*:
  Rozmiar diagramu oblicza się następująco:
  Rozmiar diagramu (w ilości słów) = 214 + 17 * liczba kroków diagramu + 2 * całkowita liczba skonfigurowanych kroków + 4 * liczba zaprogramowanych akcji

Uwaga: Komentarz programu zajmuje 1 bajt dla jednego znaku.

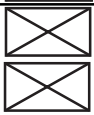
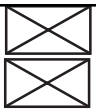
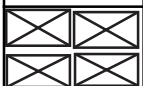
Komentarz

Liczby zamieszczone w tabelach są wartościami średnimi, oszacowanymi dla typowych aplikacji. Nie ma możliwości dokładnego obliczenia rozmiaru zajętej pamięci, ponieważ PL7 optymalizuje wykorzystanie pamięci uwzględniając zawartość i strukturę aplikacji.

W rozdziale 8.4.1 opisano różne obszary pamięci, które są wykorzystywane przez aplikację.

8.2 Sterownik TSX 37

8.2.1 Instrukcje logiczne

LD	IL	ST	Obiekty	Czas wykonywania (s)			Rozmiar (słowa)
				3710	3720 RAM	3720 Zasobnik	37xx
				0.25	0.13	0.19	1
	LD, LDN		%M1 (1)	0.25	0.13	0.19	1
			%M1[%MW2]	13.10	12.85	12.85	7
			%MW0:X0 (2)	6.06	5.75	5.75	4
			%IWi.j:Xk (3)	77.04	69.25	69.25	8
			%MW0[%MW10]:X0	16.29	15.55	15.55	8
			%KW0[%MW10]:X0	87.27	79.05	79.05	12
	LDR, LDF		%M1	0.50	0.25	0.38	2
			%M1[%MW2]	13.01	12.75	12.75	7
	AND, ANDN , AND (, AND (N , idem OR			idem LD, LDN			
	ANDR, ANDF, AND (R, AND (F, idem OR			idem LDR LDF			
	XOR, XORN		%M1	1.25	0.63	0.94	5
			%M1[%MW2]	26.94	26.08	26.26	13
			%MW0:X0	12.86	11.88	12.06	10
			%IWi.j:Xk	83.84	75.38	75.56	14
			%MW0[%MW10]:X0	33.33	31.48	31.66	14
			%KW0[%MW10]:X0	104.31	94.98 1	95.16	18
	XORR, XORF		%M1	2.25	1.13	1.69	9
			%M1[%MW2]	27.28	26.13	26.44	19
	ST, STN, S, R		%M1	0.50	0.25	0.38	2
			%M1[%MW2]	13.10	12.85	12.85	7
			%MW0:X0	5.88	5.60	5.60	4
			%NW{i}j:Xk (3)	76.86	69.10	69.10	8
			%MW0[%MW10]:X0	16.41	15.65	15.65	8
sprzężenie wielokrotne w *Ladder*, «koszt» 2-giego sprzężenia i następnych				0.25	0.13	0.19	1
blok operacyjny	[akcja]		blok wykonywany	0.74	0.75	0.75	1
			nie wykonywany	5.55	5.40	5.40	1

(1) Obejmuje wszystkie obiekty, co do których można stosować wymuszenie: %I,%Q,%X,%M,%S

(2) Inne obiekty tego typu: bity wyjściowe bloku funkcyjnego %TMi.Q ..., bity wydzielone ze słów systemowych %SWi:Xj

(3) Inne obiekty tego typu: bity wydzielone ze słów wspólnych %NW{i}j:Xk, bity wydzielone ze słów I/O %IWi.j.Xk, %QWi.j.Xk, bity wydzielone ze słów %KW, bity błędów %Ii.j.ERR

LD	IL	ST	Obiekty	Czas wykonywania (s)			Rozmiar (słowa)
				3710	3720 RAM	3720 Zasobnik	37xx
blok porównywania poziomy	LD [porównywanie]		czas dodawany do operacji porównywania	0.00	0.00	0.00	0
blok porównywania pionowy			pomiędzy 2 %MWi	12.38	11.85	11.85	4
zamknięcie (zbieg)	)	)		0.25	0.13	0.19	1
otwarcie nie poprzedzone zamknięciem			ladder, 1 otwarcie	0.25	0.13	0.19	1
	MPS, MPP, MRD		lista MPS+MPP	0.75	0.38	0.56	3
			lista MRD	0.25	0.13	0.19	1

8.2.2 Bloki funkcyjne

LD	IL	ST	Obiekty/warunki	Czas wykonywania (s)		Rozmiar (słowa)
				3710	3720	37xx

Zegar IE

zbocze rosnące na IN	IN %TM1 (zb. rosnące)	START %TM1	start zegara	43.39	41.11	3
zbocze opadające na IN	IN %TM1 (zbocze opadające)	DOWN %TM1	zatrzymanie zegara	17.47	17.01	
IN =1	IN %TM1 (=1)		zegar liczy (*on*)	18.74	17.99	
IN =0	IN %TM1 (=0)		zegar nie liczy (*off*)	17.40	16.67	

Zegar PL7-3

		START %T1	uaktywnienie			3
		STOP %T1	zamrożenie	12.63	12.15	
E=0		RESET %T1	kasowanie	12.94	12.15	
			zegar liczy (*on*)	17.55	17.00	
			zegar nie liczy (*off*)			

Licznik dwukierunkowy

kasowanie, R=1	R %C8 (=1)	RESET %C8	kasowanie	18.69	17.92	3
nastawianie, S=1	S %C9 (=1)	PRESET %C9	nastawianie	20.42	19.73	
zbocze rosnące na CU	CU %C8 (zb. rosnące)	UP %C8	zliczanie *up*	19.92	19.10	
zbocze rosnące na CD	CD %C9 (zb. rosnące)	DOWN %C9	odliczanie *down*	19.92	19.10	
wejścia nie aktywne	R/S/CU/CD bit nie aktywny		brak akcji	13.27	12.81	

Bloki funkcyjne (ciąg dalszy)

LD	IL	ST	Obiekty/warunki	Czas wykonywania (s)		Rozmiar (słowa)
				3710	**3720**	**37xx**

Przerzutnik monostabilny

zbocze rosnące na S	S %MN0, zb. rosnące	START %MN0	start	35.08	33.16	3
S=1	S %MN0, S=1/0		przerzutnik aktywny	11.64	11.17	

Rejestr

zbocze na I	I %R2(zbocze)	PUT %R2	zapisywanie	21.90	21.27	3
zbocze na O	O%R2(zbocze)	GET %R2	odczytywanie	21.90	21.27	
R=1	R %R1 (=1)	RESET %R2	kasowanie	16.90	16.02	
wejścia nie aktywne	I/O/R, bit nie aktywny		brak akcji	12.61	12.19	

Bęben

zbocze na U	U %DR0	UP %DR1	do przodu, ustalony	181.37	169.13	3
			na bit sterujący	19.30	19.30	
R=1	R %DR1	RESET %DR2	kasowanie, stałe	174.15	162.03	
			na bit sterujący	19.30	19.30	
wejścia nie aktywne	R/U, bit nie akt.		brak akcji, stały	175.92	164.00	
			na bit sterujący	19.30	19.30	

8.2.3 Operacje arytmetyczne - obiekty całkowite i zmiennoprzecinkowe

Współczynniki korekcyjne uwzględniające typ obiektu

Czas trwania i rozmiary podane na kolejnych stronach dotyczą %MW0, %MD0 lub %MF0

		Czas wykonywania (s)		Rozmiar (słowa)
		3710	3720	37xx

∞ Wartość, którą należy odliczyć dla wartości bezpośrednich

16#1234/%MW0		1.20	1.10	0
16#12345678 / %MD0 lub %MF/0		1.21	0.75	1

∞ Wartość jaką należy dodać dla słów indeksowanych, podwójnych, zmiennoprzecinkowych

%MW2[%MW0] lub %MD2[%MW0] lub %MF2[%MW0]	obiekt przed := lub lub	10.52	10.05	4
	1-a operacja : pierwszy argument nie indeksowany albo operacja przypisania	11.20	10.60	5
	2-gi argument jeśli 1-y argument jest indeksowany	13.37	12.60	5

∞ Wartość jaką należy dodać dla poniżej wymienionych obiektów:

%KWi, %KWi[%MW0], %KDi, %KFi, słowa wspólne, słowa I/O

		70.98	63.50	2

Współczynniki korekcyjne uwzględniające kontekst operacji

∞ Wartość jaką należy dodać, gdy operacja zajmuje co najmniej 2-gą pozycję w wyrażeniu, np. *%MW2 w := %MW0 * %MW1 * %MW2 zawiera następujące operacje

%MW0		0.69	0.55	0
%MD0 i %MF0		0.99	0.75	0

∞ Wartość jaką należy dodać dla operacji o wyższym priorytecie,lub której wynik mieści się w nawiasie, np. %MW0 + %MW2 + (...)

%MW0		2.86	2.55	1
%MD0 i %MF0		3.60	3.15	1

ST	Obiekty	Warunki	Czas wykonywania (s)		Rozmiar (słowa)
			3710	3720	37xx
obiekty za :=	%MW0		4.81	4.50	2
	%MD0,%MF0		6.45	5.70	2
:=	%MW0		4.46	4.30	2
	%MD0 and %MF0		5.15	4.85	2
=, <>, <=, <, >, >=	%MW0		8.94	8.50	4
	%MD0		10.71	10.26	4
	%MF0		29.06	28.39	4
AND, OR, XOR	%MW0		7.29	6.90	3
	%MD0		9.21	8.55	3
+, -	%MW0		7.29	6.90	3
	%MD0		9.21	8.55	3
	%MF0		62.83	61.20	3
*	%MW0		9.75	9.10	3
	%MD0		39.63	36.50	3
	%MF0		58.26	56.90	3
/, REM	%MW0		10.69	10.08	3
	%MD0		205.21	201.38	3
/	%MF0		62.47	60.25	3
ABS, -obiekt	%MW0		7.20	6.95	3
	%MD0		9.97	9.53	3
	%MF0		13.01	12.50	3
NOT	%MW0		6.69	6.45	3
	%MD0		7.80	7.40	3
SQRT	%MW0		17.02	16.70	3
	%MD0		85.73	85.25	3
	%MF0		165.04	158.40	3
INC, DEC	%MW0		4.86	4.40	2
	%MD0		5.20	4.75	2
SHL, SHR, ROL, ROR	%MW0	dla 1 bitu	17.74	17.05	5
	%MD0	dla 1 bitu	20.58	19.15	5
		na każdy bit dodatkowy	0.063		
LN	%MF0		1371.60	1270.00	3
LOG	%MF0		1458.00	1350.00	3
EXP	%MF0		1155.60	1070.00	3
EXPT	%MF0		2988.00	2490.00	3
TRUNC	%MF0		204.00	170.00	3
COS	%MF0		2829.60	2620.00	3
SIN	%MF0		2840.40	2630.00	3
TAN	%MF0		2937.60	2720.00	3
ACOS	%MF0		4082.40	3780.00	3
ASIN	%MF0		4082.40	3780.00	3
ATAN	%MF0		2786.40	2580.00	3
DEG_TO_RAD	%MF0		852.00	710.00	3
RAD_TO_DEG	%MF0		720.00	600.00	3

8.2.4 Instrukcje programowe

ST	Obiekty	Warunki	Czas wykonywania (s)		Rozmiar (słowa)
			3710	3720	37xx
Jump %Li			41.93	38.20	3
Maskevt			12.21	10.80	1
Unmaskevt			40.27	37.10	1
SRi			48.68	42.88	3
Return			42.18	38.33	3

8.2.5 Struktura komend

ST		Czas wykonywania (s)		Rozmiar (słowa)
		3710	3720	37xx
<war>	warunek wykonania			
bit, który można wymuszać	patrz instrukcje logiczne LD %M1			
porównywanie	patrz porównywanie =,<,> ...			
if <war> then <akcja> end_if;	czasy i rozmiary zamieszczone poniżej powinny być dodawane do akcji zamieszczonych w strukturze			
warunek typu prawda		3.60	3.30	2
warunek typu fałsz (skok)		5.55	5.40	
If <war> then <akcja1> else <akcja2> end_if;				
warunek typu prawda		9.15	8.70	4
warunek typu fałsz		5.55	5.40	
while <war> do.<akcja> end_while				
wywołanie pętli sprzężonej		9.15	8.70	2
wyjście z pętli		5.55	5.40	
repeat <akcja> until <war> end_repeat				
wywołanie pętli sprzężonej		5.55	5.40	2
ostatnie przejście pętli		3.60	3.30	
for <słowo1:=słowo2>to <słowo3> do <akcja> end_for				
wywołanie komendy for, wykonywanej tylko 1 raz		8.58	8.25	15
wywołanie pętli sprzężonej (ze sprzężeniem zwrotnym)		29.38	27.35	
wyjście z pętli		20.42	19.40	

8.2.6 Konwersja numeryczna

ST	Czas wykonywania (s)			Rozmiar (słowa)
	3710	**3720 RAM**	**3720 zas.**	**37xx**
BCD_TO_INT	25.03	24.55	24.55	3
INT_TO_BCD	21.66	21.15	21.15	3
GRAY_TO_INT	36.98	36.55	36.55	3
INT_TO_REAL	40.90	40.75	40.75	3
DINT_TO_REAL	33.32	32.55	32.55	3
REAL_TO_INT	58.75	58.55	58.55	3
REAL_TO_DINT	44.59	44.05	44.05	3
DBCD_TO_DINT	1 324.85	1 065.15	1 134.70	5
DBCD_TO_INT	1 265.54	925.70	986.15	5
DINT_TO_DBCD	1 124.85	825.15	879.10	5
INT_TO_DBCD	564.85	445.15	474.40	5

8.2.7 Łańcuch bitów

ST	warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	**3720 RAM**	**3720 zas.**	**37xx**
Inicjacja tablicy bitowej					
%M30:8 := 0	8 bitów	19.38	18.88	18.88	6
%M30:16 := 1	16 bitów	20.38	19.88	19.88	6
%M30:24 := 2	24 bity	24.25	23.35	23.35	6
%M30:32 := 2	32 bity	25.25	24.35	24.35	6

ST	warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	3720 RAM	3720 zas.	37xx
Kopiowanie tablicy bitowej do tablicy bitowej					
%M30:8 := %M20:8	8 bitów	25.54	24.79	24.79	6
%M30:16 := %M20:16	16 bitów	26.16	25.41	25.41	6
%M30:24 := %M20:24	24 bity	33.41	32.26	32.26	6
%M30:32 := %M20:32	32 bity	35.91	34.76	34.76	6
%M30:16 := COPY_BIT(%M20:16)	16 bitów	281.63	230.00	244.95	9
	32 bity	440.82	360.00	383.40	9
	128 bitów	1 261.22	1 030.00	1 096.95	9
Operacje logiczne na tablicach bitowych					
AND_ARX, OR_ARX, XOR_ARX					
%M0:16 := AND_ARX(%M30:16,%M50:16)	16 bitów	397.42	320.00	340.80	12
%M0:32 := AND_ARX(%M30:32,%M50:32)	32	620.97	500.00	532.50	12
%M0:128 := AND_ARX(%M30:128,%M50:128)	128	1 887.74	1 520.00	1 618.80	12
NOT_ARX					
%M0:16 := NOT_ARX(%M30:16)	16 bitów	281.63	230.00	244.95	9
	32	440.82	360.00	383.40	9
	128	1 261.22	1 030.00	1 096.95	9
Kopiowanie tablicy bitowej do tablicy słów					
%MW1 := %M30:8	8 bitów	14.84	14.36	14.36	5
%MW1 := %M30:16	16 bitów	16.34	15.86	15.86	5
%MD2 := %M30:24	24 bity	14.54	14.23	14.23	5
%MD2 := %M30:32	32 bity	16.04	15.73	15.73	5
%MW1:4 := BIT_W(%M40:80.0,17.2)	17 bitów	501.43	390.00	415.35	16
%MD1:4 := BIT_D(%M30:80.0,33.0)	33 bity	379.53	530.00	564.45	16
Kopiowanie tablicy słów do tablicy bitowej					
%M30:8 := %MW1	8 bitów	19.28	18.68	18.68	5
%M30:16 := %MW2	16 bitów	20.28	19.68	19.68	5
%M30:24 := %MD1	24 bity	21.20	20.37	20.37	5
%M30:32 := %MD3	32 bity	22.20	21.37	21.37	5
%M30:32 := W_BIT(%MW200:2.0,2.0)	32 bity	488.68	370.00	394.05	16
%M30:32 := D_BIT(%MD0:1.0,2.0)	32 bity	567.33	460.00	489.90	16

8.2.8 Tablice słów, słów podwójnych i wartości zmiennoprzecinkowych

ST	Warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	3720 RAM	3720 zas.	37xx
IInicjacja tablicy słów za pomocą słowa					
%MW0:10 := %MW100	10 słów	47.46	42.15	42.15	7
	na jedno słowo	0.34	0.20	0.20	
%MD0:10 := %MD100	10 słów podw.	81.27	74.45	74.45	7
	na słowo podw.	2.87	2.65	2.65	
Kopiowanie tablicy słów do tablicy słów					
%MW0:10:=%MW20:10;	10 słow	95.80	85.35	85.35	9
	na jedno słowo	0.77	0.50	0.50	
%MD0:10:=%MD20:10;	10 słów podw.	111.13	97.65	97.65	9
	na słowo podw.	1.54	1.00	1.00	
Operacje logiczne i arytmetyczne między dwiema tablicami słów					
+, -					
%MW0:10 :=%MW10:10 + %MW20:10;	10 słów	168.04	151.95	151.95	14
	na jedno słowo	7.13	6.35	6.35	
%MD0:10:=%MD10:10+%MD20:10;	10 słów podw.	239.17	214.40	214.40	14
	na słowo podw.	13.84	12.25	12.25	
*					
%MW0:10:=%MW10:10 * %MW20:10;	10 słów	189.32	175.40	175.40	14
	na jedno słowo	9.27	8.70	8.70	
%MD0:10:=%MD10:10 * %MD20:10;	10 słów podw.	710.35	603.80	603.80	14
	na słowo podw.	61.64	51.20	51.20	
/, REM					
%MW0:10:=%MW10:10 / %MW20:10;	10 słów	224.76	181.40	181.40	14
	na jedno słowo	13.14	9.30	9.30	
%MD0:10:=%MD10:10 / %MD20:10;	10 słów podw.	2 192.38	2 157.35	2 157.35	14
	na słowo podw.	209.16	206.55	206.55	
AND, OR, XOR					
%MW0:10:=%MW10:10 AND %MW20:10;	10 słów	163.69	147.40	147.40	14
	na jedno słowo	6.66	5.85	5.85	
%MD0:10:=%MD10:10 AND %MD20:10;	10 słów podw.	240.14	215.90	215.90	14
	na słowo podw.	13.94	12.40	12.40	

ST	Warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	3720 RAM	3720 zas.	37xx

Operacje logiczne i arytmetyczne między jedną tablicą słów a jednym słowem

+, -					
%MW0:10 :=%MW10:10 + %MW20;	10 słów	119.12	108.55	108.55	12
or %MW0:10 := %MW20 + %MW10:10	na jedno słowo	2.87	2.65	2.65	
%MD0:10 :=%MD10:10 + %MD20;	10 słów podw.	159.68	147.45	147.45	12
	na słowo podw.	6.57	6.25	6.25	
*					
%MW0:10 :=%MW20*%MW10:10;	10 słów	166.86	132.45	132.45	12
	na jedno słowo	7.94	5.05	5.05	
%MD0:10:=%MD20*%MD10:10;	10 słów podw.	587.01	522.95	522.95	12
	na słowo podw.	49.18	43.80	43.80	
/, REM					
%MW0:10 :=%MW10:10 / %MW30;	10 słów	196.69	155.85	155.85	12
	na jedno słowo	10.86	7.30	7.30	
%MD0:10:=%MD10:10 / %MD30;	10 słów podw.	2 230.17	2 173.95	2 173.95	12
	na słowo podw.	213.66	208.90	208.90	
AND, OR, XOR					
%MW0:10 :=%MW10:10 AND %MW20;	10 słów	117.20	106.45	106.45	12
	na jedno słowo	2.64	2.40	2.40	
%MD0:10:=%MD20*%MD10:10;	10 słów podw.	587.01	522.95	522.95	12
	na słowo podw.	6.47	6.15	6.15	
NOT					
%MW0:10 :=NOT(%MW10:10);	10 słów	110.28	100.25	100.25	9
	na jedno słowo	2.96	2.75	2.75	
%MD0:10:=NOT(%MD10:10);	10 słów podw.	126.39	114.00	114.00	9
	na słowo podw.	4.50	4.05	4.05	

Funkcje sumowania w tablicy

%MW20:=SUM(%MW0:10);	10 słów	74.30	69.00	69.00	10
	na jedno słowo	2.44	2.35	2.35	
%MD20:=SUM(%MD0:10);	10 słów podw.	83.58	76.90	76.90	10
	na słowo podw.	3.17	2.95	2.95	
%MF20:=SUM_ARR(%MF0:10);	10 słów podw.	1634	1257	1257	10
	na słowo podw.				

ST	Warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	3720 RAM	3720 zas.	37xx
Porównywanie tablic					
%MW20:=EQUAL(%MW0:10;%MW10:10);	10 słów	103.78	93.50	93.50	11
	na jedno słowo	1.13	0.90	0.90	
%MD20:=EQUAL(%MD0:10;%MD10:10);	10 słów podw.	116.17	103.40	103.40	11
	na słowo podw.	2.23	1.75	1.75	
%MF20:=EQUAL_ARR(%MF0:10;%MF10:10);	10 słów podw.	741	570	607	11
	na słowo podw.				
Wyszukiwanie					
%MW20 := FIND_EQW(%MW0:10,%KW0)	10 słów, max. (od pierwszego)	340.00	250.00	266.25	15
%MD20 := FIND_EQD(%MD0:10, %KD0)	10 słów podw., max. (od 1-ego)	350.00	260.00	276.90	16
%MF20 := FIND_EQR(%MF0:10, %KF0)	10 słów podw.	833	648	690.12	15
%MF20 := FIND_EQRP(%MF0:10, %KF0)	10 słów podw.	845	650	692.25	15
%MD20 := FIND_GTR(%MF0:10, %KF0)	10 słów podw.	836	643	684.79	15
%MD20 := FIND_LTR(%MF0:10, %KF0)	10 słów podw.	836	643	684.79	15
Wyszukiwanie wartości max i min.					
%MW20 := MAX_ARW(%MW0:10)	10 słów	350.00	260.00	276.90	9
%MD20 := MAX_ARD(%MD0:10)	10 słów podw.	410.00	300.00	319.50	9
%MF20 := MAX_ARR(%MF0:10)	10 słów podw.	1366	1051	1119.31	9
%MF20 := MIN_ARR(%MF0:10)	10 słów podw.	1270	977	1040.50	9
Liczba wystąpień w tablicy					
%MW20 := OCCUR_ARW(%MW0:10, %KW0)	10 słów	350.00	250.00	266.25	15
%MD20 := OCCUR_ARD(%MD0:10, %KD0)	10 słów podw.	370.00	270.00	287.55	16
%MF20 := OCCUR_ARR(%MF0:10, %KF0)	10 słów podw.	1265	973	1036.24	16
Przesunięcie okrężne					
ROL_ARW(word or value, %MWj:10)	10 słów	550.00	400.00	426.00	9
ROL_ARD(%MDi,%MDj:10)	10 słów podw.	590.00	430.00	457.95	9
ROL_ARR(%MFi,%MFj:10)	10 słów podw.	585	450	479.25	9
Sortowanie					
SORT_ARW(%MWi,%MWj:10)	10 słów, max. (od pierwszego)	970.00	700.00	745.50	9
SORT_ARD(%MDi,%MDj:10)	5 słów podw., max. (od pierwszego)	610.00	450.00	479.25	9
SORT_ARR(%MFi,%MFj:10)	10 słów podw.	1863	1433	1526.14	9

8.2.9 Zarządzanie czasem

ST	Czas wykonywania (s)			Rozmiar (słowa)
	3710	3720 RAM	3720 zas.	37xx

Data, czas i czas trwania

%MW2:4 := ADD_DT(%MW2:4,%MD8)	4 400.00	3 300.00	3 514.50	13
%MD2 := ADD_TOD(%MD2,%MD8)	2 100.00	1 550.00	1 650.75	9
%MB2:11 := DATE_TO_STRING(%MD40)	1 370.00	900.00	958.50	9
%MW5 := DAY_OF_WEEK()	220.00	280.00	298.20	5
%MD10 := DELTA_D(%MD2, %MD4)	1 520.00	1 130.00	1 203.45	9
%MD10 := DELTA_DT(%MD2:4,%MW6:4)	3 170.00	2 300.00	2 449.50	13
%MD10 := DELTA_TOD(%MD2,%MD4)	2 330.00	1 700.00	1 810.50	9
%MB2:20 := DT_TO_STRING(%MW50:4)	2 050.00	1 450.00	1 544.25	11
%MW2:4 := SUB_DT(%MW2:4,%MD8)	4 750.00	3 500.00	3 727.50	13
%MD2 := SUB_TOD(%MD2,%MD8)	2 330.00	1 700.00	1 810.50	9
%MB2:15 := TIME_TO_STRING(%MD40)	1 560.00	1 200.00	1 278.00	9
%MB2:9 := TOD_TO_STRING(%MD40)	1 270.00	800.00	852.00	9
%MD100 := TRANS_TIME(%MD2)	500.00	500.00	532.50	7

Dostęp do zegara czasu rzeczywistego

RRTC(%MW0:4)	93.60	84.80	84.80	5
WRTC(%MW0:4)	248.61	230.85	230.85	5
PTC(%MW0:5)	97.98	88.60	88.60	5
SCHEDULE(%MW0,%MW1,%MW2, %MD10,%MD12,%M0)	1430	1100	1171.5	5

8.2.10 Łańcuchy znaków

ST	Warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	3720 RAM	3720 zas.	37xx
Przypisywanie łańcuchów znaków					
%MB0:8:=%MB10:8	8 znaków	105.16	93.80	93.80	9
	na jeden znak	1.65	1.30	1.30	
%MB0:8:='abcdefg'	8 na jeden znak	120.72	110.20	110.20	11
	na jeden znak	4.15	3.85	3.85	0.5
Konwersja Słowo <-> Łańcuch znaków					
%MW1:=STRING_TO_INT(%MB0:7)		97.69	91.95	91.95	7
%MB0:7:=INT_TO_STRING(%MW0)		104.36	96.70	96.70	7
Konwersja Słowo podwójne <-> Łańcuch znaków					
%MD1:=STRING_TO_DINT(%MB0:13)		1 070.53	965.62	965.62	7
%MB0:13:=DINT_TO_STRING(%MD0)		322.29	295.35	295.35	7
Konwersja Wartość zmiennoprzecinkowa <-> Łańcuch znaków					
%MF1:=STRING_TO_REAL(%MB0:15)		1 783.70	1 634.53	1 634.53	7
%MB0:15:=REAL_TO_STRING(%MF0)		741.75	681.20	681.20	7
Operacje na łańcuchach					
%MB10:20 := CONCAT(%MB30:10,%MB50:10)		1 170.00	770.00	820.05	15
%MB10:20 := DELETE(%MB10:22.2,3);		950.00	600.00	639.00	15
%MW0 := EQUAL_STR(%MB10:20,%MB30:20);	5-ty znak jest inny	860.00	520.00	553.80	13
%MW0 := FIND(%MB10:20,%MB30:10);		1 610.00	1 000.00	1 065.00	13
%MB10:20 := INSERT(%MB30:10,%MB50:10.4);		1 270.00	800.00	852.00	17
%MB10:20 := LEFT(%MB30:30,20);		920.00	570.00	607.05	13
%MW0 := LEN(%MB10:20);		770.00	340.00	362.10	9
%MB10:20 := MID(%MB30:30,20,10);		1 080.00	700.00	745.50	15
%MB10:20 : = REPLACE(%MB30:20,%MB50:10,10,10);		1 450.00	870.00	926.55	19
%MB10:20 := RIGHT(%MB30:30,20);		1 480.00	950.00	1 011.75	13

8.2.11 Funkcje specjalne oraz funkcje Orphee

ST	Warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	3720 RAM	3720 Zasobnik	37xx
Komunikacja					
SEND_REQ(%KW0:6,15,%MW0:1,%MW10:10,%MW30:4)		2182	1818	1936	21
SEND_TLG(%KW0:6,1,%MW0:5,%MW30:2)		1636	1364	1452	15
Interfejs Człowiek - Maszyna					
SEND_MSG(ADR#1.0,%MW0:2,%MW10:2)		2 240	2 000	2208	19
SEND_ALARM(ADR#1.0,%MW0:2,%MW10:2)		2 240	2 000	2208	19
GET_MSG(ADR#1.0,%MW0:2,%MW10:2)		2 240	2 000	2 208	19
GET_VALUE(ADR#1.0,%MW0,%MW10:2)		1 120	1 000	1 104	17
ASK_MSG(ADR#1.0,%MW0:2,%MW10:2,%MW20:2)		2 240	2 000	2 208	23
ASK_VALUE(ADR#1.0,%MW0,%MW10:2,%MW20:2)		2 240	2 000	2 208	21
DISPLAY_ALRM(ADR#1.0,%MW0,%MW10:2)		1 120	1 000	1 104	17
DISPLAY_GRP(ADR#1.0,%MW0,%MW10:2)		1 120	1 000	1 104	17
DISPLAY_MSG(ADR#1.0,%MW0,%MW10:2)		1 120	1 000	1 104	17
CONTROL_LEDS(ADR#1.0,%MW0:2,%MW10:2)		2 240	2 000	2 208	19
ASSIGN_KEYS(ADR#1.0,%MW0:2,%MW10:2)		2 240	2 000	2 208	19
PANEL_CMD(ADR#1.0,%MW0:2,%MW10:2)		2 240	2 000	2 208	19
Sterownie procesem					
PID("PIDS1",'Unit',%IW3.5,%MW12,%M16,%MW284:43)	deval_mmi=0	1320	1100	1172	24
	deval_mmi=1	1080	900	958.5	
PWM(%MW11,%Q2.1,%MW385:5)		600	500	532.5	11
SERVO(%MW12,%IW3.6,%Q2.2,%Q2.3,%MW284:43, %MW390:10)		960	800	852	19
PID_MMI(ADR#0.0.4,%M1,%M2:5,%MW410:62)	EN=1	1140	950	1012	20

ST	Warunki	Czas wykonywania (s)			Rozmiar (słowa)
		3710	3720 RAM	3720 Zas.	37xx
Funkcje Orphee					
DSHL_RBIT(%MD102,16,%MD204,%MD206)	czytanie10 słów	440	320	341	13
DSHR_RBIT(%MD102,16,%MD204,%MD206)	zapisanie10 słów	660	480	511	13
DSHRZ_C(%MD102,16,%MD204,%MD206)	"odwrócenie"10 słów	410	310	330	13
WSHL_RBIT(%MW102,8,%MW204,%MW206)	wymiana 10 słów	300	220	234	13
WSHR_RBIT(%MW102,8,%MW204,%MW206)	20 bajtów	390	280	298	13
WSHRZ_C(%MW102,8,%MW204,%MW206)	20 bajtów	300	220	234	13
SCOUNT(%M100,%MW100,%M101,%M102,%MW101, %MW102,%M200,%M201,%MW200,%MW201)	20 bajtów	510	410	437	25

8.2.12 Jawna wymiana I/O

Read_Sts %CHi.MOD					
Dowolna aplikacja z wyjątkiem aplikacji dla kanału komunikacyjnego procesora		**30**	**30**	**32**	**2**
Read_Sts %CHi					
Wejście analogowe		180	180	216	6
Wyjscie analogowe		90	70	74	
Moduł licznika CTZ		110	95	104	
Write_Param %CHi					
Wejście analogowe		790	570	790	6
Moduł licznika CTZ		1127	1080	1083	
Read_Param %CHi					
Wejście analogowe		260	290	316	6
Moduł licznika CTZ		338	295	300	
Save_Param %CHi					
Wejście analogowe		1234	1220	1240	6
Moduł licznika CTZ		1370	1220	1240	
Restore_Param %CHi					
Wejście analogowe		550	510	535	6
Moduł licznika CTZ		1160	1080	1097	
Write_Cmd %CHi					
Wyjście dyskretne		50	47	52	6

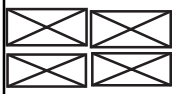
8.3 Sterownik TSX 57

W niniejszym rozdziale pogrupowano procesory w trzy rodziny, w związku z czym w tablicach będą zamieszczane tylko symbole odpowiadające tym grupom:

Oznaczenie	Procesory
A	TSX 57 10 i PCX 57 10
B	TSX 57 20, TSX 57 25, TSX 57 30, TSX 57 35 PMX 57 10, PCX 57 35
C	TSX 57 40, TSX 57 45 PMX 57 20, PMX 57 35, PMX 57 45

Uwaga: W dalszej części instrukcji, w stopce strony wyświetlana będzie informacja na temat przynależności procesorów do odpowiednich grup.

8.3.1 Instrukcje logiczne

LD	IL	ST	Obiekty	Czas wykonywania (s)						Size (słowa)
				A	A > 4K	B/C RAM	B/C RAM > 4K	B/C Zas.	B/C Zas. > 4K	57xx
				0.29		0.12		0.21		1
	LD,		%M1 (1)	0.58		0.25	0.37	0.37	0.58	1
	LDN									
			%M1[%MW2]	2.33		1.00	1.00	1.58	1.58	6
			%MW0:X0 (2)	1.46		0.62	0.62	1.00	1.00	4
			%IWi.j:Xk (3)	2.33		1.00	1.00	1.58	1.58	6
			%MW0[%MW10]:X0	3.50		1.50	1.50	2.37	2.37	9
			%KW0[%MW10]:X0	3.50		1.50	1.50	2.37	2.37	9
	LDR,		%M1	0.87		0.37	0.50	0.58	0.79	2
	LDF									
			%M1[%MW2]	2.62		1.12	1.12	1.79	1.79	7
	AND,			idem LD, LDN						
	ANDN , AND (, AND (N , idem OR									
	ANDR, ANDF, AND (R, AND (F, idem OR			idem LDR, LDF						
	XOR, XORN		%M1	2.04		0.87	1.12	1.37	1.79	5
			%M1[%MW2]	5.54		2.37	1.62	3.79	2.62	13
			%MW0:X0	4.08		1.75	1.37	2.83	2.25	10
			%IWi.j:Xk	4.96		2.12	1.75	3.42	2.83	14
			%MW0[%MW10]:X0	7.87		3.37	2.25	5.37	3.62	14
			%KW0[%MW10]:X0	7.87		3.37	2.25	5.37	3.62	18
	XORR, XORF		%M1	2.25		1.13	1.62	1.69	2.62	9
			%M1[%MW2]	27.28		26.13	2.87	26.44	4.62	19
	ST, STN,		%M1	1.17		0.50	0.62	0.75	0.96	2
	S, R									
			%M1[%MW2]	2.62		1.12	1.12	1.75	1.75	7
			%MW0:X0	1.75		0.75	0.75	1.17	1.17	4
			%NW{i}j:Xk (3)	2.62		1.12	1.12	1.75	1.75	8
			%MW0[%MW10]:X0	3.79		1.25	1.25	2.54	2.54	8
sprzężenia wielokrotne w Ladder, «koszt» 2-go sprzężenia i następnych				0.87		0.37	0.50	0.54	0.75	1
blok operacyjny	[akcja]		blok wykonywany	0.58		0.25	0.25	0.42	0.42	1
			nie wykonywany	0.71		0.37	0.37	0.54	0.54	1

(1) Obejmuje wszystkie obiekty, co do których można stosować wymuszenie: %I,%Q,%X,%M,%S

(2) Inne obiekty tego typu: bity wyjściowe bloku funkcyjnego %TMi.Q ..., bity wydzielone ze słów systemowych %SWi:Xj, bity wydzielone ze słów %KW, bity błędów %Ii.j.ERR

(3) Inne obiekty tego typu: bity wydzielone ze słów wspólnych %NW{i}j:Xk, bity wydzielone ze słów I/O %IWi.j.Xk, %QWi.j.Xk

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

LD	IL	ST	Obiekty	Czas wykonywania (s)						Rozmiar (słowa)
				A	A > 4K	B/C RAM	B/C RAM > 4K	B/C Zas.	B/C Zas. > 4K	57xx
blok porównywania poziomy	LD [porownyw.]		czas dodawany do porównywania	0.00	0.00	0.00	0.00	0.00	0.00	0
blok porównywania pionowy			pomiędzy 2 %MWi	2.04	2.04	0.87	0.87	1.37	1.37	5
zamknięcie (zbieg)	)	)		0.29	0.29	0.12	0.12	0.21	0.21	1
otwarcie nie poprzedzone zamknięciem			ladder, 1 otwarcie	0.29	0.29	0.12	0.12	0.21	0.21	1
	MPS, MPP,		lista MPS+MPP	0.87	0.87	0.37	0.37	0.62	0.62	3
	MRD		lista MRD	0.29	0.29	0.12	0.12	0.21	0.21	1

8.3.2 Bloki funkcyjne

LD	IL	ST	Obiekty/warunki	Czas wykonywania (s)			Rozmiar (słowa)
				A	B	C	57xx

Zegar IE

zbocze rosnące na IN	IN %TM1 (zb. rosnące)	START %TM1	start zegara	67.87	46.92	13.52	3
zbocze opadające na IN (zbocze opadające)	IN %TM1	DOWN %TM1	zatrzymanie zegara	27.84	19.50	5.01	
IN =1	IN %TM1 (=1)		zegar pracuje *on*	32.15	22.66	5.80	
IN =0	IN %TM1 (=0)		zegar nie pracuje *off*	28.60	20.18	5.40	

Zegar PL7-3

		START %T1	uaktywnienie				3
		STOP %T1	zamrożenie	23.08	16.16	5.13	
E=0		RESET %T1	kasowanie	23.46	16.26	5.85	
			zegar pracuje *on*	30.40	21.51	5.78	
			zegar nie pracuje *off*				

Licznik dwukierunkowy

kasowanie, R=1	R %C8 (=1)	RESET %C8	kasowanie	30.38	21.26	6.11	3
nastawianie, S=1	S %C9 (=1)	PRESET %C9	nastawianie	33.99	23.55	6.25	
zbocze rosnące na CU	CU %C8 (zb. rosnące)	UP %C8	zliczanie *up*	33.81	23.49	6.50	
zbocze rosnące na CD	CD %C9 (zb. rosnące)	DOWN %C9	odliczanie *down*	33.81	23.49	6.50	
kroki nie aktywne	R/S/CU/CD bit nie aktywny		brak akcji	22.37	16.22	4.58	

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

Bloki funkcyjne (ciąg dalszy)

LD	IL	ST	obiekty/warunki	Czas wykonywania (s)			Rozmiar (słowa)
				A	B	C	57xx

Przerzutnik monostabilny

zbocze rosnące na S	S %MN0, (zb. rosnące)	START %MN0	start	57.42	40.65	11.88	3
S=1	S %MN0, S=1/0		przerzutnik aktywny	20.38	14.11	4.04	

Rejestr

zbocze na I	I %R2 (front)	PUT %R2	zapisywanie	35.69	24.92	6.72	3
zbocze na O	O %R2 (front)	GET %R2	odczytywanie	35.69	24.92	6.72	
R=1	R %R1 (=1)	RESET %R2	kasowanie	26.83	18.80	6.11	
wejścia nie aktywne bit	I/O/R, nie akt.		brak akcji	20.71	15.50	4.62	

Bęben

zbocze na U	U %DR0	UP %DR1	do przodu, ustalony	268.69	185.41	58.68	3
			na bit sterujący	25.00	25.00	25.00	
R=1	R %DR1	RESET %DR2	kasowanie, stałe	257.01	176.06	57.00	
			na bit sterujący	25.00	25.00	25.00	
wejścia nie aktywne	R/U, bit nie akt		brak akcji, stały	259.07	179.06	56.82	
			na bit sterujący	25.00	25.00	25.00	

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

8.3.3 Operacje arytmetyczne - obiekty całkowite i zmiennoprzecinkowe

Współczynnik korekcji uwzględniający typ obiektu

Czas i wartości zamieszczone na następnych stronach dotyczą: %MW0, %MD0 i %MF0

Czas wykonywania (s)					Rozmiar (słowa)
A	B RAM	B zas.	C RAM	C zas.	57xx

- Wartość, którą należy odjąć dla wartości bezpośrednich

16#1234 / %MW0	0.29	0.12	0.17	0.12	0.17	0
16#12345678 / %MD0 or %MF0	0.29	0.12	0.12	0.12	0.12	1

- Wartość jaką należy dodać dla słów podwójnych, indeksowanych i zmiennoprzecinkowych

%MW2[%MW0] or %MD2[%MW0] or %MF2[%MW0]	obiekt poprzedzający :=	2.04	0.87	1.37	0.87	1.37	5
	1-a operacja, przypisanie	2.04	0.87	1.37	0.87	1.37	5
	2-ga operacja (lub 1-y indeksowany argument)	2.04	0.87	1.37	0.87	1.37	5

- Wartości, które należy dodać dla następujących obiektów:

Słowa wspólne, słowa I/O	0.87	0.37	0.58	0.37	0.58	2

Współczynniki korekcji uwzględniające kontekst operacji

- Wartość jaką należy dodać, gdy operacja zajmuje co najmniej 2-gą pozycję w wyrażeniu, np. *%MW2 w := %MW0 * %MW1 * %MW2 zawiera następujące operacje: *,/,REM (na słowach, słowach podwojnych) i operacje na wartościach zmiennoprzecinkowych

%MW0	0.58	0.25	0.37	0.25	0.37	1
%MD0 i %MF0	0.87	0.37	0.54	0.37	0.54	1

- Wartość jaką należy dodać dla operacji o wyższym priorytecie,lub której wynik mieści się w nawiasie, np. %MW0 + %MW2 + (...)

%MW0	0.29	0.12	0.17	0.12	0.17	1
%MD0 i %MF0	0.58	0.25	0.33	0.25	0.33	1

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

ST	obiekty	warunki	Czas wykonywania (s)					Rozmiar (słowa)
			A	B RAM	B zas.	C RAM	C zas.	57xx
obiekt za :=	%MW0		0.87	0.37	0.58	0.37	0.58	2
		%MW0+(..., lub przed *, /, REM	1.17	0.50	0.75	0.50	0.75	2
	%MD0		1.17	0.50	0.75	0.50	0.75	2
		%MD0+(..., lub przed *, /, REM	1.75	0.75	1.08	0.75	1.08	2
	%MF0		1.75	0.75	1.08	0.75	1.08	2
:=	%MW0		0.87	0.37	0.58	0.37	0.58	2
	%MD0 and %MF0		1.17	0.50	0.75	0.50	0.75	2
=, <>, <=, <, >, >=	%MW0		1.17	0.50	0.79	0.50	0.79	4
	%MD0		1.46	0.62	1.04	0.62	1.04	4
	%MF0		48.36	33.88	34.13	4.17	4.42	4
AND, OR, XOR	%MW0		0.87	0.37	0.58	0.37	0.58	3
	%MD0		1.17	0.50	0.75	0.50	0.75	3
+, -	%MW0		0.87	0.37	0.58	0.37	0.58	3
	%MD0		1.17	0.50	0.75	0.50	0.75	3
	%MF0		99.42	71.51	71.76	4.42	4.67	3
*	%MW0		10.83	9.14	9.35	4.65	4.86	3
	%MD0		55.31	42.71	42.96	15.26	15.51	3
	%MF0		87.60	63.61	63.86	4.42	4.67	3
/, REM	%MW0		11.93	9.99	10.20	4.94	5.15	3
	%MD0		333.15	226.54	226.79	49.38	49.63	3
/	%MF0		95.83	68.51	68.76	5.72	5.97	3
ABS, -obiekt	%MW0		0.87	0.37	0.58	0.37	0.58	2
	%MD0		1.17	0.54	0.75	0.54	0.75	2
	%MF0		18.82	13.01	13.26	3.67	3.92	2
NOT	%MW0		0.87	0.37	0.58	0.37	0.58	2
	%MD0		1.17	0.54	0.75	0.54	0.75	2
SQRT	%MW0		22.20	16.24	16.45	5.53	5.74	3
	%MD0		111.29	89.66	89.91	16.14	16.39	3
	%MF0		233.62	173.01	173.26	5.10	5.35	3
INC, DEC	%MW0		1.17	0.50	0.75	0.50	0.75	2
	%MD0		1.75	0.75	1.08	0.75	1.08	2
SHL, SHR, ROL, ROR	%MW0	dla 1 bitu	2.92	1.25	1.96	1.25	1.96	10
	%MD0	dla 1 bitu	3.21	1.37	2.12	1.37	2.12	10
		na 1 bit dodatkowy	0.042	0.042	0.042	0.042	0.042	
LN	%MF0		1711	1270	1270	4.53	4.78	3
LOG	%MF0		1819	1350	1350	4.53	4.78	3
EXP	%MF0		1442	1070	1070	12.03	12.28	3
EXPT	%MF0		3412	2624	2775	98.70	98.70	3
TRUNC	%MF0		215	165	175	52.40	52.40	3
COS	%MF0		3530	2620	2620	4.53	4.78	3
SIN	%MF0		3544	2630	2630	4.53	4.78	3
TAN	%MF0		3665	2720	2720	4.53	4.78	3
ACOS	%MF0		5093	3780	3780	5.33	5.58	3
ASIN	%MF0		5093	3780	3780	6.73	6.98	3
ATAN	%MF0		3476	2580	2580	4.33	4.58	3
DEG_TO_RAD	%MF0		814	626	662	139.90	139.90	3
RAD_TO_DEG	%MF0		791	608	643	140.20	140.20	3

8.3.4 Instrukcje programowe

ST	Obiekty	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
			A	B RAM	B zas.	C RAM	C zas.	57xx
Skok %Li			2.58	1.25	1.71	1.25	1.71	3
Maskevt			33.98	23.96	23.96	9.59	9.59	1
Unmaskevt			34.54	24.41	24.41	9.77	9.77	1
SRi			3.92	1.75	2.42	1.75	2.42	2
Return			1.00	0.50	0.71	0.50	0.71	2

8.3.5 Struktura komend

ST		Czas wykonywania (s)			Rozmiar (słowa)
		A	B/C RAM	B/C Zasobnik	57xx
<war>	oszacowywanie warunku				
bity, kóre można wymuszać	patrz instrukcje logiczne LD %M1				
porównywanie	patrz porównywanie =,<,> ...				
if <war > then <akcja> end_if;	czasy i rozmiary podane poniżej powinny zostać dodane do tych akcji, które znajdują się w strukturze				
warunek typu prawda		0.58	0.25	0.42	2
warunek typu fałsz (skok)		0.71	0.37	0.54	
If <war> then <akcja1> <akcja2> end_if;					
warunek typu prawda		1.29	0.62	0.96	4
warunek typu fałsz		0.71	0.37	0.54	
while <war> do.<akcja> end_while					
wywołanie pętli sprzężonej		1.29	0.62	0.96	2
wyjście z pętli <war> end_repeat		0.71		0.37	0.54
repeat <akcja> until					
wywołanie pętli sprzężonej		0.71	0.37	0.54	2
ostatnie "przejście" pętli		0.58	0.25	0.42	
for <Słowo1:=słowo2>to <słowod3><akcja> end_for					
wywołanie komendy for, wykonywanej tylko 1 raz		1.75	0.75	1.17	15
wywołanie pętli sprzężonej		5.08	2.12	3.29	
wyjście z pętli		2.46	1.12	1.71	

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

8.3.6 Konwersja numeryczna

ST	Czas wykonywania (s)					Rozmiar (słowa)
	A	B RAM	B zas.	C RAM	C zas.	57xx
BCD_TO_INT	30.33	24.44	24.65	4.75	4.96	3
INT_TO_BCD	24.48	20.19	20.40	4.51	4.72	3
GRAY_TO_INT	59.73	40.79	41.00	5.82	6.02	3
INT_TO_REAL	60.35	40.64	40.85	3.25	3.46	3
DINT_TO_REAL	49.14	34.76	35.01	3.67	3.92	3
REAL_TO_INT	91.59	60.86	61.11	3.38	3.63	3
REAL_TO_DINT	68.84	48.31	48.56	3.67	3.92	3
DBCD_TO_DINT	1 625.73	1 069.50	1 192.30	378.40	378.15	5
DBCD_TO_INT	1 379.53	909.63	1 014.07	323.83	323.62	5
DINT_TO_DBCD	1 251.05	819.50	913.55	314.88	314.63	5
INT_TO_DBCD	620.01	439.50	489.85	168.73	168.48	5

8.3.7 Łańcuch bitów

ST	warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas.	C RAM	C zas.	57xx

Inicjacja tablicy bitowej

%M30:8 := 0	8 bitów	6.71	2.87	4.12	2.87	4.12	7
%M30:16 := 1	16 bitów	11.37	4.87	6.79	4.87	6.79	7
%M30:24 := 2	24 bity	24.47	15.31	16.81	15.31	16.81	12
%M30:32 := 2	32 bity	29.13	17.31	19.48	17.31	19.48	12

Kopiowanie tablicy bitów do tablicy bitów

%M30:8 := %M20:8	8 bitów	13.71	5.87	8.17	5.87	8.17	8
%M30:16 := %M20:16	16 bitów	16.62	7.12	9.83	7.12	9.83	8
%M30:24 := %M20:24	24 bity	45.46	24.31	28.85	24.31	28.85	13
%M30:32 := %M20:32	32 bity	57.13	29.31	35.52	29.31	35.52	13
%M30:16 := COPY_BIT(%M20:16)	16 bitów	322.00	230.00	256.45	99.20	99.20	17
	32 bity	490.00	350.00	390.25	153.20	153.20	17
	128 bitów	1526.00	1090.00	1215.35	470.20	470.20	17

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas.	C RAM	C zas.	57xx

Operacje logiczne na tablicach bitowych

AND_ARX, OR_ARX, XOR_ARX							
%M0:16 := AND_ARX(%M30:16,%M50:16)	16 bitów	434.00	310.00	345.65	136.10	136.10	24
%M0:32 := AND_ARX(%M30:32,%M50:32)	32	686.00	490.00	546.35	213.00	213.00	24
%M0:128 := AND_ARX(%M30:128,%M50:128)	128	2 198.00	1 570.00	1 750.55	674.40	674.40	24
NOT_ARX							
%M0:16 := NOT_ARX(%M30:16)	16 bitów	322.00	230.00	256.45	99.20	99.20	17
	32	490.00	350.00	390.25	153.20	153.20	17
	128	1 526.00	1 090.00	1 215.35	470.20	470.20	17

Kopiowanie tablicy bitowej do tablicy słów

%MW1 := %M30:8	8 bitów	8.75	3.75	5.25	3.75	5.25	6
%MW1 := %M30:16	16 bitów	15.75	6.75	9.25	6.75	9.25	6
%MD2 := %M30:24	24 bity	23.04	10.54	13.83	10.54	13.83	6
%MD2 := %M30:32	32 bity	30.04	13.54	17.83	13.54	17.83	6
%MW1:2 := BIT_W(%M40:17,0,17,0)	17 bitów	518.00	370.00	412.55	150.50	150.50	23
%MD1:2 := BIT_D(%M30:33,0,33,0)	33 bity	728.00	520.00	579.80	200.00	200.00	23

Kopiowanie słowa lub tablicy słów do tablicy bitowej

%M30:8 := %MW1	8 bitów	6.71	2.87	4.08	2.87	4.08	6
%M30:16 := %MW2	16 bitów	11.37	4.87	6.75	4.87	6.75	6
%M30:24 := %MD1	24 bity	24.76	16.11	17.36	16.11	17.36	11
%M30:32 := %MD3	32 bity	29.42	18.11	20.02	18.11	20.02	11
%M30:32 := W_BIT(%MW0:2,0,2,0)	32 bity	518.00	370.00	412.55	142.31	142.31	23
%M30:32 := D_BIT(%MD0:1,0,2,0)	32 bity	616.00	440.00	490.60	169.23	169.23	23

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

8.3.8 Tablice słów, słów podwójnych i zmiennoprzecinkowych

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas.	C RAM	C zas.	57xx
Inicjacja tablicy słów za pomocą słowa							
%MW0:10 := %MW100	10 słów	74.48	50.16	51.12	27.04	28.00	10
	na jedno słowo		0.47	0.25	0.25	0.37	0.37
%MD0:10 := %MD100	10 words	115.45	78.29	79.29	33.59	34.59	10
	na jedno słowo		4.41	2.95	2.95	0.90	0.90
Kopiowanie tablicy słów do tablicy słów							
%MW0:10:=%MW20:10;	10 słów	139.71	93.76	95.26	45.65	47.14	15
	na jedno słowo		0.95	0.50	0.50	0.74	0.74
%MD0:10:=%MD20:10;	10 słów	151.18	102.46	103.96	53.75	55.25	15
	na jedno słowo		2.02	1.30	1.30	1.54	1.54
Operacje logiczne i arytmetyczne pomiędzy dwiema tablicami słów							
+, -							
%MW0:10 :=%MW10:10 + %MW20:10;	10 słów	236.81	162.54	164.79	71.59	73.84	23
	na jedno słowo		10.05	7.15	7.15	3.07	3.07
%MD0:10:=%MD10:10+%MD20:10;	10 słów	325.79	229.79	232.04	104.03	106.28	23
	na jedno słowo		18.41	13.55	13.55	6.12	6.12
*							
%MW0:10:=%MW10:10 * %MW20:10;	10 słów	246.52	184.99	187.24	76.30	78.55	23
	na jedno słowo		11.03	9.40	9.40	3.55	3.55
%MD0:10:=%MD10:10 * %MD20:10;	10 słów	881.77	658.44	660.69	241.22	243.47	23
	na jedno słowo		74.04	56.40	56.40	19.82	19.82
/, REM							
%MW0:10:=%MW10:10 / %MW20:10;	10 słów	249.44	192.64	194.89	77.99	80.24	23
	na jedno słowo		11.35	10.15	10.15	3.71	3.71
%MD0:10:=%MD10:10 / %MD20:10;	10 słów	3 669.10	2 501.99	2 504.24	544.65	546.90	23
	na jedno słowo		352.83	240.75	240.75	50.19	50.19
AND, OR, XOR							
%MW0:10:=%MW10:10 AND %MW20:10;	10 słów	235.38	160.14	162.39	71.15	73.40	23
	na jedno słowo		9.94	6.90	6.90	3.02	3.02
%MD0:10:=%MD10:10 AND %MD20:10;	10 słów	322.35	231.09	233.34	104.96	107.21	23
	na jedno słowo		18.05	13.65	13.65	6.20	6.20

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas.	C RAM	C zas.	57xx

Operacje logiczne i arytmetyczne między 1 tablicą słów a 1 słowem

+, -							
%MW0:10 :=%MW10:10 + %MW20;	10 słów	170.69	115.14	116.85	48.99	50.70	18
or %MW0:10 := %MW20 + %MW10:10	na jedno słowo		4.37	2.85	2.85	0.88	0.88
%MD0:10 :=%MD10:10 + %MD20;	10 słów podwójnych	230.52	156.91	158.66	67.61	69.36	18
	na słowo podwójne	9.92	6.75	6.75	2.62	2.62	
*							
%MW0:10 :=%MW20*%MW10:10;	10 słów	180.13	137.04	138.75	53.64	55.35	18
	na jedno słowo		5.31	5.05	5.05	1.35	1.35
%MD0:10:=%MD20*%MD10:10;	10 słów podwójnych	747.33	568.16	569.91	205.84	207.59	18
	na słowo podwójne	61.59	47.85	47.85	16.43	16.43	
/, REM							
%MW0:10 :=%MW10:10 / %MW30;	10 słów	212.87	166.39	168.10	65.15	66.86	18
	na jedno słowo		8.48	7.90	7.90	2.42	2.42
%MD0:10:=%MD10:10 / %MD30;	10 słów podwójnych	3 536.66	2 418.56	2 420.31	509.41	511.16	18
	na słowo podwójne	340.51	232.90	232.90	46.82	46.82	
AND, OR, XOR							
%MW0:10 :=%MW10:10 AND %MW20;	10 słów	170.89	115.59	117.30	49.09	50.79	18
	na jedno słowo		4.39	2.90	2.90	0.89	0.89
%MD0:10:=%MD20*%MD10:10;	10 słów podwójnych	747.33	568.16	569.91	205.84	207.59	18
	na słowo podwójne	9.81	6.50	6.50	2.57	2.57	
NOT							
%MW0:10 :=NOT(%MW10:10);	10 słów	161.71	109.06	110.56	46.70	48.20	15
	na jedno słowo		4.37	2.85	2.85	0.88	0.88
%MD0:10:=NOT(%MD10:10);	10 słów podwójnych	184.14	125.51	127.01	55.34	56.84	15
	na słowo podwójne	6.61	4.50	4.50	1.75	1.75	

Sumowanie w tablicy

%MW20:=SUM(%MW0:10);	10 słów	111.87	76.61	77.57	29.61	30.57	16
	na jedno słowo		3.40	2.40	2.40	0.53	0.53
%MD20:=SUM(%MD0:10);	10 słów podwójnych	130.74	87.74	88.74	34.03	35.03	16
	na słowo podwójne	4.96	3.30	3.30	0.93	0.93	
%MF20:=SUM_ARR(%MF0:10);	10 słów podwójnych	1905	1270.00	1313.82	303.20	303.20	16
	na słowo podwójne						

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas.	C RAM	C zas.	57xx
Porównywanie tablic							
%MW20:=EQUAL(%MW0:10;%MW10:10);	10 słów	142.90	99.14	100.85	47.31	49.02	17
	na jedno słowo		1.14	0.95	0.95	0.83	0.83
%MD20:=EQUAL(%MD0:10;%MD10:10);	10 słów podwójnych	155.56	110.06	111.81	56.02	57.77	17
	na słowo podwójne	2.33	2.00	2.00	1.69	1.69	
%MF20:=EQUAL_ARR	10 słów podwójnych	825	550.00	568.98	137.50	137.50	17
(%MF0:10;%MF10:10);	na słowo podwójne						
Wyszukiwanie							
%MW20 := FIND_EQW(%MW0:10,%KW0)	10 słów	374.68	240.00	267.60	97.00	97.00	14
%MD20 := FIND_EQD(%MD0:10, %KD0)	10 słów podwójnych	394.40	260.00	289.90	100.00	100.00	15
%MF20 := FIND_EQR(%MF0:10, %KF0)	10 słów podwójnych	936	624.00	645.53	135.70	135.70	15
%MF20 := FIND_EQRP(%MF0:10, %KF0)	10 słów podwójnych	936	624.00	645.53	135.70	135.70	15
%MD20 := FIND_GTR(%MF0:10, %KF0)	10 słów podwójnych	936	624.00	645.53	135.70	135.70	15
%MD20 := FIND_LTR(%MF0:10, %KF0)	10 słów podwójnych	936	624.00	645.53	135.70	135.70	15
Wyszukiwanie wartości max i min.							
%MW20 := MAX_ARW(%MW0:10)	10 słów	404.26	260.00	289.90	100.00	100.00	12
%MD20 := MAX_ARD(%MD0:10)	10 słów podwójnych	483.14	310.00	345.65	119.23	119.23	12
%MF20 := MAX_ARR(%MF0:10)	10 słów podwójnych	1555	1037.00	1072.78	261.20	261.20	12
%MD20 := MIN_ARR(%MF0:10)	10 słów podwójnych	1443	962.00	995.19	240.50	240.50	12
Liczba wystąpień							
%MW20:=OCCUR_ARW(%MW0:10,%KW0)	10 słów	394.40	260.00	289.90	100.00	100.00	14
%MD20:=OCCUR_ARD(%MD0:10,%KD0)	10 słów podwójnych	423.98	280.00	312.20	107.69	107.69	15
%MF20 := OCCUR_ARR(%MF0:10, %KF0)	10 słów podwójnych	1435	957.00	990.02	239.25	293.25	15
Przesuniecie okrężne							
ROL_ARW(word or value, %MWj:10)	10 słów	621.18	400.00	446.00	153.85	153.85	12
ROL_ARD(%MDi,%MDj:10)	10 słów podwójnych	670.48	430.00	479.45	165.38	165.38	12
ROL_ARR(%MFi,%MFj:10)	10 słów podwójnych	654	436.00	451.04	109.00	109.00	12
Sortowanie							
SORT_ARW(%MWi,%MWj:10)	10 słów	1 133.90	720.00	802.80	276.92	276.92	12
SORT_ARD(%MDi,%MDj:10)	10 słów podwójnych	700.06	440.00	490.60	169.23	169.23	12
SORT_ARR(%MFi,%MFj:10)	10 słów podwójnych	2150	1433.00	1482.44	358.25	358.25	12
Obliczanie długości							
LENGTH_ARW(tab_word)		71.42	50	55.74	19.28	19.28	12
LENGTH_ARD(tab_dword)		71.42	50	55.74	19.28	19.28	12
LENGTH_ARW(tab_real)		71.42	50	55.74	19.28	19.28	12
LENGTH_ARX(tab_bit)		71.42	50	55.74	19.28	19.28	12

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

8.3.9 Zarządzanie czasem

ST	Czas wykonywania (s)					Rozmiar (słowa)
	A	B RAM	B zas.	C RAM	C zas.	57xx

Data, czas, czas trwania

%MW2:4 := ADD_DT(%MW2:4,%MD8)	5 176.50	3 500.00	3 902.50	1203.20	1203.20	19
%MD2 := ADD_TOD(%MD2,%MD8)	2 435.42	1 640.00	1 828.60	630.77	630.77	9
%MB2:11 := DATE_TO_STRING(%MD40)	1 429.70	970.00	1 081.55	373.08	373.08	12
%MW5 := DAY_OF_WEEK()	552.16	390.00	434.85	108.45	108.45	5
%MD10 := DELTA_D(%MD2, %MD4)	1 725.50	1 170.00	1 304.55	450.00	450.00	9
%MD10 := DELTA_DT(%MD2:4,%MW6:4)	3 549.60	2 410.00	2 687.15	926.92	926.92	19
%MD10 := DELTA_TOD(%MD2,%MD4)	2 632.62	1 780.00	1 984.70	684.62	684.62	9
%MB2:20 := DT_TO_STRING(%MW50:4)	2 297.38	1 550.00	1 728.25	596.15	596.15	17
%MW2:4 := SUB_DT(%MW2:4,%MD8)	5 492.02	3 750.00	4 181.25	1203.20	1203.20	19
%MD2 := SUB_TOD(%MD2,%MD8)	2 622.76	1 780.00	1 984.70	684.62	684.62	9
%MB2:15 := TIME_TO_STRING(%MD40)	1 922.70	1 270.00	1 416.05	488.46	488.6	12
%MB2:9 := TOD_TO_STRING(%MD40)	1 281.80	830.00	925.45	319.23	319.23	12
%MD100 := TRANS_TIME(%MD2)	788.80	530.00	590.95	203.85	203.85	7

Dostęp do zegara czasu rzeczywistego

RRTC(%MW0:4)	164.81	111.44	112.19	46.43	47.18	8
WRTC(%MW0:4)	152.94	103.79	104.54	43.47	44.22	8
PTC(%MW0:5)	165.36	111.79	112.54	46.45	47.20	8
SCHEDULE(%MW0,%MW1,%MW2, %MD10,%MD12,%M0)	1500	1070	1190	411	411	8

8.3.10 Zegar

FTON	80.1	53.40	53.40	35.10	35.10	
FTOF	80.1	53.40	53.40	35.10	35.10	
FTP	80.1	53.40	53.40	35.10	35.10	
FPULSOR	406	290.4	323	112	112	

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

8.3.11 Łańcuchy znaków

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas.	C RAM	C zas.	57xx

Przypisywanie łańcucha znaków

%MB0:8:=%MB10:8	8 znaków	147.75	103.56	103.56	51.96	51.96	15
	na jeden znak	1.64	1.25	1.25	1.21	1.21	
%MB0:8:='abcdefg'	8 znaków	193.86	136.88	136.88	56.53	56.53	14
	na jeden znak	5.94	4.35	4.35	1.69	1.69	0.5

Konwersja Słowo <-> łańcuch znaków

%MW1:=STRING_TO_INT(%MB0:7)		145.69	104.31	104.52	37.93	38.14	10
%MB0:7:=INT_TO_STRING(%MW0)		149.67	109.21	109.42	35.52	35.73	10

Konwersja Słowo podwójne <-> łańcuch znaków

%MD1:=STRING_TO_DINT(%MB0:13)		1 408.43	1 061.01	1 061.01	364.71	364.71	10
%MB0:13:=DINT_TO_STRING(%MD0)		411.64	317.69	317.94	97.35	97.60	10

Konwersja Wartość zmiennoprzecinkowa <-> łańcuch znaków

%MF1:=STRING_TO_REAL(%MB0:15)		2 606.63	1 815.08	1 815.33	635.96	635.96	10
%MB0:15:=REAL_TO_STRING(%MF0)		1 084.46	752.94	753.27	389.47	389.81	10

Manipulacja na łańcuchach

%MB10:20 := CONCAT(%MB30:10,%MB50:10)	1 106.00	790.00	880.85	303.85	303.85	24
%MB10:20 := DELETE(%MB10:22,2,3);	896.00	640.00	713.60	246.15	246.15	21
%MW0 := EQUAL_STR(%MB10:20,%MB30:20);	756.00	540.00	602.10	207.69	207.69	19
%MW0 := FIND(%MB10:20,%MB30:10);	1 456.00	1 040.00	1 159.60	400.00	400.00	19
%MB10:20 := INSERT(%MB30:10,%MB50:10,4);	1 162.00	830.00	925.45	319.23	319.23	26
%MB10:20 := LEFT(%MB30:30,20);	826.00	590.00	657.85	226.92	226.92	19
%MW0 := LEN(%MB10:20);	490.00	350.00	390.25	134.62	134.62	12
%MB10:20 := MID(%MB30:30,20,10);	994.00	710.00	791.65	273.08	273.08	21
%MB10:20 := REPLACE(%MB30:20,%MB50:10,10,10);	1 246.00	890.00	992.35	342.31	342.31	28
%MB10:20 := RIGHT(%MB30:30,20);	1 358.00	970.00	1 081.55	373.08	373.08	19

8.3.12 Wydzielanie słów

LW	72	51.40	51.40	24.70	24.70	
HW	72	51.40	51.40	24.70	24.70	
CONCATW	72	51.40	51.40	24.70	24.70	

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

8.3.11 Funkcje specjalne i funkcje Orphee

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas	C RAM	C zas.	57xx
Komunikacja							
SEND_REQ(%KW0:6,15,%MW0:1,%MW10:10, %MW30:4)		2 800	2 000	2 230			33
SEND_TLG(%KW0:6,1,%MW0:5,%MW30:2)		2 100	1 500	1 673			24
Interfejs Człowiek - Maszyna							
SEND_MSG(ADR#1.0,%MW0:2,%MW10:2)		2 800	2 000	2 230	800		
25							
SEND_ALARM(ADR#1.0,%MW0:2,%MW10:2)		2 800	2 000	2 230	800		25
GET_MSG(ADR#1.0,%MW0:2,%MW10:2)		2 800	2 000	2 230	800		25
GET_VALUE(ADR#1.0,%MW0,%MW10:2)		1 400	1 000	1 115	400		20
ASK_MSG(ADR#1.0,%MW0:2,%MW10:2,%MW20:2)		2 800	2 000	2 230	800		32
ASK_VALUE(ADR#1.0,%MW0,%MW10:2,%MW20:2)		2 800	2 000	2 230	800		27
DISPLAY_ALRM(ADR#1.0,%MW0,%MW10:2)		1 400	1 000	1 115	400		20
DISPLAY_GRP(ADR#1.0,%MW0,%MW10:2)		1 400	1 000	1 115	400		20
DISPLAY_MSG(ADR#1.0,%MW0,%MW10:2)		1 400	1 000	1 115	400		20
CONTROL_LEDS(ADR#1.0,%MW0:2,%MW10:2)		2 800	2 000	2 230	800		25
ASSIGN_KEYS(ADR#1.0,%MW0:2,%MW10:2)		2 800	2 000	2 230	800		25
PANEL_CMD(ADR#1.0,%MW0:2,%MW10:2)		2 800	2 000	2 230	800		25
Sterowanie procesem							
PID("PIDS1","Unit",%IW3.5,%MW12,%M16, %MW284:43)	deval_mmi=0	1700	1100	1227			32
	deval_mmi=1	1500	900	1004			
PWM(%MW11,%Q2.1,%MW385:5)		700	500	557.5			17
SERVO(%MW12,%IW3.6,%Q2.2,%Q2.3, %MW284:43,%MW390:10)		1000	800	892			31
PID_MMI(ADR#0.0.4,%M1,%M2:5,%MW410:62)	EN=1	1400	1000	1115			30

A TSX 5710 i PCX 5710 **B** TSX5720/25/30/35,PMX5710,PCX5735 **C** TSX 57 40/45 i PMX 57 20/35/45

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas	C RAM	C zas.	57xx

Funkcje Orphee

DSHL_RBIT(%MD102,16,%MD204,%MD206)	czytanie 10 słów	493	320	357	123		17
DSHR_RBIT(%MD102,16,%MD204,%MD206)	zapis 10 słów	749	510	569	196		17
DSHRZ_C(%MD102,16,%MD204,%MD206)	"odbicie" 10 słów	493	310	346	119		17
WSHL_RBIT(%MW102,8,%MW204,%MW206)	wymiana 10 słów	365	220	245	85		17
WSHR_RBIT(%MW102,8,%MW204,%MW206)	20 bajtów	424	290	323	112		17
WSHRZ_C(%MW102,8,%MW204,%MW206)	20 bajtów	365	220	245	85		17
SCOUNT(%M100,%MW100,%M101,%M102, %MW101,%MW102,%M200,%M201, %MW200,%MW201)	20 bajtów	670	420	468	162		38

8.3.12 Jawna wymiana I/O

Read_Sts %CHi.MOD							
Dowolna aplikacja z wyjątkiem kanału komunikacyjnego procesora		997	712	748	316.4	316.4	2
Read_Sts %CHi							
Wejście dyskretne		462	330	347	147	147	6
Wyjście dyskretne		630	450	473	201	201	
Wejście analogowe		510	380	390	170	170	6
Wyjście analogowe		500	370	380	165	165	
CTY		518	370	389	165	165	
CFY		756	540	567	241	241	
CAY		532	380	399	170	170	
Write_Param %CHi							
Wejście analogowe		860	620	630	277	277	6
Wyjście analogowe		810	580	600	259	259	
CTY		1 134	810	851	362	362	
CFY		1 064	760	798	339	339	
CAY		784	560	588	250	250	

ST	Warunki	Czas wykonywania (s)					Rozmiar (słowa)
		A	B RAM	B zas.	C RAM	C zas.	57xx
Read_Param %CHi							
Wejście analogowe		180	120	130	54	54	6
Wyjście analogowe		180	120	130	54	54	
CTY		532	380	399	170	170	
CFY		644	460	483	205	205	
CAY		630	450	473	201	201	
Save_Param %CHi							
Wejście analogowe		1 300	880	890	393	393	6
Wyjście analogowe		1 300	890	900	397	397	
CTY		1722	1230	1292	549	549	
CFY		700	500	525	223	223	
CAY		700	500	525	223	223	
Restore_Param %CHi							
Wejście analogowe		800	570	590	254	254	6
Wyjście analogowe		800	570	590	254	254	
CTY		1148	820	861	366	366	
CFY		1092	780	819	348	348	
CAY		826	590	620	263	263	
Write_Cmd %CHi							
Wyjście dyskretne		448	320	336	143	143	6
Wejście analogowe							
. wymuszanie wejść		200	140	150	63	63	
. wymuszanie wejść		1 390	1 020	1 040	455	455	
Wyjścia analogowe: wymuszanie wyjść							
Wyjście dyskretne		210	150	150	67	67	
Smove %CHi							
CFY		1176	840	882	375	375	19
CAY		1148	820	861	366	366	

A TSX 57 10 i PCX 57 10 | **B** TSX 57 20/25/30/35 | **C** TSX 57 40/45 i PMX 57 20/35/45

8.3.13 Bloki własne użytkownika DFB

Obszar zajmowany przez bloki typu DFB

Rozmiar dla bloku DFB = Rozmiar zmiennych i parametrów DFB + Rozmiar kodu DFB

- Rozmiar zmiennych i parametrów DFB =
  110 + Suma opisów parametrów i zmiennych + Suma rozmiarów zajmowanych przez każdą zmienną i parametr

  Opis (deskryptor) parametru lub zmiennej = 5.5 + (Liczba znaków w nazwie zmiennej lub parametru)/2

  Rozmiar zajmowany przez zmienną lub parametr:

Typ	IN	IN/OUT	OUT	GLOBALNE	WŁASNE
EBOOL	0.5	2	0.5	0.5	0.5
BOOL	0.5	2	0.5	0.5	0.5
WORD	1	2	1	1	1
DWORD	2	2	2	2	2
REAL	2	2	2	2	2
AR_X	3	3	0.5*N	0.5*N	0.5*N
AR_W	3	3	N	N	N
AR_D	3	3	2*N	2*N	2*N
AR_R	3	3	2*N	2*N	2*N
STRING	3	3	0.5*N	0.5*N	0.5*N

  N = liczba elementów tablicy lub długość łańcucha znaków (STRING)

- Rozmiar kodu DFB = 11 + Suma rozmiarów wszystkich instrukcji (1)

  (1) Do rozmiaru instrukcji należy dodać następujące wartości (w zależności od parametru lub zmiennej):

  Obiekty nie indeksowane

Opis	Typ	Wartość
IN	EBOOL	0.5
	BOOL, WORD, DWORD, REAL	0
	AR_X,AR_W,AR_D,AR_R,STRING	3
IN/OUT	EBOOL	3
	BOOL, WORD, DWORD, REAL	3
	AR_X,AR_W,AR_D,AR_R,STRING	3
OUT,GLOBALNE	EBOOL	0.5
WŁASNE	BOOL, WORD, DWORD, REAL	0
	AR_X,AR_W,AR_D,AR_R,STRING	0

Obiekty indeksowane

Opis	Typ	Wartość
IN	AR_X,AR_W,AR_D,AR_R,	7
IN/OUT	AR_X,AR_W,AR_D,AR_R,	7
OUT, GLOBALNE, WŁASNE	AR_X,AR_W,AR_D,AR_R,	6

Rozmiar zajmowany przez użytkownika bloku DFB

- wywołanie bloku zastępczego DFB bez żadnych parametrów: 6 słów
- obliczenia dla parametru:

Opis	Typ	Wartość
IN	EBOOL,BOOL,WORD,DWORD,REAL	idem := przypisanie
	AR_X,AR_W,AR_D,AR_R,STRING	14
IN/OUT	EBOOL,WORD,DWORD,REAL	10
	BOOL,AR_X,AR_W,AR_D,AR_R,	14
OUT	Wszystkie rodzaje	idem := przypisanie

- wykorzystanie zmiennej bloku zastępczego (*instance*): należy dodać 1 słowo z uwzględnieniem %M.

Czas wykonywania

Całkowity czas wykonywania bloku DFB = Próg dla kodu DFB + Suma czasów dostępu do zmiennych i parametrów bloku DFB + Wywołanie bloku DFB (bez parametrów) + Suma czasów dostępu dla parametrów

	Czas wykonywania (s)				
	A	**B RAM**	**B zas.**	**C RAM**	**C zas.**
Próg dla kodu DFB	16.24	7.00	8.00		
Dostęp do zmiennej lub parametru DFB (1)					
Obiekty indeksowane					
• IN					
- EBOOL	0.290	0.125	0.208		
- BOOL,WORD,DWORD,REAL	0	0	0		
- AR_X,AR_W,AR_D,AR_R,STRING	1.739	0.750	1.282		
• IN/OUT (wszystkie typy)	1.739	0.750	1.282		
• OUT,PUBLIC,PRIVATE					
- EBOOL	0.290	0.125	0.208		
- BOOL,WORD,DWORD,REAL	0	0	0		
- AR_X,AR_W,AR_D,AR_R,STRING	0	0	0		
Obiekty nie indeksowane					
• IN, IN/OUT					
- AR_X,AR_W,AR_D,AR_R,	3.496	1.5	1.472		
• OUT,PUBLIC,PRIVATE					
- AR_X,AR_W,AR_D,AR_R,	2.616	1.125	1.931		
WywołaniaDFB (bez param.)	4.770	2.125	3.018		
Wartości przypadające na jeden parametr (1)					
• IN					
- EBOOL		idem :=			
- BOOL,WORD,DWORD,REAL		idem :=			
- AR_X,AR_W,AR_D,AR_R,STRING	4.393	2.125	3.269		
• IN/OUT					
- EBOOL,WORD,DWORD,REAL	3.48	1.5	2.318		
- BOOL,AR_X,AR_W,AR_D,AR_R,STRING	4.393	2.125	3.269		
• OUT wszytkie typy		idem :=			

(1) Wartość jest różna w zależności od operacji przypisanych obiektom typu %M

A	TSX 57 10 i PCX 57 10	**B**	TSX 57 20/25/30/35	**C**	TSX 57 40/45 i PMX 57 20/35/45

8.4 Rozmiar aplikacji

8.4.1 Opis stref pamięci

Pamięć aplikacji jest podzielona na kilka stref pamięci:

∞ strefa pamięci bitowej:
- ta strefa występuje oddzielnie tylko w TSX 37 i jest ograniczona do 1280 bitów,
- w sterownikach TSX 57 jest ona częścią strefy pamięci danych,

∞ strefa pamięci danych (słowa),

∞ strefa aplikacji, zawierającą:
- konfigurację,
- program,
- stałe.

Obszary pamięci bitowej i pamięci danych zawsze lokowane są w wewnętrznej pamięci RAM, natomiast obszar aplikacji może być lokowany w wewnętrznej pamięci RAM lub w zasobniku pamięci (*memory card*).
Struktura pamięci została opisana w rozdziale 1.3 części A.

8.4.2 Obszar pamięci zajmowany przez obiekty PL7

	Pamięć bitowa (w słowach)	Dane (w słowach)	Aplikacja (w słowach)
Kroki *Grafcet-u* (%Xi, %Xi.T)	0.5	1	
%Mi	0.5		
Obiekty numeryczne (%MWi)		1	
Stałe (%KWi)			1.25
%NWi		1	
%Ti		4	2
%TMi		5	2
%MNi		4	2
%Ci		3	1
%Ri (długość lg)		6+lg	2
%DRi		6	49

Dane interpretera *Grafcet-u* = 355 + 2 x Liczba skonfigurowanych kroków aktywnych + (Liczba skonfigurowanych przejść aktywnych) / 2

8.4.3 Rozmiar pamięci modułu

Uwaga

Ta informacja podawana jest dla konkretnej wersji procesora. Moze ona ulegać niewielkim zmianom wraz z rozwojem produktów.

Poniżej określono, dla każdego rodzaju modułu, rozmiar zajmowanej przez niego pamięci (z rozbiciem na poszczególne obszary pamięci) oraz wartości jakie należy dodać, do wartości z tabeli zużycia pamięci, przy pierwszym zastosowaniu funkcji specjalnej.

Tabela zużycia pamięci modułu dla TSX 37

Procesory	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
TSX 37-10	70	1560	920
TSX 37-21	70	1570	930
TSX 37-22	70	2110	1280
Zastos. zadania FAST (TSX 37)		260	
Wyk. pierwszego zdarzenia (TSX 37)		520	

Grupa urz. dyskretnych	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
8 wejść dyskretnych	4	12	40
16 wejść dyskretnych	8	12	50
4 wyjścia dyskretne	2	12	40
8 wyjść dyskretnych	4	12	40
16 wejść dyskretnych / 12 wyjść dyskr.	16	20	100
32 wejścia dyskr./ 32 wyjścia dyskr.	32	20	142

Rodzina wejść 4 ANA	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
AEZ414	0	156	56
Wartość dodatkowa dla 1-go modułu w grupie wejść 4 ANA			120

Rodzina wejść 8 ANA	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
AEZ801/AEZ802	0	212	72
Wartość dodatkowa dla 1-go modułu w grupie wejść 8 ANA			120

Rodzina wyjść ANA	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
ASZ200	0	52	40
ASZ401	0	100	59
Wartość dodatkowa dla 1-go modułu w grupie wyjść ANA			120

Rodzina liczników	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
CTY1A	16	108	64
CTY2A	32	212	106
Wart. dod. dla 1-go kanału licznika UP			144
Wart. dod. dla 1-go kanału licznika DN			144
Wartość dodatkowa dla 1-go kanału licznika 2-kierunkowego *Up*/*Down*			144

Rodzina urz. komunikacyjnych	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
STZ010	0	36	168
SCP111/ SCP112/ SCP114 (w UTW CPU)	0	40	763
FPP 20 w CPU (Kanał 0 UTW)	0	40	755

Tabela zużycia pamięci modułu dla TSX/PCX/PMX 57

Procesory	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
TSX/PCX/PMX 57-10	70	4714	1720
TSX 57-20/25/30/35/40/45 PCX 57-10/35 PMX 57-10/20/35/45	70	4714	1784
Wykonanie zadania FAST (TSX 57)		520	
Wart. dod. dla 1-go modułu w konfig.		600	

Procesory PMX: pętla sterująca	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
PMX 57-10 : na 1 pętlę		500	
Wart. dodatkowa dla 1-ej pętli			25000
PMX 57-20/35/45 : na 1 petlę		500	
Wart. dodatkowa dla 1-ej pętli			5000

Rodzina pojedynczych wejść dyskretnych	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
8 wejść dyskretnych	4	100	100
16 wejść dyskretnych	8	130	110
32 wejścia dyskretne	16	230	120
64 wejścia dyskretne	32	430	190
Wartość dodatkowa dla 1-go modułu pojedynczych wejść dyskretnych			610

Rodzina pojedynczych wyjść dyskretnych	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
8 wyjść dyskretnych	4	110	100
16 wyjść dyskretnych	8	160	110
32 wyjścia dyskretne	16	280	120
64 wyjścia dyskretne	32	550	190
Wartość dodatkowa dla 1-go modułu pojedynczych wyjść dyskretnych			570

Dyskretne wejścia wyzwalane zdarzeniami	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
16 wejść dyskretnych (DEY 16FK)	8	220	130
Wartość dodatkowa dla 1-go modułu wejść dyskretnych EVT			680

Rodzina mieszanych I/O	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
16 wejść/12 wyjść (DMY 28FK)	16	304	152
Wart. dodatkowa dla 1-go modułu rodziny mieszanych urządzeń I/O			1432

Rodzina wejść analogowych	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
AEY414	4	430	160
AEY800	8	840	240
AEY1600	16	1670	430
Wart. dodatkowa dla 1-go modułu wejść analog. (AEY 414/800/1600)			2990
AEY810	8	888	248
AEY1614	16	1768	432
Wart. dodatkowa dla 1-go modułu wejść analog. (AEY 810/1614)			3056
AEY420	4	476	168
Wart. dodatkowa dla 1-go modułu wejść analog. (AEY 810/1614)			2080

Rodzina wyjść analogowych	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
ASY410	4	430	160
Wart. dodatkowa dla 1-go modułu wyjść analogowych ASY410			1700
ASY800	8	744	248
Wart. dodatkowa dla 1-go modułu wyjść analogowych ASY800			1760

Rodzina liczników	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
Moduł CTY2A	32	410	170
Moduł CTY4A	64	800	250
Wart. dodatkowa dla 1-go skonfigurowanego kanału licznika			1740
CTY2C module	48	672	184
Wart. dodatkowa dla 1-go skonfigurowanego kanału licznika			1992

Rodzina Servo-Motor	**Pamięć bitowa (słowa)**	**Dane (słowa)**	**Strefa aplikacji (słowa)**
CAY21	78	1050	280
CAY41	156	2090	480
Wart. dodatkowa dla 1-go skonfigurowanego kanału serwomotoru			2150

Rodzina silników krokowych	**Pamięć bitowa (słowa)**	**Dane (słowa)**	**Strefa aplikacji (słowa)**
CFY11	29	323	104
CFY21	58	646	152
Wart. dodatkowa dla 1-go skonfigurowanego kanału silnika krokowego			2368

Rodzina modułów komunikacyjnych	**Pamięć bitowa (słowa)**	**Dane (słowa)**	**Strefa aplikacji (słowa)**
SCY21600 (Kanał 0 UTW)	1	230	80
SCP111/ SCP112 / SCP114 (UTW) w SCY21600 (Kanał 1 UTW)	1	450	40
Wart. dodatkowa dla 1-go kanału skonfigurowanego jako UTW			1280
ETY 110	1	431	256
Wart. dod. dla 1-go kanału ETY 110			1984
IBY 100	1	450	40

Rodzina modułów komunikacyjnych wspomagających procesor	**Pamięć bitowa (słowa)**	**Dane (słowa)**	**Strefa aplikacji (słowa)**
SCP111/ SCP112/ SCP114 (UTW) w CPU (Kanał 0 UTW)	1	60	580
FPP 20 w CPU (Kanał 0 UTW)	1	60	580
FPP 10 w CPU (Kanał 0 UTW)	1	40	870

Rodzina modułów ASI	**Pamięć bitowa (słowa)**	**Dane (słowa)**	**Strefa aplikacji (słowa)**
SAY	3	373	176
Wart. dodatkowa dla 1-go skonfigurowanego kanału ASi			2272

Rodzina modułów wagowych	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
AWY001	1	170	120
Wart. dodatkowa dla 1-go skonfigurowanego kanału wagi			3920

Rodzina TBX	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
Wejścia dyskretne	8	152	88
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			1400
Wyjścia dyskretne	8	176	88
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			1320
Programowalny moduł podrzędny	8	160	88
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			2304
Przerzutnik	8	160	88
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			1400
AES 400	2	270	104
ASS 200	2	270	104
AMS 620	4	508	112
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			3968

Rodzina Momentum	Pamięć bitowa (słowa)	Dane (słowa)	Strefa aplikacji (słowa)
Wejścia	16	96	72
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			1384
Wyjścia	16	112	72
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			1256
Mieszane	16	104	72
Wartość dodatkowa dla 1-ej skonfigurowanej podgrupy (*sub-base*)			1424

8.4.4 Rozmiar pamięci zajmowanej przez funkcje złożone

Zamieszczone poniżej tabele zawierają informacje o rozmiarze pamięci zajmowanej przez kod funkcji złożonej (OF), umieszczony w aplikacji (obszar pamięci zwany pamięcią aplikacji), gdy następuje wywołanie tej funkcji.
Funkcje pochodzące z jednej rodziny korzystają z tego samego kodu (kod wspólny). Ten kod wspólny jest ładowany do sterownika podczas **pierwszego wywołania** funkcji z danej rodziny. Kod specyficzny dla funkcji jest ładowany podczas **pierwszego wywołania** danej funkcji.

Przykład:

- Pierwsze wywołanie funkcji z grupy funkcji konwersji numerycznych, czyli np. DBCD_TO_DINT
  Kod ładowany do strefy aplikacji:
  - kod wspólny = 154 słowa
  - kod funkcji (OF) DBCD_TO_INT = 149 słów
- Wywołanie innej funkcji z rodziny funkcji konwersji numerycznych: DINT_TO_DBCD
  Kod ładowany do strefy aplikacji:
  - Kod funkcji (OF) DINT_TO_DBCD = 203 words
  Wywołanie funkcji z rodziny funkcji konwersji numerycznych, która była już wcześniej wywoływana (DBCD_TO_DINT lub DINT_TO_DBCD): nie jest ładowany żaden kod

Konwersja numeryczna	OF	Rozmiar kodu (w słowach)
Konwersja 32-bit. liczby BCD na 32-bit. l. całkowitą	DBCD_TO_DINT	145
Konwersja 32-bit. liczby BCD na 16-bit. l. całkowitą	DBCD_TO_INT	149
Konwersja 32-bit. l. całkowitej na 32-bit. liczbę BCD	DINT_TO_DBCD	203
Konwersja 16-bit. l. całkowitej na 32-bit. liczbę BCD	INT_TO_DBCD	75
Wydzielenie mniej ważnego słowa ze słowa pod.	LW	33
Wydzielenie ważniejszego słowa ze słowa podwójnego	HW	33
Utworzenie słowa podwójnego ze dwóch słów	CONCATW	33
	kod wspólny	154

Ciąg bitów	OF	Rozmiar kodu (w słowach)
Koniunkcja dwóch tablic	AND_ARX	209
Kopiowanie tablicy bitowej do tablicy słów podwójnych	BIT_D	248
Kopiowanie tablicy bitowej do tablicy słów	BIT_W	205
Kopiowanie tablicy bitowej do tablicy bitowej	COPY_BIT	146
Kopiowanie tablicy słów podwójnych do tablicy bitowej	D_BIT	196
Logiczne dopełnienie tablicy	NOT_ARX	157
Alternatywa dwóch tablic	OR_ARX209	
Kopiowanie tablicy słów do tablicy bitowej	W_BIT	195
Nierównoważność dwóch tablic XOR	XOR_ARX	209
Długość tablicy (liczba elementów)	LENGTH_ARX	20
	kod wspólny	427

Operacje na tablicy słów	OF	Rozmiar kodu (w słowach)
Wyszukiwanie 1-go elementu równego podanej wartości	FIND_EQW	75
Wyszuk.1-go elementu większego od podanej wartości	FIND_GTW	75
Wyszuk.1-go elementu mniejszego od podanej wartości	FIND_LTW	78
Wyszukiwanie maksymalnej wartości w tablicy	MAX_ARW	78
Wyszukiwanie minimalnej wartości w tablicy	MIN_ARW	74
Liczba wystąpień danej wartości w tablicy	OCCUR_ARW	145
Okrężne przesunięcie w lewo	ROL_ARW	150
Okrężne przesunięcie w prawo	ROR_ARW	144
Sortowanie tablicy (porządek rosnący lub malejący)	SORT_ARW	164
Wyszuk. w części tablicy 1-go elemetu = podanej wartości	FIND_EQWP	77
Długość tablicy (liczba elementów)	LENGTH_ARW	20
	kod wspólny	162

Operacje na tablicach słów podwójnych	OF	Rozmiar kodu (w słowach)
Wyszukiwanie 1-go elementu równego podanej wartości	FIND_EQD	79
Wyszuk.1-go elementu większego od podanej wartości	FIND_GTD	80
Wyszuk.1-go elementu mniejszego od podanej wartości	FIND_LTD	95
Wyszukiwanie maksymalnej wartości w tablicy	MAX_ARD	95
Wyszukiwanie minimalnej wartości w tablicy	MIN_ARD	78
Liczba wystąpień danej wartości w tablicy	OCCUR_ARD	163
Okrężne przesunięcie w lewo	ROL_ARD	170
Okrężne przesunięcie w prawo	ROR_ARD	178
Sortowanie tablicy (porządek rosnący lub malejący)	SORT_ARD	
Wyszuk. w części tablicy 1-go elemetu = podanej wartości	FIND_EQWP	77
Długość tablicy (liczba elementów)	LENGTH_ARW	20
	kod wspólny	162

Operacje ne tablicach zmiennoprzecinkowych	OF	Rozmiar kodu (w słowach)
Suma elementów w tablicy liczb rzeczywistych	SUM_ARR	152
Wyszukiwanie 1-go elementu równego podanej wartości	FIND_EQR	134
Wyszukiwanie 1-go elementu równego wartości z danego rzędu	FIND_EQRP	135
Wyszuk. 1-go elementu większego od podanej wartości	FIND_GTR	134
Wyszuk. 1-go elementu mniejszego od podanej wartości	FIND_LTR	134
Wyszukiwanie wartości maksymalne w tablicy	MAX_ARR	161
Wyszukiwanie wartości minimalne w tablicy	MIN_ARR	162
Liczba wystąpień w tablicy podanej wartości	OCCUR_ARR	132
Okrężne przesunięcie w lewo w tablicy	ROL_ARR	167
Okrężne przesunięcie w prawo w tablicy	ROR_ARR	173
Sortowanie tablicy (porządek malejący lub rosnący)	SORT_ARR	271
Porównywanie dwóch tablic rzeczywistych	EQUAL_ARR	173
Długość tablicy (liczba elementów)	LENGTH_ARR	20
	kod wspólny	124

Data, czas, okres czasu	OF	Rozmiar kodu (słowa)
Dodanie okresu czasu do pełnej daty	ADD_DT	519
Dodanie czasu do pory dniaADD_TOD	188	
Konwersja daty na łańcuch znaków	DATE_TO_STRING	150
Dzień tygodnia (nazwa)	DAY_OF_WEEK	99
Różnica pomiędzy dwiema datami	DELTA_D	374
Różnica pomiędzy dwiema pełnymi datami	DELTA_DT	547
Różnica pomiędzy dwiema porami dnia	DELTA_TOD	110
Konwersja pełnej daty na łańcuch znaków	DT_TO_STRING	266
Odjęcie okresu czasu od pełnej daty	SUB_DT	548
Odjęcie czasu od pory dnia	SUB_TOD186	
Konwersja okresu czasu na łańcuch znaków	TIME_TO_STRING	413
Konwersja pory dnia na łańcuch znaków	TOD_TO_STRING	156
Przeliczenie okresu czasu na format godz.-min.-sek.	TRANS_TIME	211
Funkcja zegara czasu rzeczywistego	SCHEDULE	700
	kod wspólny	1703

Operacje na łańcuchach znaków	OF	Rozmiar kodu (słowa)
Łączenie dwóch łańcuchów znaków	CONCAT	224
Usuwanie podanego ciągu znaków z łańcucha	DELETE	279
Wyszukiwanie 1-ego znaku różniącego łańcuchy	EQUAL_STR	212
Wyszukiwanie ciągu znaków w łańcuchu	FIND	225
Wstawienie ciągu znaków do łańcucha	INSERT	287
Wydzielenie z łańcucha części znaków (z lewej strony)	LEFT	38
Długość łańcucha	LEN	70
Wydzielanie ciągu znaków z łańcucha	MID	44
Wymiana ciągu znaków w łańcuchu	REPLACE365	
Wydzielenie z łańcucha części znaków (z prawej strony)	RIGHT	55
	kod wspólny	418

Funkcje Orphee	OF	Roz. kodu (słowa)
Przes. w lewo w słowie 32-bit. z odzysk. przesuniętych bitów	DSHL_RBIT	152
Przesunięcie w prawo w słowie 32-bit. z dodaniem znaku i odzyskaniem przesuniętych bitów	DSHR_RBIT	152
Przesunięcie w prawo w słowie 32-bit. z wypełnieniem pustych miejsc zerami i odzyskaniem przesuniętych bitów	DSHRZ_C	133
Przes. w lewo w słowie 16-bit. z odzysk. przesuniętych bitów	WSHL_RBIT	91
Przesunięcie w prawo w słowie 16-bit. z dodaniem znaku i odzyskaniem przesuniętych bitów	WSHR_RBIT	103
Przesunięcie w prawo w słowie 32-bit. z wypełnieniem pustych miejsc zerami i odzyskaniem przesuniętych bitów	WSHRZ_C	90
	kod wspólny	173
Licznik 2-kier. z sygnalizacją przekroczenia limitu	SCOUNT	617
Przesunięcie okrężne w lewo, w słowie	ROLW	41
Przesunięcie okrężne w prawo, w słowie	RORW	41
Przesunięcie okrężne w lewo, w słowie podwójnym	ROLD	49
Przesunięcie okrężne w prawo, w słowie podwójnym	RORD	49

Zegar opóźniający	OF	Rozmiar kodu (w słowach)
Wyjście przebiegu prostokątnego	FPULSOR	215
Opóźnienie typu *Off-delay*	FTOF	272
Opóźnienie typu *On-delay*	FTON	217
Impuls opóźniający	FTP	245

Logarytm, funkcje trygonometryczne i wykładnicze	OF	Rozmiar kodu (w słowach)
Logarytm naturalny	LN	0
Logarytm dziesiętny	LOG	0
Exponent	EXP	0
Funkcja liczby rzeczywistej z wykładnikiem całkowitym	EXPT	523
Część całkowita liczby	TRUNC	128
Kosinus kąta (w radianach)	COS	0
Sinus kąta (w radianach)	SIN	0
Tangens kąta (w radianach)	TAN	0
Arc cos (wynik w przedziale od 0 do π)	ACOS	0
Arc sin (wynik w przedziale od $-\pi/2$ do $\pi/2$)	ASIN	0
Arc tan (wynik pomiedzy $-\pi/2$ a $\pi/2$)	ATAN	0
Konwersja stopni na radiany	DEG_TO_RAD	257
Konwersja radianów na stopnie	RAD_TO_DEG	247
	kod wspólny	392

Funkcje specjalne

Funkcje sterowania procesem	OF	Rozmiar kodu (w słowach)
Regulator mieszany PID	PID	1800
Impuls z modulacją szerokości (wartość numeryczna)	PWM	600
Wyjście PID do sterowania zaworem dyskretyzowanym	SERVO	1200
Zarządzanie CCX17 dedykowanego funkcji MMI dla sterowania pętlami PID	PID_MMI	4400
	kod wspólny	573

Interfejs MMI (Człowiek - Maszyna)	OF	Rozmiar kodu (w słowach)
Blokowanie (łączenie w bloki) zmiennych w CCX17	ASK_MSG,	46.5
Blokowanie zmiennych w komunikacie znajdującym się w CCX17	ASK_VALUE,	46.5
Dynamiczne przypisywanie funkcji klawiszom	ASSIGN_KEYS,	46.5
sterowanie funkcjami diod LED	CONTROL_LEDS,	46.5
Wyświetlanie alarmu znajdującego się w CCX17	DISPLAY_ALARM,	46.5
Wyśw. grup komunikatów znajdujących się w CCX17	DISPLAY_GRP,	46.5
Wyświetlanie komunikatu znajdującego się w CCX17	DISPLAY_MSG,	46.5
Wielokrotne wprowadzenie zmiennnej do CCX17	GET_MSG,	46.5
Wielokrotne wprowadzenie zmiennych w komunikacie znajdującym się w CCX17	GET_VALUE,	46.5
Przesłanie komendy do CCX17	PANEL_CMD,	46.5
Wyświetlenie komunikatu o alarmie znajdującego się w pamięci sterownika	SEND_ALARM,	46.5
Wyśw. komunikatu znajdującego się w pamięci ster.	SEND_MSG	46.5
	kod wspólny	573

Funkcje komunikacyjne	OF	Rozmiar kodu (w słowach)
Czytanie podstawowych obiektów językowych	READ_VAR	617
Zapisywanie podstawowych obiektów językowych	WRITE_VAR	500
Wysyłanie/odbieranie żądań UNI-TE	SEND_REQ	438
Wysyłanie i/lub odbieranie danych	DATA_EXCH	375
Wysłanie łańcucha znaków	PRINT_CHAR	476
Żądanie odczytania łańcucha znaków	INPUT_CHAR	625
Wysyłanie i/lub odbieranie łańcucha znaków	OUT_IN_CHAR	531
Wysłanie telegramu	SEND_TLG	219
Odbieranie telegramu	RCV_TLG	172
Żądanie zatrzymania (stop) trwającej komunikacji	CANCEL	133
	kod wspólny	506
Przesunięcie 1 bajtu w prawo, w tablicy bajtów	ROR1_ARB	235

Funkcje sterowania	OF	Rozmiar kodu (w słowach)
Automatyczne sterowanie ruchem	SMOVE	0

Wymiana jawna (1)	OF	Rozmiar kodu (w słowach)
Odczytanie parametrów statusu	READ_STS	0
Odczytanie parametrów regulacyjnych (*adjustment*)	READ_PARAM	0
Uaktualnienie parametrów regulacyjnych	WRITE_PARAM	0
Zapisanie parametrów regulacyjnych	SAVE_PARAM	0
Odtworzenie parametrów regulacyjnych	RESTORE_PARAM	0
Zaktualizowanie parametrów komend	WRITE_CMD	0

(1) Specyficzny kod funkcji (OF) zawiera się w rozmiarze modułu I/O.

8.5 Dodatek: metoda obliczania liczby instrukcji

Opisana poniżej metoda umożliwia obliczenie liczby instrukcji logicznych i numerycznych (kod asemblera).

Uwaga: Za pomocą tej właśnie metody obliczono parametry zamieszczone w rozdziale 1.3 w części A niniejszej dokumentacji.

Obliczanie liczby instrukcji logicznych
Pod uwagę bierze się liczbę każdego z wymienionych poniżej elementów:
∞ operacje logiczne: ładowanie *load* (LD), AND, OR, XOR, ST, itp.
∞ zamknięte nawiasy (lub zamknięcia ścieżek w *Ladder*: pionowe łączniki zamykające)
∞ bloki porównywania (AND[...], OR[...], itp.) oraz operacyjne ([...])

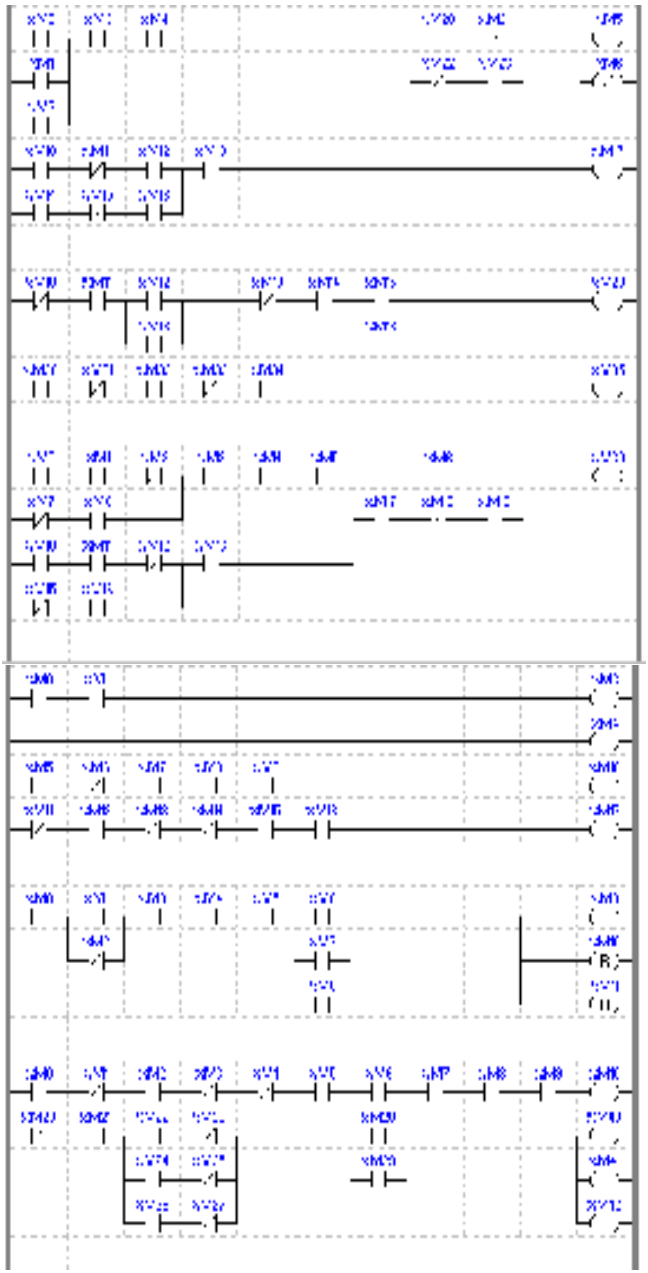
Operatory NOT, RE i FE nie są traktowane jak operacje logiczne.

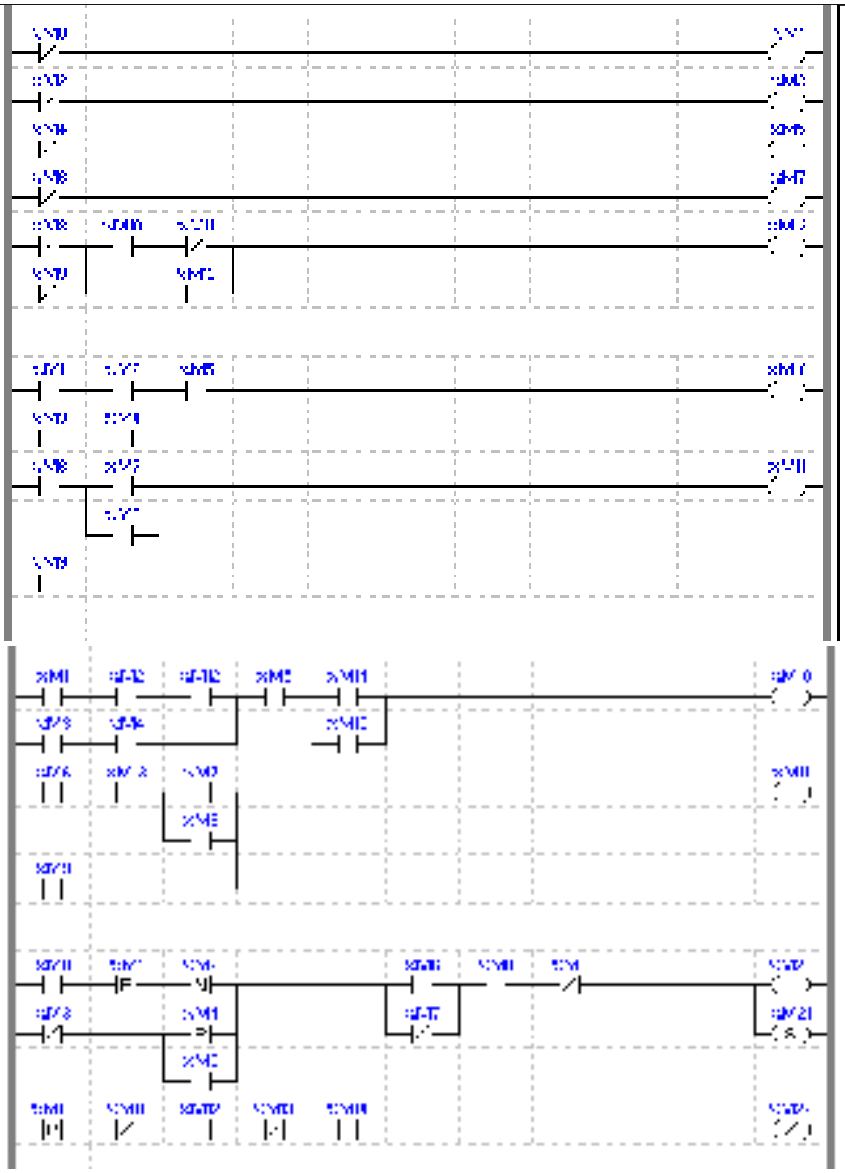
Przykład:
LD %M0
AND(%M1
OR %M2
)
ST %M3
= 5 instrukcji logicznych

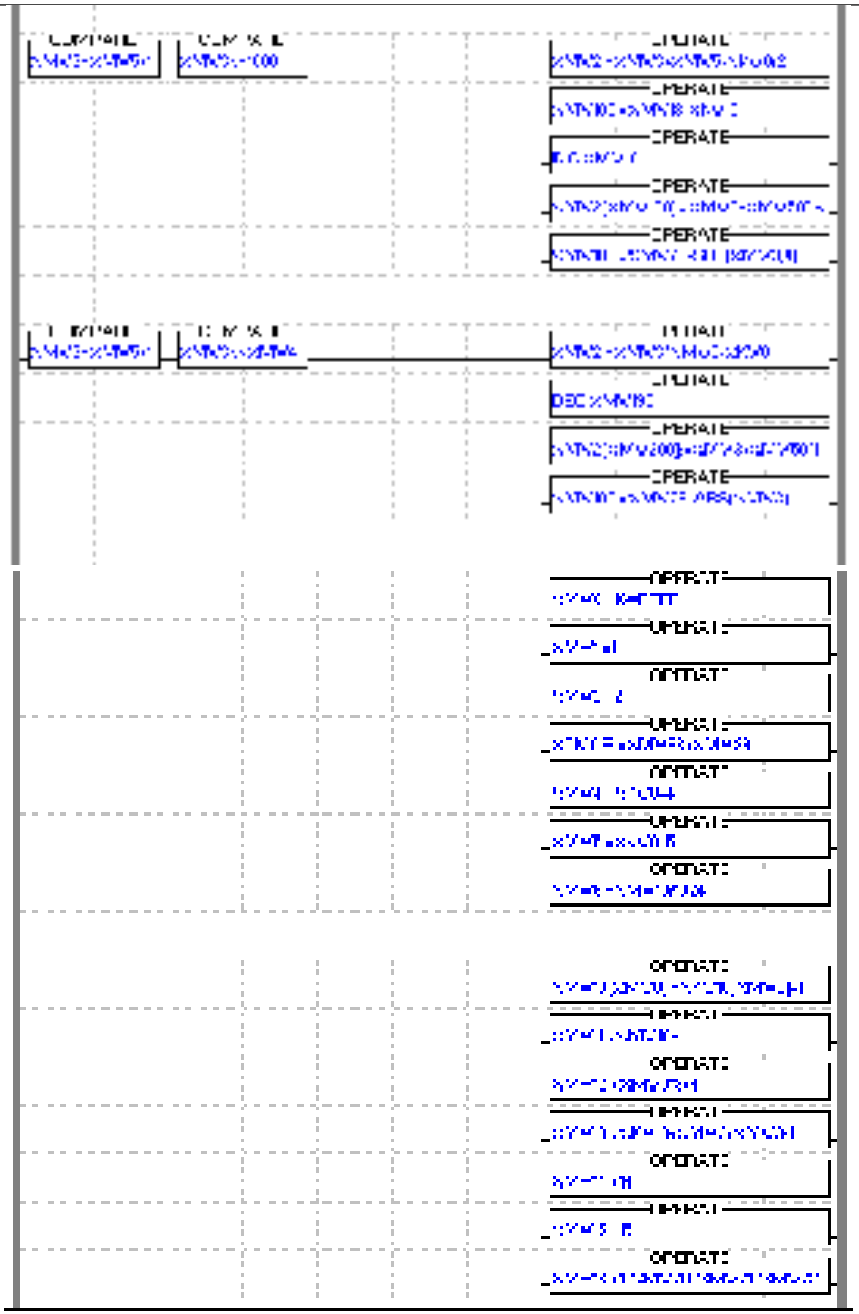
Obliczanie liczby instrukcji numerycznych
Uwzgledniana jest liczba następujących elementów:
∞ przypisania (:=)
∞ załadowanie pierwszej wartości za :=
∞ operacje arytmetyczne (+, -, *, /, <, =, itp.), operacje na słowach, tablicach słów, słowach podwójnych, wartościach zmiennoprzecinkowych
∞ operacje logiczne na słowach
∞ funkcje (OF, EQUAL, itp.) bez względu na liczbę parametrów
∞ bloki funkcyjne (lub instrukcje bloków logicznych)

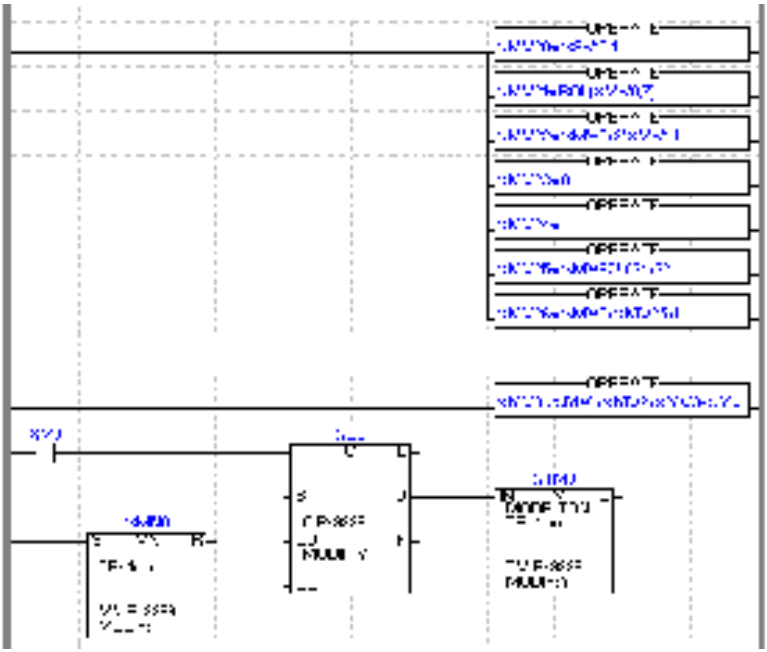
Przyklad: %MW0:=(%MW1+%MW2)*%MW3;
liczba instrukcji:
:=
%MW1 (odpowiada instrukcji ładowania)
+
*
czyli - 4 instrukcje.

Przykład aplikacji (65% instrukcji logicznych i 35% numerycznych):









(1):%MW0:=%MW1+%MW2+%MW3+%MW4+%MW5+%MW6+%MW7+%MW8+%MW9+%MW10+1

Wynik

	Liczba instrukcji	%	
Instrukcje Logiczne bez zbocza	187	54.05%	64.16%
Instrukcje logiczne ze zboczem	4	1.16%	
Blok operacyjny	31	8.96%	
Blok funkcyjny	3	0.87%	35.84%
Pojed. arytmetyka (+,-,:=,AND,...)	111	32.08%	
Arytmetyka indeksowana	4	1.16%	
*,/	6	1.73%	
Wartości bezpośrednie	24		
Całkowita liczba	346		100.00%

REF. **TLX DR PL7 30E**

ART. **87727**

Schneider Automation Inc.
One High Street
North Andover, MA 01845
Tél.: (1) 508 794 0800
Fax : (1) 508 975 9010

Schneider Automation S.A.
245, route des Lucioles - BP 147
F-06903 Sophia Antipolis
Tél. : (33) (0)4 92 96 20 00
Fax : (33) (0)4 93 65 37 15

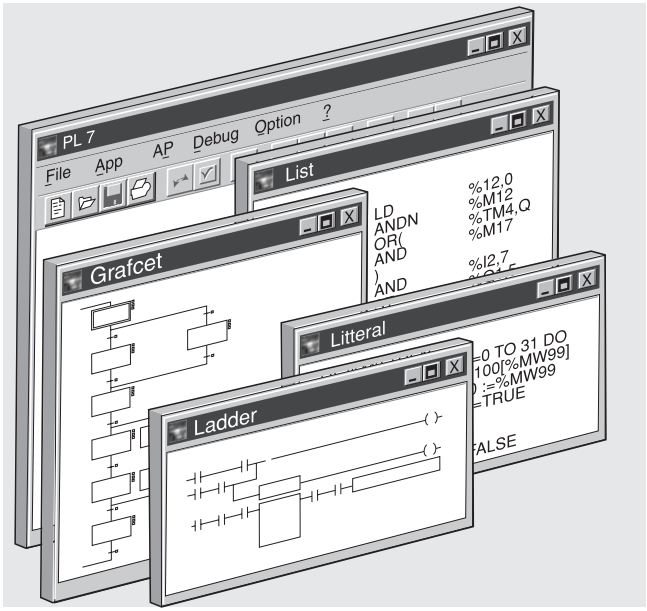
Schneider Automation GmbH
Steinheimer Straße 117
D-63500 Seligenstadt
Tél. : (49) 6182 81 2584
Fax : (49) 6182 81 2860

Tłumaczenie - BATI (0-63)243-45-81

Lipiec 1998 (837)

PL7 Micro / Junior / Pro

Instrukcja Użytkownika
(wersja V3. ■)



GROUPE SCHNEIDER

■ Merlin Gerin ■ Modicon ■ Square D ■ Telemecanique

REF. **TLX DR PL7 30E**

ART. **87727**

Schneider Automation Inc.
One High Street
North Andover, MA 01845
Tél.: (1) 508 794 0800
Fax : (1) 508 975 9010

Schneider Automation S.A.
245, route des Lucioles - BP 147
F-06903 Sophia Antipolis
Tél. : (33) (0)4 92 96 20 00
Fax : (33) (0)4 93 65 37 15

Schneider Automation GmbH
Steinheimer Straße 117
D-63500 Seligenstadt
Tél. : (49) 6182 81 2584
Fax : (49) 6182 81 2860

Tłumaczenie - BATI (0-63)243-45-81

Lipiec 1998 (837)