SPIS TREŚCI

1. Podstawowe wiadomości dotyczące sterowników SIMATIC S7 - 200

SIMATIC S7-200 firmy Siemens jest nazwą nowej rodziny sterowników programowalnych (PLC = *ang. **p**rogrammable **l**ogic **c**ontroller*) zaprojektowanych do realizacji małych i średnich zadań sterowania automatycznego. Dzięki temu istnieje możliwość łatwego i masowego wprowadzania nowoczesnych układów kontroli do najprostszych maszyn czy linii technologicznych. Charakteryzują się one małymi wymiarami, zwartą budową oraz potężnym zestawem instrukcji. Koncepcja ich budowy oparta jest na sprawdzonej strukturze sterowników przemysłowych SIMATIC S5.

Głównym elementem każdego sterownika jest tzw. centralna jednostka przetwarzająca (CPU = *ang. **c**entral **p**rocessing **u**nit*) nadzorująca pracą sterownika. Rodzaj użytej jednostki centralnej stanowi podstawę podziału sterowników S7. Najprostszymi typami sterowników S7 są: sterowniki z jednostką CPU 212 oraz sterowniki z jednostką CPU 214 (aktualnie produkowane są też jednostki CPU 215 i 216). Różnią się one przede wszystkim liczbą obsługiwanych wejść-wyjść.

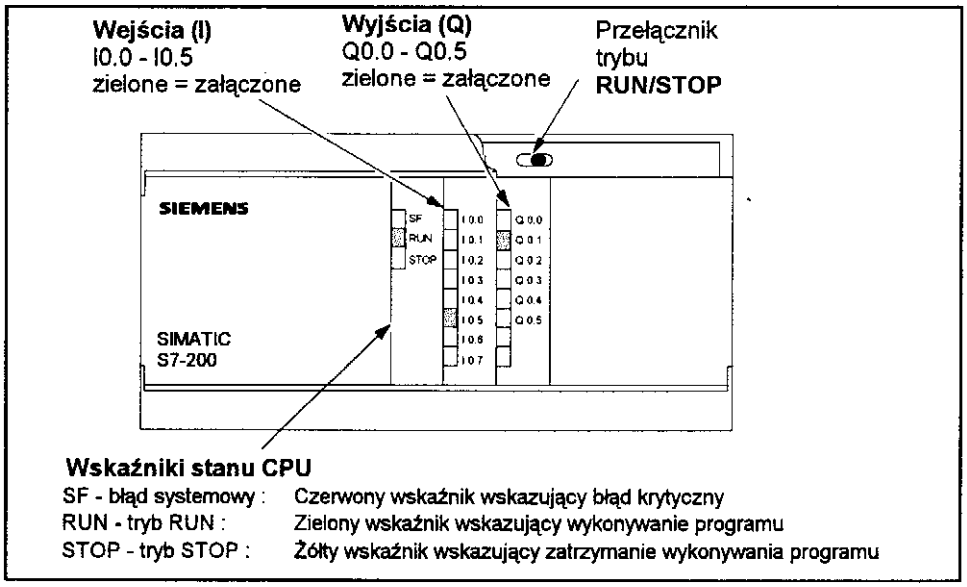
1. 1. Podstawowe dane techniczne sterowników

Sterownik Simatic S7	CPU 212	CPU 214
liczba wejść/wyjść dyskretnych	8/6	14/10
wyjście impulsowe	brak	2
pamięć programu (EEPROM)	1kB	4kB
pamięć danych (RAM)	512 słów	2000 słów
podtrzymanie pamięci danych (kondensatory)	50 godzin	190 godzin
czas wykonania 1000 instrukcji binarnych	1,3ms	0,8ms
timery	max. 64	max. 128
liczniki	max. 64	max. 128
szybki licznik impulsów (zdarzeń)	1 x 2kHz	1 x 2kHz, 2 x 7kHz
interfejs komunikacyjny	RS - 485	RS - 485
protokół komunikacyjny	PPI/Freeport	PPI/Freeport
potencjometry analogowe	1	2
zegar czasu rzeczywistego	brak	tak
hasło dostępu	3 poziomy	3 poziomy
wymienna kaseta z pamięcią EEPROM	brak	opcja
wymienna kaseta z baterią zasilającą	brak	opcja
max. liczba modułów rozszeszeń	2	7
zasilanie sterownika	(85 - 264)VAC	(85 - 264)VAC
obwody wejściowe z optoizolacją:		
zasilanie	(15 - 30)VDC	(15 - 30)VDC
obwody wyjściowe (przekaźnikowe):		
zasilanie	(5-30)VDC/250VAC	(5-30)VDC/250VAC
max. obciążenie	2A	2A

Programy dla sterowników S7, przy wykorzystaniu programu narzędziowego STEP 7 Micro/Win lub STEP 7 Micro/DOS, mogą być przedstawione w formie graficznej jako tzw. schemat drabinkowy (LAD) lub jako uporządkowana lista (ciąg) instrukcji (STL). Urządzeniem umożliwiającym tworzenie programu sterującego może użyć: komputer klasy IBM PC, komputer przenośny (laptop) lub programator kieszonkowy (ręczny).

1. 2. Opis wskaźników stanu

Wskaźniki stanu, umieszczone na płycie czołowej sterownika, wskazują aktualny tryb pracy jednostki CPU oraz aktualny stan wejść/wyjść:



Rys.1. Wskaźniki stanu.

Sterownik może znajdować się w dwóch trybach pracy: START lub STOP, które mogą być wybierane przy użyciu trójpołożeniowego przełącznika STOP/TERM/START umieszczonego pod górną przykrywką sterownika. Wyboru trybu pracy można też dokonać przy użyciu programatora, gdy przełącznik znajduje się w położeniu TERM, gdyż tylko wtedy występuje proces komunikacji między PLC, a programatorem. Ustawienie przełącznika w pozycję RUN lub STOP powoduje to, iż po wyłączeniu i ponownym załączeniu zasilania sterownik nie zmienia trybu pracy. Ponadto tryb pracy STOP jest automatycznie wybierany po ponownym zasileniu sterownika, gdy przełącznik był w pozycji TERM.

W trybie pracy STOP można:

- ładować program sterujący do pamięci sterownika,
- przeglądać i zmieniać zawartość rejestrów wewnętrznych sterownika,
-zmieniać parametry konfiguracyjne sterownika.

W trybie pracy RUN, gdy wykonywany jest program sterujący, nie można dokonać próby ładowania programu do sterownika.

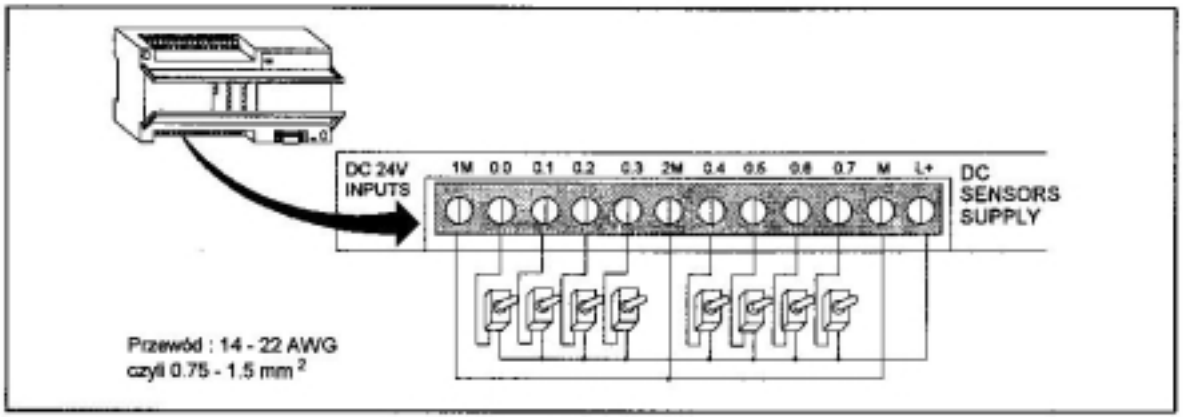
1. 3. Numeracja wejść/wyjść sterownika

Wybierając w programie sterującym określone wejście (I) lub wyjście (Q) sterownika należy podać numer identyfikacyjny (I/O) określający jednocześnie adres w pamięci wewnętrznej CPU (patrz p. 2. 2.). W poniższej tabeli przedstawiono sposób adresowania wejść i wyjść dla sterowników S7-200 z CPU 212 i 214:

numer wejścia/wyjścia	Rodzaj jednostki centralnej sterownika S7			
	CPU - 212		CPU - 214	
	wejście	wyjście	wejście	wyjście
pierwsze	I0.0	Q0.0	I0.0	Q0.0
drugie	I0.1	Q0.1	I0.1	Q0.1
trzecie	I0.2	Q0.2	I0.2	Q0.2
czwarte	I0.3	Q0.3	I0.3	Q0.3
piąte	I0.4	Q0.4	I0.4	Q0.4
szóste	I0.5	Q0.5	I0.5	Q0.5
siódme	I0.6		I0.6	Q0.6
ósme	I0.7		I0.7	Q0.7
dziewiąte			I1.0	Q1.0
dziesiąte			I1.1	Q1.1
Jedenaste			I1.2	
Dwunaste			I1.3	
Trzynaste			I1.4	
Czternaste			I1.5	
Adresy wyjść do wykorzystania jako zmienne wewnętrzne (bity)				
		Q0.6		Q1.2
		Q0.7		Q1.3
				Q1.4
				Q1.5
				Q1.6
				Q1.7

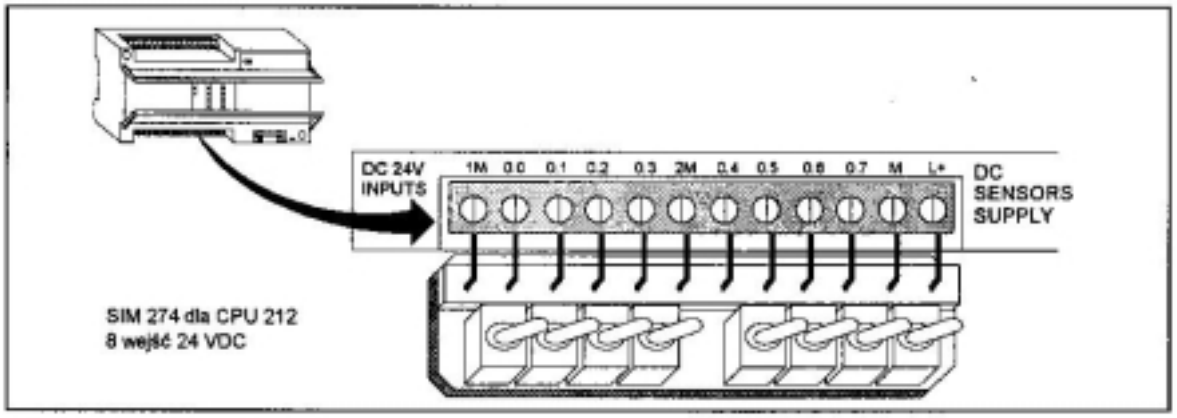
1. 4. Sposób podłączenia obwodów wejściowych i wyjściowych sterownika

Do przeprowadzenia testów działania danego programu wystarczy podłączyć przewodami do wejść sterownika przełączniki w liczbie odpowiadającej ilości wejść:



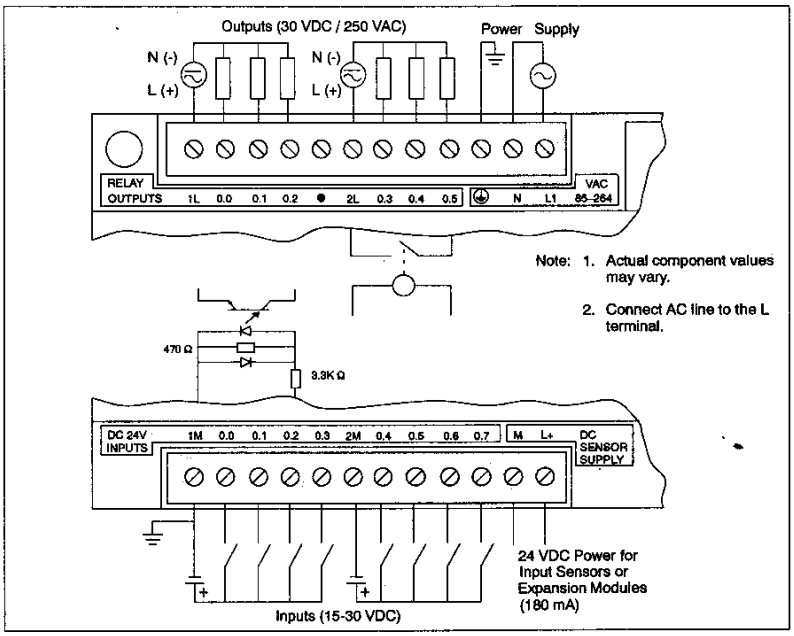
Rys.2. Sposób podłączenia 8 przełączników wykorzystujących wewnętrzny zasilacz sterownika (S7-200 z CPU 212).

Można też użyć fabrycznego symulatora wejść (SIM 274 dla CPU 212) przedstawionego na poniższym rysunku:

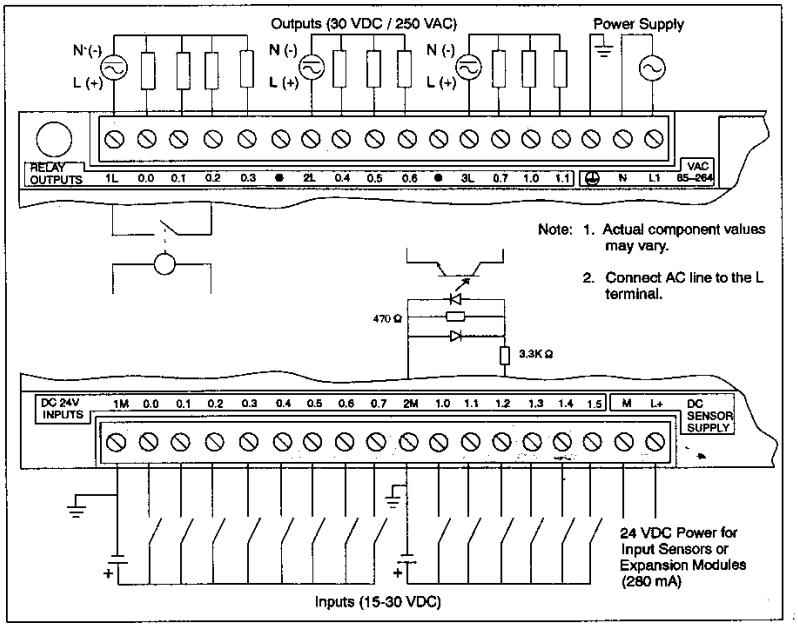


Rys.3. Sposób podłączenia symulatora wejść SIM 274.

Sposób podłączenia obwodów zewnętrznych pod listwy zaciskowe sterownika S7-200 z CPU 212 przedstawia rysunek 4, a S7-200 z CPU 214 rysunek 5.



Rys.4. Połączenia zewnętrzne dla CPU 212 (wyj. przekaźnikowe).



Rys.5. Połączenia zewnętrzne dla CPU 214 (wyj. przekaźnikowe).

2. Wprowadzenie do programowania sterowników S7 - 200

2. 1. Oznaczenia literowe identyfikatorów

W poniższej tabeli wyspecyfikowano wszystkie identyfikatory, którymi możemy się posługiwać przy tworzeniu programu sterującego:

Identyfikator	
oznaczenie	nazwa
I	zmienna wejściowa
Q	zmienna wyjściowa
M	wewnętrzna zmienna dyskretna
SM	wewnętrzna zmienna specjalna (zmienna systemowa)
V	zmienne pamięciowe
T	timer
C	licznik
AI	zmienna wejściowa analogowa
AQ	zmienna wyjściowa analogowa
AC	akumulator
HC	szybki licznik
K	stała

2. 2. Typy zmiennych

Przy opracowywaniu programów sterujących wykorzystuje się zmienne dyskretne (bitowe) oraz rejestrowe (wielobitowe). Ma to również odzwierciedlenie w sposobie oznaczania identyfikatorów. W poniższej tabeli przedstawiono zmienne rejestrowe oraz ich reprezentację w systemie dziesiętnym i heksadecymalnym (szesnastkowym):

zmienna rejestrowa	liczba całkowita bez znaku		liczba całkowita ze znakiem	
	zapis dziesiętny	zapis szesnastkowy	zapis dziesiętny	zapis szesnastkowy
B (bajt - dana zawierająca 8 bitów)	0 do 255	0 do FF	-128 do +127	80 do 7F
W (słowo - dana awierająca 16 kolejnych bitów)	0 do 65535	0 do FFFF	-32.768 do +32.767	8000 do 7FFF
D (podwójne słowo - dana zawierająca 32 kolejne bity)	0 do 4.294.967.295	0 do FFFF FFFF	-2.147.483.648 do +2.147.483.647	8000 0000 do 7FFF FFFF

2. 3. Przykłady oznaczania identyfikatorów zmiennych

Nazwa identyfikatora	Typ zmiennej			
	1 bitowa (bit)	8 bitowa (B - Byte)	16 bitowa (W - Word)	32 bitowa (D - Double Word)
zmienna pamięciowa (**V** - Variable Memory)	-	VB	VW	VD
zmienna wejściowa (**I** - Input)	I	IB	IW	ID
zmienna wyjściowa (**Q** - Output)	Q	QB	QW	QD
wewnętrzna zmienna (**M** - Internal Memory)	M	MB	MW	MD
wewnętrzna zmienna specjalna - systemowa (**SM** - Specjal Memory)	SM	SMB	SMW	SMD
zmienna wejściowa analogowa (**AI** - Analog Input)	-	-	AIW	-
zmienna wyjściowa analogowa (**AQ** - Analog Output)	-	-	AIQ	-

2. 4. Przestrzeń adresowa jednostek centralnych

Przy pisaniu programów obok oznaczenia literowego identyfikatora należy podać odpowiednią cyfrę (liczbę) określającą miejsce (adres) pamięci CPU, w którym przypisana mu zmienna będzie umieszczona. Przestrzeń adresowa jednostek centralnych CPU - 212 i CPU - 214, przedstawiona poniżej, pozwala w sposób prawidłowy określić właściwy adres identyfikatora:

	CPU - 212	**CPU - 214**
	MSB* 7 LSB* 0	MSB* 7 LSB* 0
Zmienne pamięciowe - odczyt/zapis (V - memory)	V0	V0
	Pamięć nieulotna od adresu V0 do V199	Pamięć nieulotna od adresu V0 do V1023
	V199	V1023
Zmienne pamięciowe - odczyt/zapis (V - memory)	V200	V1024
	. . .	. . .
	V1023	V4095
Rejestry wejściowe	I0.7 . . . I0.0	I0.7 . . . I0.0
	. .	. .

- odczyt/zapis	.	.
	I7.7 . . . I7.0	I7.7 . . . I7.0
Rejestry wyjściowe - odczyt/zapis	Q0.7 . . . Q0.0 . . . Q7.7 . . . 7.0	Q0.7 . . . Q0.0 . . . Q7.7 . . . Q7.0
Wewnętrzne zmienne -odczyt/zapis	M0.7 . . . M0.0 . . . M15.7 . . . M15.0	M0.7 . . . M0.0 . . . M31.7 . . . M31.0
Wewnętrzne zmienne specjalne (systemowe) - tylko do odczytu	SM0.7 . . . SM0.0 . . . SM29.7 . . . M29.0	SM0.7 . . . SM0.0 . . . SM29.7 . . . SM29.0
Wewnętrzne zmienne specjalne (systemowe) - odczyt/zapis	SM30.7 . . . M30.0 . . . SM45.7 . . . SM45.0	SM30.7 . . . M30.0 . . . SM85.7 . . . M85.0

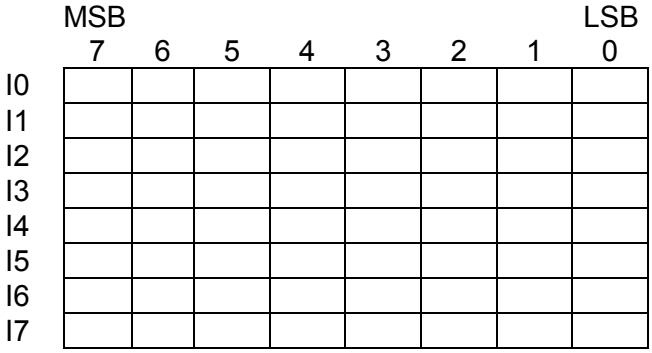
–

	CPU - 212		CPU - 214	
	MSB* 15 … LSB* 0	bit wyj.	MSB* 15 … LSB* 0	bit wyj.
Timery - odczyt/zapis	T0	T0	T0	T0
	. . .	. . .	. . .	. . .
	T63	T63	T127	T127
		bit wyj.		bit wyj.
Liczniki - odczyt/zapis	C0	C0	C0	C0
	. . .	. . .	. . .	. . .
	C63	C63	C127	C127
Wejścia analogowe - tylko odczyt	AIW0		AIW0	
	AIW2		AIW2	
	. . .		. . .	
	AIW30		AIW30	
Wyjścia analogowe -tylko zapis	AQW0		AQW0	
	AQW2		AQW2	
	. . .		. . .	
	AQW30		AQW30	
	MSB* 31 … LSB* 0			
Rejestry akumulatorów - odczyt/zapis	AC0			
	AC1			
	AC2			
	AC3			
Szybkie liczniki	HC0 (2kHz)			
	HC1 (tylko dla CPU 214 - 7kHz)			
	HC2 (tylko dla CPU 214 - 7kHz)			

/* MSB (most significant bit) - bit najbardziej znaczący,
LSB (least significant bit) - bit najmniej znaczący

2. 5. Przykłady adresowania identyfikatorów zmiennych

W przypadku określania miejsca w przestrzeni adresowej dla zmiennej bitowej (dyskretnej) podaje się najpierw numer (adres) bajtu a następnie po kropce numer wybranego bitu, np:

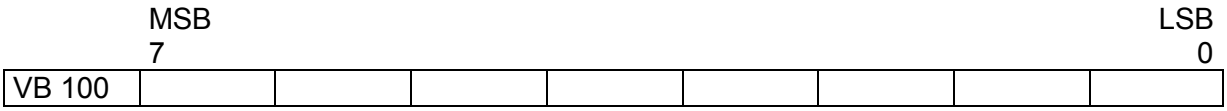


I 3 . 4 - zmienna wejściowa bitowa o numerze 4 wybrana z bajtu o adresie 3.

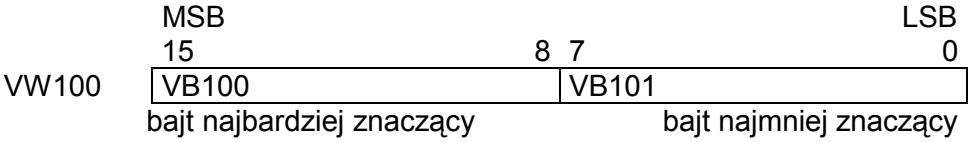
MSB - bit najbardziej znaczący
LSB - bit najmniej znaczący

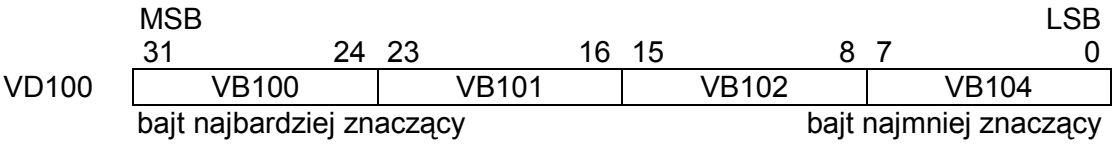
Rys.6. Przestrzeń adresowa dla zmiennych wejściowych

W przypadku określania miejsca w przestrzeni adresowej dla zmiennych rejestrowych (bajt - B, słowo - W, podwójne słowo - D) obok identyfikatora podaje się adres najbardziej znaczącego bajtu (danej 8 - bitowej). Mniej znaczące bajty zajmują kolejne bajty przestrzeni adresowej. Poniższe przykłady ilustrują rozmieszczenie zmiennych w przestrzeni adresowej CPU w przypadku tego samego adresu:



zmienna 8 bitowa (bajt)
VB 100 - zmienna 8 bitowa [VB] o adresie 100





VW 100 - zmienna 16 bitowa [VW] o adresie 100

VD 100 - zmienna 32 bitowa [VD] o adresie 100

MSB bit najbardziej znaczący
LSB bit najmniej znaczący

2. 6. Zmienne z pamięcią stanu

Przestrzeń adresowa sterownika podzielona jest na 6 obszarów, które domyślnie ustawiane są jako obszary z pamięcią stanu (retentive) czyli takie, w których zmienne zachowują swój stan nawet po wyłączeniu zasilania:

Domyślnie ustawionie zakresy zmiennych z pamięcią stanu (retentive)

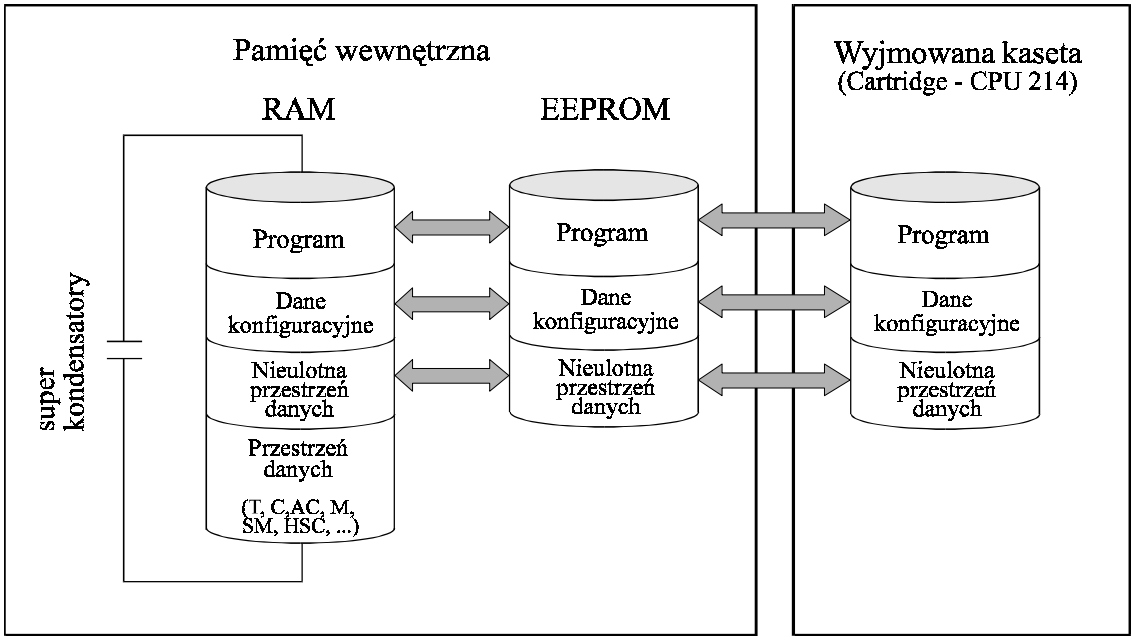
Obszar z pamięcią stanu	Jednostka CPU-212	Jednostka CPU-214
Obszar o numerze 0	V0 do V1023	V0 do V4095
Obszar o numerze 1	nie używany	nie używamy
Obszar o numerze 2	To do T31	T0 do T31
Obszar o numerze 3	nie używany	T64 do T95
Obszar o numerze 4	C0 do C63	C0 do C127
Obszar o numerze 5	M0 do M15	M0 do M31

Od momentu wyłączenia napięcia zasilającego pamięć danych umieszczona w pamięci RAM jest podtrzymywana przez super kondensatory lub przez bateryjki umieszczone w wyjmowanym module (tylko dla CPU-214 opcjonalnie).

Po pewnym czasie tj. około 50 godzin dla CPU-212 i 190 godzin dla CPU-214 od momentu wyłączenia zawartość pamięci RAM zostaje stracona. O fakcie tym informuje bit specjalny SM0.2 (patrz dodatek).

Korzystając z oprogramowania narzędziowego można na etapie konfigurowania sterownika zdefiniować zmienne z pamięcią stanu (patrz. Micro/WIN - Programming Software).

Oprócz pamięci RAM sterownik posiada też pamięć nieulotną (non-volatile) EEPROM. Przechowywać w niej można zmienne o adresie: od V0 do V199 dla CPU-212 oraz od V0 do V1023 dla CPU-214 (patrz przestrzeń adresowa). Aby móc zapisywać dane w pamięci EEPROM należy oprócz samej struktury logicznej programu wpisać do sterownika tzw. bazę danych (DB1), która została omówiona w instrukcji - Micro/WIN - Oprogramowanie narzędziowe). Służy ona do ustawiania zmiennych (V-memory). Pierwszych 200 bajtów dla CPU-212 oraz 1023 dla CPU-214 jest wtedy automatycznie wpisywane do pamięci EEPROM.



Rys.7. Struktura pamięci wewnętrznej sterownika

Po doprowadzeniu napięcia zasilającego do sterownika zmienne bez pamięci stanu (non-retentive) umieszczone w pamięci EEPROM są z niej kopiowane do pamięci RAM. Jeżeli w pamięci RAM umieszczone były zmienne z pamięcią (retentive) oraz w chwili doprowadzenia napięcia pamięć RAM była podtrzymana przez super kondensatory, zawartość tej pamięci nie ulega zmianie, natomiast jeżeli pamięć RAM utraciła te dane, to zmienne z pamięcią (retentive) są kopiowane do niej z pamięci EEPROM. Niezależnie od tego czy blok danych (DB1) został załadowany do sterownika, czy też nie można zapisywać zmienne do pamięci EEPROM korzystając ze zmiennych systemowych: SMB31 oraz SMW32 - patrz załącznik.

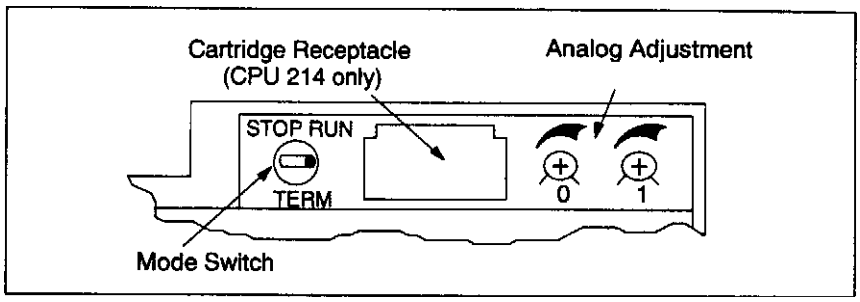
2. 7. Zmienne systemowe - SM (Special Memory bits)

Jest to grupa zmiennych, które umożliwiają dostęp do danych systemowych takich jak: informacje o statusie, błędach działania oraz uszkodzeniach wewnętrznych sterownika (awaria CPU, uszkodzenie pamięci RAM, ROM, słabe baterie, błędne hasło dostępu itp.), błędach działania układów wejścia/wyjścia sterownika, błędach programowych, błędach związanych z transferem danych między sterownikiem a programatorem, błędach związanych z konfiguracją systemu, informacje dotyczące aktualnego czasu i daty, itp. Błędy i usterki, o których mowa, mogą zostać tylko zarejestrowane poprzez ustawienie określonych zmiennych systemowych (błędy diagnostyczne) lub spowodować zatrzymanie sterownika (błędy krytyczne). Wystąpieniu błędu krytycznego towarzyszy zapalenie się następujących wskaźników stanu: RUN, STOP i wskaźnika błędu krytycznego SF oraz

wyzerowanie wszystkich wyjść sterownika. W takiej sytuacji należy zrestartować sterownik PLC w celu uruchomienia funkcji diagnostycznych. Jeśli te wykryją w dalszym ciągu istnienie błędu krytycznego, ponownie zapali się wskaźnik stanu SF. W innym przypadku sterownik PLC ustawi się w tryb pracy normalnej.
Bity specjalne (special memory bits) mogą być wykorzystane jako zmienne: bitowe, bajtowe, jako słowa lub podwójne słowa. Bajty specjalne od SM0 do SM29 umożliwiają tylko odczyt informacji. Ich zawartość jest uaktualniana przez sterownik programowalny i dostarczają one informacji o statusie sterownika. Natomiast bajty o adresach SM30 do SM85 są typu zapis/odczyt umożliwiają wybór i sterowanie różnorodnymi funkcjami
Szczegółowe informacje dotyczące przeznaczenia zmiennych systemowych zawarte są w dokumentacji - "SIMATIC S7 Reference Manual".
Opis niektórych z nich zamieszczony został w załączniku na końcu niniejszego opracowania.

2. 8. Potencjometry analogowe

Szczególnym typem zmiennych systemowych są rejestry, których aktualna wartość zmieniana jest przy pomocy potencjometrów analogowych umieszczonych w sterowniku pod górną przykrywką uchylną:



Rys.8. Umiejscowienie potencjometrów analogowych.

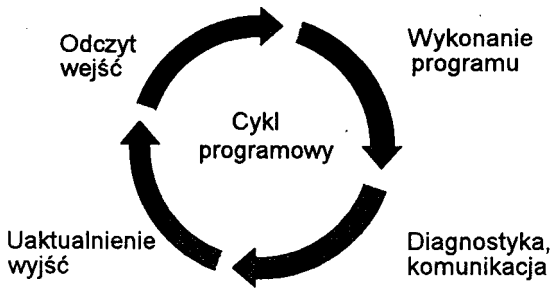
Potencjometrom tym przypisane są rejestry specjalne 8 bitowe: SMB28 dla sterownika z jednostką centralną CPU 212 oraz SMB28 i SMB29 dla CPU 214. Wartość tych rejestrów, w pełnym zakresie zmian położenia potencjometru, zmienia się w zakresie od 0 do 255. Pozwalają one na ręczne strojenie zmiennych analogowych. Mogą być użyte w programie do zmiany nastaw, zmiany granic itp.

2. 9. Budowa programu sterującego w sterownikach SIMATIC S7

Oprogramowanie narzędziowe MICRO/WIN pozwala programować na dwa wzajemnie uzupełniające się sposoby, tj przy pomocy:

- schematu drabinkowego (LAD)
- listy instrukcji (STL)

Obie wersje programu sterującego są równoważne i dają te same możliwości funkcjonalne. Różnią się tylko sposobem zapisu, a wybór jednego z nich zależy od doświadczenia i przyzwyczajenia programisty.
Decydując się na jeden ze sposobów pisania programu sterującego, drugi wpisywany jest automatycznie jako program alternatywny. Przejście na drugi sposób zapisu jest możliwe w każdej chwili.
Każdy program sterujący, bez względu na jego postać, wykonywany jest cyklicznie. Po wykonaniu ostatniej instrukcji (STL) lub szczebla drabiny (LAD) program sterujący jest analizowany od początku, co ilustruje schemat przedstawiony poniżej:



Rys.9. Cykl programowy (ang. = *scan cycle*)

Każdy cykl programowy rozpoczyna się obsługą wejść polegającą na odczytaniu aktualnych stanów na wejściach sterownika i wpisaniu ich do rejestrów wejściowych. Na tej podstawie rozpoczyna się proces wykonania części logicznej programu sterującego (analiza programu). Po wykonaniu programu PLC realizuje proces komunikacji poprzez port komunikacyjny z programatorem lub mudułami zewnętrznymi oraz przeprowadza samodiagnostykę.

Ostatnią fazą cyklu programowego jest obsługa wyjść polegająca na uaktualnieniu stanu wyjść, gdyż efektem wykonania programu może być zmiana wartości rejestrów wyjściowych, którym przypisuje się fizyczne wyjścia sterownika. Czas wykonania cyklu zależy od rozmiaru programu użytkowego, ilości użytych wejść i wyjść sterownika i ilości danych wymienianych podczas komunikacji (np. komunikowanie się sterownika z oprogramowaniem wizualizacyjnym typu SCADA).
W rejestrach specjalnych o adresach: SMW22, SMW24 i SMW26 można odczytać w milisekundach odpowiednio: czas ostatniego cyklu pracy sterownika (ostatniego skanowania), czas minimalny i maksymalny cyklu pracy liczony od momentu uruchomienia sterownika).

2. 10. Sposoby programowania sterowników S7 - 200

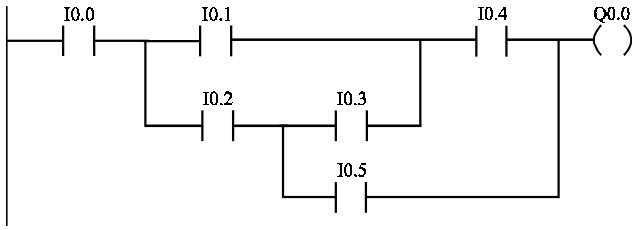
2. 10. 1. Schemat drabinkowy (LAD)

Część logiczna programu sterującego składa się z umieszczonych jeden pod drugim tzw. szczebli programowych. Przypominają one typowy elektryczny schemat połączeń. W skład szczebla wchodzą: elementy logiczne (styki), przekaźniki, jak i bardziej złożone bloki funkcyjne. Schemat drabinkowy posiada symboliczne źródło zasilania. Zakłada się przepływ sygnału od szyny umieszczonej po lewej stronie schematu do przekaźników lub bloków funkcyjnych umieszczonych po prawej stronie danego szczebla. Kolejne szczeble drabiny odczytywane są kolejno od góry do dołu. Po dojściu do ostatniego szczebla proces śledzenia programu rozpoczyna się od początku.

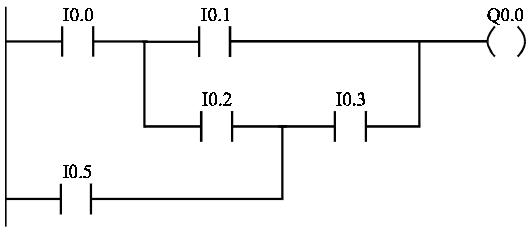
Struktura szczebla drabiny logicznej

Szczebel drabiny logicznej (oznaczany w programie jako NETWORK) musi posiadać odpowiedni format i składnię. Oto kilka najważniejszych zasad:

- każdy szczebel może zawierać do 16 linii równoległych, każda linia może zawierać do 16 elementów logicznych połączonych szeregowo,
- ostatnim elementem szeregowego połączenia w danym szczeblu musi być jeden z przekaźników lub blok funkcyjny,
- szczebel może zawierać maksymalnie do 16 przekaźników,
- szczebel musi zawierać przynajmniej jeden styk przed wystąpieniem przekaźnika, bloku funkcyjnego lub połączenia pionowego,
- nie może wystąpić rozgałęzienie mające początek lub koniec wewnątrz innego odgałęzienia:



W powyższym przykładzie rozgałęzienie (linia zawierająca styk I0.5) bierze początek w niewłaściwym miejscu szczebla (wewnątrz innego odgałęzienia).



W powyższym przykładzie styk I0.5 jest nieprawidłowo połączony z wnętrzem odgałęzienia zawierającego styki: I0.2 oraz I0.3.

2. 10. 2. Lista instrukcji (STL)

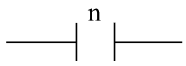
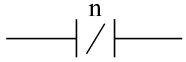
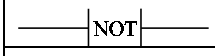
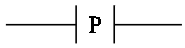
Jest to zapis bardziej symboliczny stanowiący ciąg kolejno ułożonych jedna pod drugą instrukcji (rozkazów). Wykonywane są one cyklicznie od góry do dołu. Jest to zapis bardziej zwarty operujący na skrótach literowych symbolizujących np: wejścia/wyjścia sterownika, operacje logiczne i matematyczne oraz inne bloki funkcyjne.

3. Opis elementów logicznych oprogramowania MicroWin

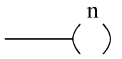
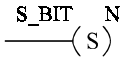
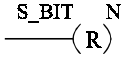
W rozdziale tym zostaną opisane podstawowe elementy logiczne programu sterującego, sposób ich działania oraz typy zmiennych, które mogą być przypisane wejściom oraz wyjściom elementów logicznych. W oprogramowaniu narzędziowym MICRO/WIN elementy te zostały pogrupowane w następujące kategorie:

Nazwa kategorii	Zawartość
Elementy stykowe *(Contacts)*	styki normalnie otwarte/zamknięte, komparatory, styk negacji, styki impulsowe.
Przekaźniki wyjściowe *(Output Coils)*	przekaźnik o stykach otwartych, przekaźniki ustawialne SET/RESET.
Timery i liczniki *(Timers/Counters)*	timery z i bez pamięci, liczniki zliczające w górę oraz w górę i w dół.
Operacje matematyczne *(Math/Inc/Dec)*	funkcje dodawania, odejmowania, mnożenia i dzielenia liczb, funkcja pierwiastka kwadratowego, zwiększanie/ zmniejszanie wartości o 1.
Kopiowanie, przesuwanie, obrót *(Move/Shift/Rotate/Fill)*	kopiowanie zmiennych, przesuwanie zmiennych w prawo/lewo, obrót (rotacja) zmiennych w prawo/lewo.
Funkcje sterujące *(Program Control)*	funkcje końca lub zatrzymania programu, funkcje obsługi podprogramów funkcje skoków programowych
Operacje logiczne *(Logical Operations)*	iloczyn, suma logiczna słów, alternatywa wyłączająca słów (XOR - albo), inwersja słów.
Konwersja *(Conversion)*	zamiana danych BCD-4/liczbę całkowitą i odwrotnie, zamiana kodu ASCII na Hex i odwrotnie, moduł wyświetlacza 7 - segmentowego, inne.
Szybkie operacje *(High Speed Operations)*	definiowanie parametrów szybkich liczników, wyjście impulsowe (tylko dla CPU 214).
Zegar czasu rzeczywistego *(Real Time Clock)*	odczyt aktualnej daty i czasu (rejestr 8-bajtowy), ustawianie powyższych parametrów.
Linie *(Lines)*	linia pozioma, linia pionowa.
Operacje tablicowe *(Table/Find)*	Wpisywanie do tablicy danych, wyprowadzanie danych z tablicy, wyszukiwanie w tablicy określonych danych
Przerwania i komunikacja *(Interrupt/Comunications)*	Bloki funkcyjne i przekaźniki obsługujące procedury przerwań programowych, bloki funkcyjne obsługujące pracę sieciową sterowników
Wszystkie kategorie *(All Categories)*	wszystkie elementy i bloki funkcjonalne poszczególnych kategorii zgrupowane w jedną w porządku alfabetycznym.

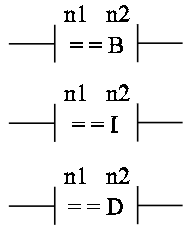
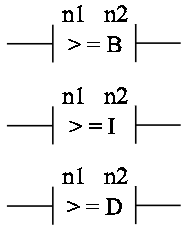
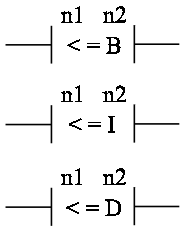
3. 1. Styki

LAD	Opis	Zmienna
	Styk normalnie otwarty. Przewodzi sygnał (zwiera styki), gdy wartość logiczna przypisanej zmiennej wynosi "1". **(Normally Open)**	n: I, Q, M, SM, T, C, V (bit)
	Styk normalnie zamknięty. Przewodzi sygnał (zwiera styki), gdy wartość logiczna przypisanej zmiennej wynosi "0". **(Normally Closed)**	
	Styk negacji. Gdy dochodzi do niego sygnał jego styk jest otwarty. **(NOT)**	bez dodatkowych oznaczeń
	Styk zwierny na czas jednego cyklu pracy sterownika, gdy sygnał podany do tego styku zmienia wartość z "0" na "1". (styk impulsowy) **(Positiv Transition)**	
	Styk zwierny na czas jednego cyklu pracy sterownika, gdy sygnał podany do tego styku zmienia wartość z "1" na "0". (styk impulsowy) **(Negative Transition)**	

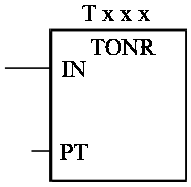
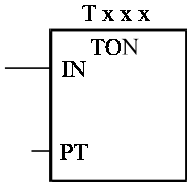
3. 2. Przekaźniki

LAD	Opis	Zmienna
	Przekaźnik ustawia wartość przypisanej zmiennej na "1", gdy podany zostanie do niego sygnał. Jest to przekaźnik o stykach otwartych. **(Output)**	n: I, Q, M, SM, T, C, V (bit)
	Przekaźnik ustawialny "SET". Przekaźnik ustawia na "1" kilka przypisanych mu zmiennych dyskretnych, których liczbę określa parametr "N". Zmienna S_BIT określa adres początkowy tych zmiennych. Jest to przekaźnik z pamięcią, gdyż jego stan jest podtrzymywany w przypadku wyłączenia zasilania sterownika lub po przejściu sterownika w tryb STOP. Wartość "1" jest utrzymywana do momentu, aż sygnał podany zostanie do przekaźnika "RESET". **(Set)**	S_BIT:I, Q, M, SM, T, (bit) C, V N:IB, QB, MB, SMB, (bajt) VB, AC, K
	Przekaźnik ustawialny "RESET" przystosowany do współpracy z przekaźnikiem "SET". Gdy do przekaźnika podany zostanie sygnał wartość przypisanej mu zmiennej S_BIT oraz kilka kolejnych zmiennych dyskretnych określonych parametrem "N" ustawiona zostanie na "0". Jest to przekaźnik z pamięcią. **(Reset)**	

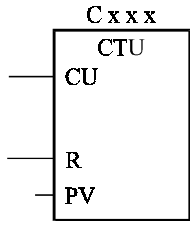
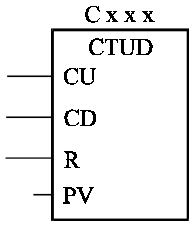
3. 3. Styki komparatorów

LAD	Opis	Zmienna
	Zestyk jest zwarty (przewodzi sygnał), gdy: n1 = n2 B = bajt (zmienna 8 bitowa) I = liczba całkowita ze znakiem (zmienna 16 bitowa) D = liczba całkowita ze znakiem (zmienna 32 bitowa) R = wartość rzeczywista **(= = Byte)**, **(= = Word Integer)**, **(= = Double Word Integer)**	
	Zestyk jest zwarty (przewodzi sygnał), gdy: n1 > = n2 B = bajt (zmienna 8 bitowa) I = liczba całkowita ze znakiem (zmienna 16 bitowa) D = liczba całkowita ze znakiem (zmienna 32 bitowa) R = wartość rzeczywista **(> = Byte)**, **(> = Word Integer)**, **(> = Double Word Integer)**	n1, n2: VB,IB, QB, MB, (bajt) SMB, AC, K n1, n2: VW, T, C, IW, QW, (słowo) MW, AC, SMW, AIW, K n1, n2: VD, ID, QD, K, (32bity) MD, SMD, AC, HC
	Zestyk jest zwarty (przewodzi sygnał), gdy: n1 < = n2 B = bajt (zmienna 8 bitowa) I = liczba całkowita ze znakiem (zmienna 16 bitowa) D = liczba całkowita ze znakiem (zmienna 32 bitowa) R = wartość rzeczywista **(< = Byte)**, **(< = Word Integer)**, **(< = Double Word Integer)**	

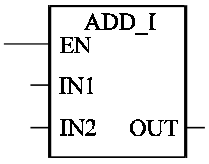
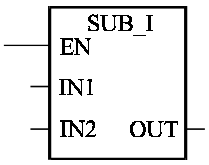
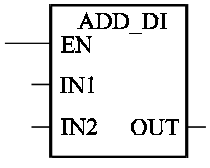
3. 4. Timery

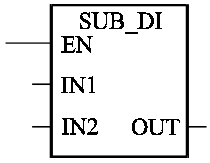
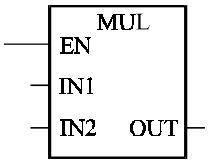
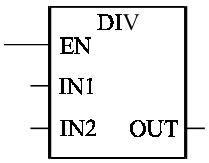
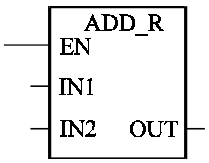
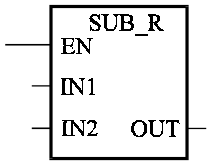
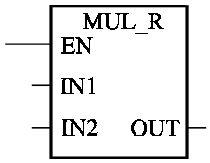
LAD	Opis	Zmienna
	Timer zlicza czas, gdy podany zostanie do niego sygnał "IN" i zatrzymuje naliczoną wartość, gdy sygnał jest nieaktywny. Po ponownym pojawieniu się tego sygnału zliczanie czasu jest kontynuowane. Może zliczyć maksymalnie +32767 jednostek czasu. Gdy wartość bieżąca zrówna się z wartością zadaną "PT" timer zostaje uaktywniony, tzn. bit wyjściowy timera "Txxx" ustawia się na "1". **(Timer - Retentive On Delay)** CPU 212/214 CPU 214 1ms T0 T0 - T64 10ms T1 - T4 T65 - T68 100ms T5 - T31 T69 - T95	Txxx: CPU 212: 0 - 31 (słowo)CPU 214: 0 -1, 64 - 95 PT: VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K
	Timer zlicza czas, gdy podany zostanie do niego sygnał "IN" i zostaje wyzerowany, gdy sygnał przestaje być aktywny. Po pojawieniu się tego sygnału zliczanie czasu rozpoczyna się od początku. Zakres zliczanych jednostek czasu: +32767. Gdy wartość bieżąca zrówna się z wartością zadaną "PT" timer zostaje uaktywniony, tzn. bit wyjściowy timera "Txxx" ustawia się na "1". **(Timer - On Delay)** CPU 212/214 CPU 214 1ms T32 T96 10ms T33 - T36 T97 - T100 100ms T37 - T63 T101 - T127	Txxx: CPU 212: 32 - 63 (słowo)CPU 214: 32 - 63, 96 - 127 PT:VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K

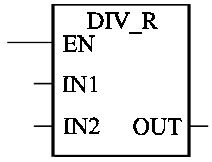
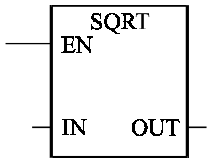
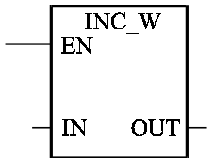
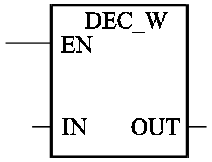
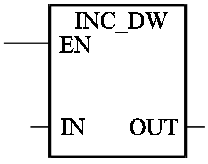
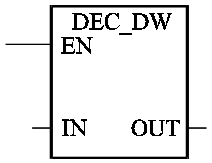
3. 5. Liczniki

LAD	Opis	Zmienna
	Licznik zliczający w górę służy do zliczania impulsów (zbocze narastające) doprowadzonych do wejścia "CU". Zakres licznika wynosi 32767imp. Licznik jest wyzerowany tak długo, jak długo do wejścia kasującego "R" podany zostaje sygnał. Przy zrównaniu się liczby zliczanych impulsów z wartością zadaną "PV" licznik zostaje uaktywniony, tzn. bit wyjściowy licznika "Cxxx" ustawia się na "1". **(Count Up)**	Cxxx:CPU 212: 0 - 63 (słowo)CPU 214:0-127 PV:VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K
	Licznik umożliwia zliczanie w górę impulsów (zbocze narastające) doprowadzonych do wejścia "CU" lub w dół - wejście "CD". Zakres zliczanych impulsów wynosi: -32768 do +32767. Licznik jest wyzerowany tak długo, jak długo do wejścia kasującego "R" podany zostaje sygnał. Przy zrównaniu się liczby zliczanych impulsów z wartością zadaną "PV" licznik zostaje uaktywniony, tzn. bit wyjściowy licznika "Cxxx" ustawia się na "1". **(Count Up/Down)**	Cxxx: CPU 212: 0 - 63 (słowo)CPU 214: 0 -27, PV: VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K

3. 6. Operacje matematyczne

LAD	Opis	Zmienna
	Blok funkcyjny umożliwiający dodawanie dwóch liczb całkowitych 16 bitowych ze znakiem (Signet Integer) w zakresie: -32768 do +32767 doprowadzonych do wejść: "IN1" oraz "IN2". Wynik działania , jako zmienna 16 bitowa, wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał./1,2,3 **(Add Integer)**: IN1 + IN2 = OUT	I N1, IN2: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K
	Blok funkcyjny umożliwiający odejmowanie dwóch liczb całkowitych 16 bitowych ze znakiem (Signet Integer) w zakresie: -32768 do +32767 doprowadzonych do wejść: "IN1" oraz "IN2". Wynik działania , jako zmienna 16 bitowa, wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał./1,2,3 **(Subtract Integer)**: IN1 - IN2 = OUT	OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC
	Blok funkcyjny umożliwiający dodawanie dwóch liczb całkowitych 32 bitowych ze znakiem (Signet Integer) w zakresie: -2147483648 do +2147483647 doprowadzonych do wejść: "IN1"oraz "IN2". Wynik działania , jako zmienna 32 bitowa, wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał./1,2,3 **(Add Double Integer)**: IN1 + IN2 = OUT	IN1, IN2: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC

	Blok funkcyjny umożliwiający odejmowanie dwóch liczb całkowitych 32 bitowych ze znakiem (Signet Integer) w zakresie: -2147483648 do +2147483647 doprowadzonych do wejść: "IN1"oraz "IN2". Wynik działania , jako zmienna 32 bitowa, wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał./1,2,3 **(Subtract Double Integer)**: IN1 - IN2 = OUT	IN1, IN2: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC
	Blok funkcyjny umożliwiający mnożenie dwóch liczb całkowitych 16 bitowych ze znakiem (Signet Integer) w zakresie: -32768 do +32767 doprowadzonych do wejść: "IN1"oraz "IN2". Wynik działania , jako zmienna 32 bitowa, wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał./2,3 **(Multiply Integer)**: IN1 * IN2 = OUT	IN1, IN2: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC
	Blok funkcyjny umożliwiający dzielenie dwóch liczb całkowitych 16 bitowych ze znakiem (Signet Integer) w zakresie: -32768 do +32767 doprowadzonych do wejść: "IN1"oraz "IN2". Wynik działania , jako zmienna 32 bitowa, wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał. /1,2,3 **(Divide Integer)**: IN1 / IN2 = OUT	Uwaga: *Część całkowita ilorazu przechowywana jest w pierwszych 16 mniej znaczących bitach, a reszta w kolejnych 16 bar- dziej znaczących bitach zmiennej wyjściowej 32 bitowej.*
	Operacja dodawania dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał. /1,2,3 **(Add Real)**: IN1 + IN2 = OUT	
	Operacja odejmowania dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał. /1,2,3 **(Subtract Real)**: IN1 - IN2 = OUT	N: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K
	Operacja mnożenia dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał. /1,2,3 **(Multiply Real)**: IN1 * IN2 = OUT	OUT: VD, ID, QD, SMD, (32 bity) AC

	Operacja dzielenia dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i wyprowadzony jest na wyjście "OUT".Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał. /1,2,3,4 **(Divide Real)**: IN1 / IN2 = OUT	N:VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, SMD, (32 bity) AC
	Blok funkcyjny umożliwiający wyliczenie pierwiastka kwadratowego z liczby rzeczywistej 32 bitowej, która jest jego parametrem wejściowym "IN". Wynik (część całkowita), jako zmienna 32 bitowa, wyprowadzony jest na wyjście "OUT". Operacja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał. /1, 2, 3 **(Square Root of Real Number)**	IN: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC
	Blok funkcyjny zwiększa wartość słowa wejściowego 16 bitowego ze znakiem (-32768 do +32767) doprowadzonego do wejścia "IN" o "1", gdy podany zostaje sygnał do wejścia "EN". Wynik wyprowadzo- ny jest jako zmienna 16 bitowa ze znakiem na wyjście "OUT"./1,2,3 **(Increment Word)**: IN + 1 = OUT	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC
	Blok funkcyjny zmniejsza wartość słowa wejściowego 16 bitowe- go ze znakiem (-32768 do +32767) doprowadzonego do wejścia "IN" o "1", gdy podany zostanie sygnał do wejścia "EN". Wynik wypro- wadzony jest jako zmienna 16 bitowa ze znakiem na wyjście "OUT"./1,2,3 **(Decrement Word)**: IN - 1 = OUT	
	Blok funkcyjny zwiększa wartość słowa wejściowego 32 bitowego ze znakiem (-2147483648 do +2147483647) doprowadzonego do wejścia "IN" o "1", gdy podany zostanie sygnał do wejścia "EN". Wynik wyprowadzony jest jako zmienna 32 bitowa ze znakiem na wyjście "OUT"./1,2,3 **(Increment Double Word)**: IN + 1 = OUT	IN:VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC
	Blok funkcyjny zmniejsza wartość słowa wejściowego 32 bito- wego ze znakiem (-2147483648 do +2147483647) doprowadzo- nego do wejścia "IN" o "1", gdy podany zostaje sygnał do wejścia "EN". Wynik wyprowadzony jest jako zmienna 32 bitowa ze znakiem na wyjście "OUT"./1,2,3 **(Decrement Double Word)**: IN - 1 = OUT	

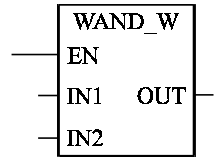
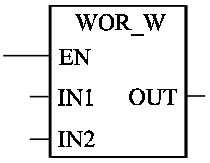
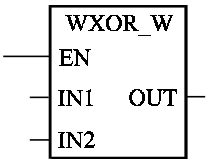
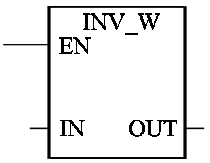
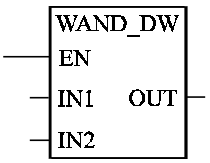
UWAGI: /1 *W przypadku, gdy wynik operacji przekracza zakres dopuszczalny bit specjalny SM1.1 ustawia się na "1",*

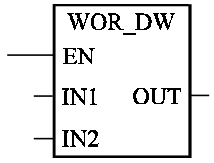
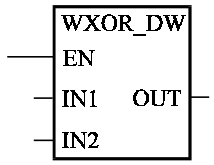
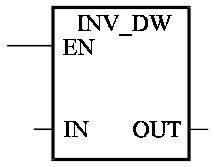
/2 *W przypadku, gdy wynik operacji wyniesie zero bit specjalny SM1.0 ustawia się na "1",*

/3 W przypadku, gdy wynik operacji jest ujemny bit specjalny SM1.2 ustawia się na "1",

/4 W przypadku, gdy wystąpi próba dzielenia przez zero bit specjalny SM1.3 ustawi się na "1".

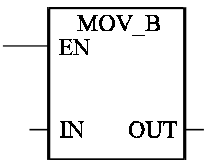
3. 7. Operacje logiczne

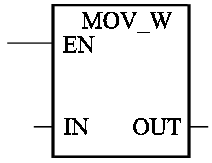
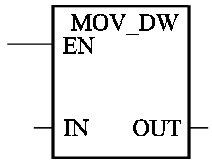
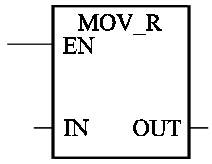
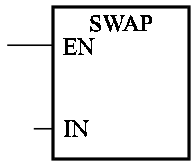
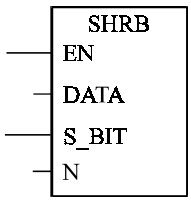
LAD	Opis	Zmienna
	AND (I) - koniunkcja dwóch słów 16 bitowych bez znaku (zakres od 0 do 65535) doprowadzonych do wejść: "IN1" oraz "IN2". Operacja zostaje wykonana po doprowadzeniu sygnału do wejścia "EN", a wynik przypisany do wyjścia "OUT". Operacja ta jest wykonywana dla każdej pary bitów o tych samych wagach począwszy od najmniej znaczących bitów./1 **(AND Word)**	IN1, IN2: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT:VW, T, C, IW, (słowo) QW, MW, SMW, AC
	OR (LUB) - alternatywa dwóch słów 16 bitowych bez znaku (zakres od 0 do 65535) doprowadzonych do wejść: "IN1" oraz "IN2". Operacja zostaje wykonana po doprowadzeniu sygnału do wejścia "EN", a wynik przypisany do wyjścia "OUT". Operacja ta jest wykonywana dla każdej pary bitów o tych samych wagach począwszy od najmniej znaczących bitów./1 **(OR Word)**	
	XOR (WYŁĄCZNIE - LUB) - suma modulo 2 dwóch słów 16 bitowych bez znaku (zakres od 0do 65535) doprowadzonych do wejść: "IN1" oraz "IN2". Operacja zostaje wykonana po doprowadzeniu sygnału do wejścia "EN", a wynik przypisany do wyjścia "OUT". Operacja ta jest wykonywana dla każdej pary bitów o tych samych wagach począwszy od najmniej znaczących bitów./1 **(XOR Word)**	IN1, IN2: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC
	NOT (NIE) - funkcja negacji, która powoduje zmianę stanu każdego bitu słowa 16 bitowego ze znakiem na przeciwny. Słowo to jest parametrem wejściowym "IN". Wynik operacji jest wyprowadzony na wyjście"OUT". Operacja ta jest wykonana, gdy do bloku funkcyjnego podany zostanie sygnał "EN"./1 **(Invert Word)**	IN:VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC
	AND (I) - koniunkcja dwóch słów 32 bitowych bez znaku (zakres od 0 do 4294967295) doprowadzonych do wejść: "IN1" oraz "IN2". Operacja zostaje wykonana po doprowadzeniu sygnału do wejścia "EN", a wynik przypisany do wyjścia "OUT". Operacja ta jest wykonywana dla każdej pary bitów o tych samych wagach począwszy od najmniej znaczących bitów./1 **(AND Doble Word)**	IN1, IN2: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC

	OR (LUB) - alternatywa dwóch słów 32 bitowych bez znaku (zakres od 0 do 4294967295) doprowadzonych do wejść: "IN1" oraz "IN2". Operacja zostaje wykonana po doprowadzeniu sygnału do wejścia "EN", a wynik przypisany do wyjścia "OUT". Operacja ta jest wykonywana dla każdej pary bitów o tych samych wagach począwszy od najmniej znaczących bitów./1 **(OR Doble Word)**	IN1, IN2: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC
	XOR (WYŁĄCZNIE - LUB) - suma modulo 2 dwóch słów 32 bitowych bez znaku (zakres od 0do 4294967295) doprowadzonych do wejść: "IN1" i "IN2". Operacja zostaje wykonana po doprowadzeniu sygnału do wejścia "EN", a wynik przypisany do wyj. "OUT". Operacja ta jest wykonywana dla każdej pary bitów o tych samych wagach począwszy od najmniej znaczących bitów./1 **(XOR Double Word)**	
	NOT (NIE) - funkcja negacji, która powoduje zmianę stanu każdego bitu słowa 32 bitowego ze znakiem na przeciwny. Słowo to jest parametrem wejściowym "IN". Operacja ta jest wykonana, gdy do bloku funkcyjnego podany zostaje sygnał "EN", a wynik jest przypisany do wyjścia "OUT". /1 **(Invert Double Word)**	IN:VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC

UWAGA: /1 *W przypadku, gdy wynik operacji wyniesie zero bit specjalny SM1.0 ustawia się na "1"*

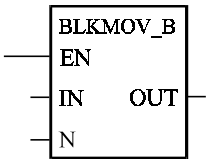
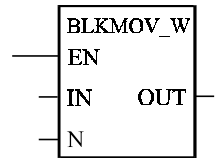
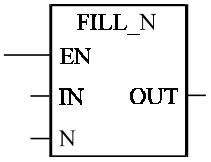
3. 8. Funkcje przemieszczenia

LAD	Opis	Zmienna
	Blok funkcyjny przemieszczający stałą bez znaku lub zmienną określoną 8 bitowym adresem doprowadzoną do wejścia "IN" w miejsce docelowe określone adresem na wyjściu "OUT". Zmienna wejściowa nie ulega zmianie po wykonaniu operacji przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. **(Move Byte)**	IN:VB, IB, QB, MB, (bajt) SMB, AC,K OUT: VB, IB, QB, MB, (bajt) SMB, AC

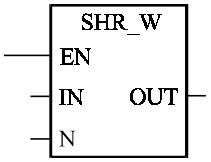
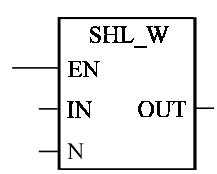
	Blok funkcyjny przemieszczający stałą bez znaku lub zmienną określoną 16 bitowym adresem doprowadzoną do wejścia "IN" w miejsce docelowe określone adresem na wyjściu "OUT". Zmienna wejściowa nie ulega zmianie po wykonaniu operacji przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. **(Move Word)**	IN:VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT:VW, T, C, IW, (słowo) QW, MW, SMW, AC, AQW
	Blok funkcyjny przemieszczający stałą bez znaku lub zmienną określoną 32 bitowym adresem doprowadzoną do wejścia "IN" w miejsce docelowe określone adresem na wyjściu "OUT". Zmienna wejściowa nie ulega zmianie po wykonaniu operacji przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. **(Move Double Word)**	IN:VD, ID, QD, MD, (32 bity) SMD, AC, HC, K
	Blok funkcyjny przemieszczający zmienną rzeczywistą 32-bitową "IN" w miejsce docelowe określone parametrem wyjściowym "OUT". Zmienna wejściowa nie ulega zmianie po wykonaniu operacji przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. **(Move Real)**	OUT: VD, ID, QD, MD, (32 bity) SMD, AC
	Blok funkcyjny powodujący zamianę miejscami dwóch bajtów (zmiennych 8 bitowych) słowa 16 bitowego - mniej (LSB) i bardziej (MSB) znaczącego. **(Swap)**	IN:VW, T, C, IW, (słowo) QW, MW, SMW, AC,
	Blok funkcyjny rejestru szeregowego przesuwającego o długości określonej parametrem "N" (max 64). Kierunek wpisywania danych "DATA" określa znak przy parametrze "N" ("+" lub "-" oznacza odpowiednio wpisywanie z prawej strony - w kierunku bitów rosnących lub z lewej strony - w kierunku bitów malejących). Parametr "S_BIT" określa pierwszy (w przypadku +N lub ostatni w przypadku -N bit rejestru szeregowego. Operacja przemieszczenia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. /1 **(Shift Register Bit)**	DATA, S_BIT:I, Q, M, SM, T, C, W, V N: VB, IB, QB, MB, SMB, AC, K

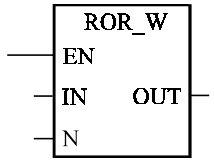
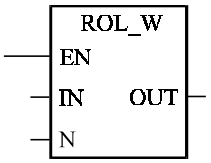
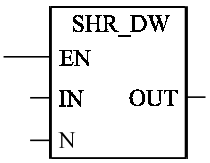
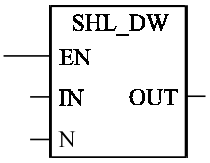
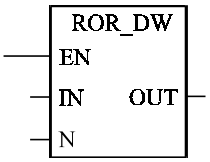
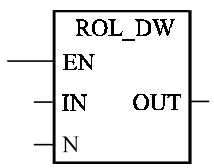
UWAGA: /1 *Bit specjalny SM1.1 zawiera bit rejestru szeregowego, który aktualnie jest z niego "wypychany".*

3. 9. Operacje na blokach danych

LAD	Opis	Zmienna
	Blok funkcyjny przemieszczający określoną parametrem "N" liczbę bajtów (zmiennych 8-bitowych) z przestrzeni o adresie początkowym "IN" do miejsca o adresie początkowym zdefiniowanym przez parametr "OUT". Ilość przemieszczanych bajtów "N" zawiera się w przedziale: (1 - 255). Operacja przemieszczenia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. **(Block Move Byte)**	IN:VB, MB, SMB, (bajt) IB, QB, OUT:VB, IB, QB, MB, (bajt) SMB, N: VB, IB, QB, MB, (bajt) SMB, AC, K
	Blok funkcyjny przemieszczający określoną parametrem "N" liczbę słów (zmiennych 16-bitowych) z przestrzeni o adresie początkowym "IN" do miejsca o adresie początkowym zdefiniowanym przez parametr "OUT". Ilość przemieszczanych bajtów "N" zawiera się w przedziale: (1 - 255). Operacja przemieszczenia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. **(Block Move Word)**	IN:VW, T, C, IW, AC, (słowo) QW, MW, SMW, OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AQW N:VB, IB, QB, MB, (bajt) SMB, AC, K
	Blok funkcyjny wypełniający przestrzeń adresową począwszy od adresu początkowego "OUT" słowami (zmiennymi 16-bitowymi) o wartości określonej parametrem "IN". Ilość przemieszczanych słów "N" zawiera się w przedziale: (1 - 255). Operacja wypełnienia zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał inicjujący. **(Memory Fill)**	IN:VW, T, C, IW, (słowo) QW, MW, SMW, AIW, K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AQW N:VB, IB, QB, MB, (bajt) SMB, AC, K

3. 10. Funkcje przesunięcia

LAD	Opis	Zmienna
	Blok funkcyjny przesuwający wszystkie bity jednego słowa (16 bitowego) wejściowego "IN" w prawo w kierunku bitów o male- jących wagach o wyszczególnioną parametrem "N" liczbę miejsc. W puste miejsca zostają wpisane zera. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygnału zezwalającego "EN". **(Shift Right Word)**./1,2	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K
	Blok funkcyjny przesuwający bity jednego słowa (16 bitowego) wejściowego "IN" w lewo w kierunku bitów o wzrastających wagach o wyszczególnioną parametrem "N" liczbę miejsc. W puste miejsca zostają wpisane zera. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygnału zezwalającego "EN". **(Shift Left Word)**./1,2	N: VB, IB, QB, K, (bajt) MB, SMB, AC, OUT:VW, T, C, IW, (słowo) QW, MW, SMW,

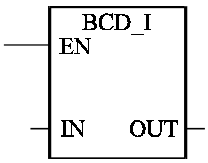
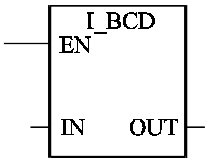
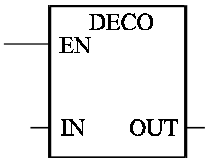
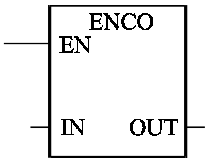
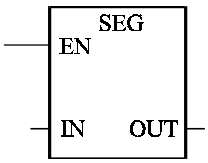
	Blok funkcyjny przesuwający bity jednego słowa (16 bitowego) wejściowego "IN", w obiegu zamkniętym (rotacja), w prawo o wyszczególnioną parametrem "N" liczbę miejsc, przy czym najmniej znaczące bity (wypchnięte z prawej strony słowa) zostają wpisane na puste miejsca z lewej strony słowa. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygnału wej. "EN". **(Rotate Right Word)**./1,2	
	Blok funkcyjny przesuwający bity jednego słowa (16 bitowego) wejściowego "IN", w obiegu zamkniętym (rotacja), w lewo o wyszczególnioną parametrem "N" liczbę miejsc, przy czym najbardziej znaczące bity (wypchnięte z lewej strony słowa) zostają wpisane na puste miejsca z prawej strony słowa. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygnału wej. "EN". **(Rotate Left Word)**./1,2	
	Blok funkcyjny przesuwający bity jednego słowa (32 bitowego) wejściowego "IN" w prawo w kierunku bitów o malejących wagach o wyszczególnioną parametrem "N" liczbę miejsc. W puste miejsca zostają wpisane zera. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygnału zezwalającego "EN". **(Shift Right Double Word)**./1,2	IN:VD, ID, QD, MD, (32 bity) SMD, AC, HC, K N:VB, IB, QB, K, (bajt) MB, SMB, AC OUT: VD, ID, QD, MD, (32bity) SMD, AC
	Blok funkcyjny przesuwający bity jednego słowa (32 bitowego) wejściowego "IN" w lewo w kierunku bitów o wzrastających wagach o wyszczególnioną parametrem "N" liczbę miejsc. W puste miejsca zostają wpisane zera. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygnału zezwalającego "EN". **(Shift Left Double Word)**./1,2	
	Blok funkcyjny przesuwający bity jednego słowa (32 bitowego) wejściowego "IN", w obiegu zamkniętym (rotacja), w prawo o wyszczególnioną parametrem "N" liczbę miejsc, przy czym najmniej znaczące bity (wypchnięte z prawej strony słowa) zostają wpisane na puste miejsca z lewej strony słowa. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygn. wej. "EN". **(Rotate Right Double Word)**./1,2	
	Blok funkcyjny przesuwający bity jednego słowa (32 bitowego) wej. "IN", w obiegu zamkniętym (rotacja), w lewo o wyszczególnioną parametrem "N" liczbę miejsc, przy czym najbardziej znaczące bity (wypchnięte z lewej strony słowa) zostają wpisane na puste miejsca z prawej strony słowa. Operacja, której wynik jest przypisany do wyjścia "OUT", dokonuje się w chwili pojawienia się sygnału wej. "EN". **(Rotate Left Double Word)**./1,2	

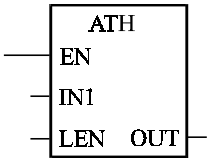
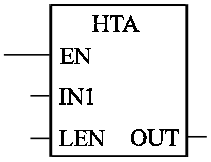
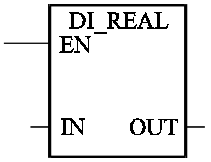
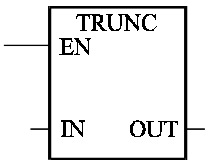
UWAGI:

/1Bt specjalny SM1.0 ustawia się na "1", gdy wynik danej operacji wynosi OUT = 0,

/2Bit specjalny SM1.1 ustawia się na "1", gdy ostatnim aktualnie bitem, podczas operacji przesuwania lub obrotu danego słowa, jest bit = 1.

3. 11. Funkcje konwersji

LAD	Opis	Zmienna
	Konwersja danych zapisanych w kodzie BCD-4 na dane zapisane jako liczby całkowite (Integer). Gdy do bloku funkcyjnego podany zostaje sygnał wejściowy "EN" zostaje dokonana konwersja zadanej przez parametr "IN" wartości, a jej wynik zostaje zapisany jako parametr wyjściowy "OUT"./1 **(BCD to Integer)**	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC,
	Konwersja danych zapisanych jako liczba całkowita (max. 9999) na równoważną liczbę zapisaną w kodzie BCD-4. Gdy do bloku funkcyjnego podany zostaje sygnał wejściowy "EN" zostaje dokonana konwersja zadanej przez parametr "IN" wartości, a jej wynik zostaje zapisany jako parametr wyjściowy "OUT"./1 **(Integer to BCD)**	
	Blok funkcyjny pozwalający na ustawienie wartości określonego bitu zmiennej wyjściowe "OUT" 16 bitowej. Pozostałe bity słowa wyjściowego ustawiane są na "0". Numer pozycji (0 do 15) ustawianego bitu określa zmienna "IN". Operacja zostaje dokonana po doprowadzeniu sygnału wejściowego "EN". **(Decode)**	IN: VB, IB, QB, MB, (bajt) SMB, AC,K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC,
	Blok funkcyjny pozwalający na lokalizację w słowie 16 bitowym wejściowym "IN" pierwszego bitu o wartości "1". Operacja ta jest dokonywana po dopłynięciu sygnału wejściowego "EN" i rozpoczyna się od najniższego bitu słowa wejściowego. Numer pozycji zlokalizowanego bitu, w zapisie dwójkowym, udostępniany jest na wyjście "OUT". **(Encode)**	IN:VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT:VB, IB, QB, MB, (bajt) SMB, AC
	Blok sterujący wyświetlaczem 7-segmentowym. Zakres przetwarzanych danych wejściowych "IN" zawiera się od 0 do 15. Dla liczb większych (max. 255) cykl konwersji powtarza się, i tak np: liczba 16 odpowiada 0, 17 - 1, ... , 31-15 itd. Konwersja zachodzi, gdy do wejścia "EN" dochodzi sygnał wejściowy. **(Segment)**	IN: VB, IB, QB, MB, (bajt) SMB, AC,K OUT: VB, IB, QB, MB, (bajt) SMB, AC

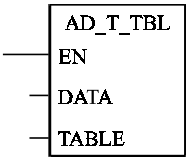
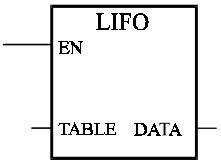
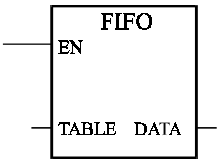
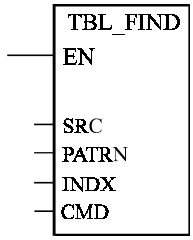
	Przetwarzanie ciągu znaków zapisanych w formacie ASCII o długości "LEN"" (1 do 255) rozpoczynających się od adresu początkowego "IN" na ciąg liczb heksadecymalnych o adresie początkowym określonym parametrem "OUT". W przypadku LEN=0 proces konwersji nie jest realizowany. Konwersja zachodzi, gdy do wejścia "EN" dochodzi sygnał wejściowy. **(ASCII to Hex)** Przetwarzane znaki ASCII (0-9 i A-F) przedstawione w postaci liczb heksadecymalnych zawierają się w prze- łach: (16#30 - 16#39) dla (0-9) i (16#41 - 16#46) dla (A-F)	LEN:VB, IB, QB, MB, (bajt) SMB, AC, K IN:VB, IB, QB, MB, (bajt) SMB, OUT:VB, IB, QB, MB, (bajt) SMB,
	Przetwarzanie ciągu znaków heksadecymalnych (0-9, A-F) o długości "LEN"" (max. 255) rozpoczynających się od adresu "IN" na ciąg znaków ASCII o adresie początkowym określonym parametrem "OUT". Konwersja zachodzi, gdy do wejścia "EN" dochodzi sygnał wejściowy./3 **(Hex to ASCII)**	IN:VB, IB, QB, MB, (bajt) SMB OUT: VB, IB, QB, MB, (bajt) SMB, LEN:VB, IB, QB, MB, (bajt) SMB, AC, K
	Blok funkcyjny przetwarzający zmienną 32-bitową ze znakiem (signet integer) na 32-bitową liczbę rzeczywistą o adresie "OUT". Konwersja zachodzi, gdy do wejścia "EN" dochodzi sygnał wejściowy. **Double Word Integer to Real (DI_REAL)**	IN: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT:VD, ID, QD, MD, (32 bity) SMD, AC
	Blok funkcyjny przetwarzający 32-bitową zmienną rzeczywistą "IN" na 32-bitową zmienną ze znakiem (signet integer) o adresie "OUT". Operacja konwersji zachodzi, gdy pierwszym bitem stosu jest "**1**". /2 **Truncate (TRUNC)**	IN:VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT:VD, ID, QD, MD, (32 bity) SMD, AC

UWAGA: /1 *Gdy zmienna wejścia "IN" bloku I_BCD lub BCD_I będzie miała wartość nieprawidłową , bit specjalny SM1.6 ustawia się na "1",*

/2 *Gdy zmienna wyjściowa przekroczy wartość dopuszczalną bit specjalny SM1.1 ustawi się na "1",*

/3 *W przypadku próby konwersji nieprawidłowego znaku ASCII bit specjalny SM1.7 ustawia si na "1".*

3. 12. Operacje na tablicach

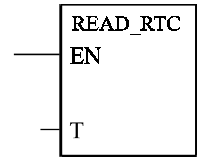
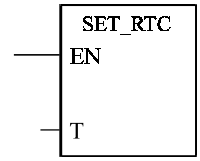
STL	Opis	Zmienna
	Blok umożliwiający wpisywanie kolejnych danych 16 bitowych do tablicy o adresie początkowym (TABLE). Adres ten wskazuje zmienną zawierającą zadaną pojemność tablicy (od 0 do 99 słów). Kolejny adres zawiera zmienną o aktualnej zawartości tablicy (ilość wprowadzonych słów). Pod kolejnym adresem umieszczona będzie pierwsza z wpisywanych danych 16 bitowych. Operacja wpisywania jest powtarzana za każdym razem, gdy do wejścia (EN) podany zostaje sygnał. /1,3 **Add To Table (AD_T_TBL)**	DATA: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K TABLE: VW, T, C, IW, (słowo) QW, MW, SMW
	Operacja pozwalająca na wyprowadzanie danych 16 bitowych (DATA) z tablicy o adresie (TABLE) w kolejności odwrotnej w stosunku do operacji wprowadzania. Operacja wyprowadzania jest powtarzana za każdym razem, gdy do wejścia (EN) podany zostaje sygnał. Wartość zmiennej określającej aktualną zawartość tablicy maleje za każdym razem o "1". /2,3 **Last-In-First-Out (LIFO)**	TABLE: VW, T, C, IW, (słowo) QW, MW, SMW DATA: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AQW
	Operacja pozwalająca na wyprowadzanie danych 16 bitowych (DATA) z tablicy o adresie (TABLE) w kolejności zgodnej w stosunku do operacji wprowadzania. Operacja wyprowadzania jest powtarzana za każdym razem, gdy do wejścia (EN) podany zostaje sygnał. Wartość zmiennej określającej aktualną zawartość tablicy maleje za każdym razem o "1". /2,3 **First-In-First-Out (FIFO)**	
	Blok funkcyjny pozwalający przeszukiwać zawartość wybranej tablicy, w której umieszczone są zmienne 16 bitowe. Parametr (SRC) określa miejsce w tablicy, od którego ma się rozpocząć proces przeszukiwania. Parametr (PATRN) określa wartość, z którą poszukiwany element tablicy ma pozostawać w określonej relacji. Typ relacji określa cyfra od 1 do 4 na wejściu (CMD), której przypisana jest odpowiednio relacja: (=, <>, <, i >). Numer - indeks (INDX) określa pozycję pierwszego, znalezionego elementu w przeszukiwanej tablicy. Aby przeszukać resztę tablicy należy zwiększyć o jeden numer indeksu (INDX) i ponownie uruchomić proces przeszukiwania. W przypadku nie znalezienia elementu parametr (INDX) równa się wartości licznika wejściowego tablicy (EC) określającej liczbę elementów przeszukiwanej tablicy. Operacja zostanie wykonana, gdy do wejścia (EN) podany zostaje sygnał. **Table Find (TBL_FIND)**	SRC:VW, T, C, IW, (słowo) QW, MW, SMW PATRN: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K INDX:VW, T, C, IW, (słowo) QW, MW, SMW, AC CMD: 1 do 4

UWAGA: /1 *W przypadku próby wpisania danej do zapełnionej wcześniej tablicy bit specjalny SM1.4 ustawia się na "1",*

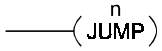
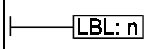
/2 *W przypadku próby wyprowadzenia danej z pustej tablicy bit specjalny SM1.5 ustawi się na "1",*

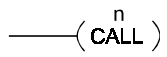
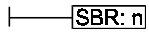
/* *Aby sprawdzić działanie powyższych operacji należy doprowadzić do wejścia "EN" sygnał wejściowy poprzez styk impulsowy "EU", który spowoduje ustawienie na czas jednego cyklu sygnału inicjującego daną operację.*

3. 13. Funkcje zapisu/odczytu zegara systemowego

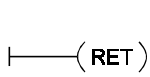
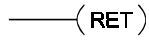
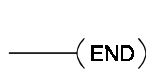
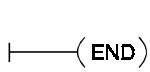
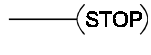
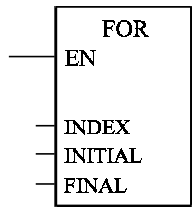
STL	Opis	Zmienna
	Blok funkcyjny umożliwiający odczyt bieżącej wartości czasu i daty z zegara wewnętrznego. Informacje te ładowane są do 8 bajtowego bufora danych. Adres początkowy wskazany przez parametr "T" jest adresem pierwszego bajtu. **Read Real Time Clock (READ_RTC)**	T: VB, IB, QB, MB, (bajt) SMB
	Instrukcja umożliwiająca wpisywanie bieżącej wartości czasu i daty do zegara wewnętrznego. Informacje te ładowane są do 8 bajtowego bufora zegarowego o adresie początkowym wskazanym przez parametr "T", który jest adresem pierwszego bajtu. **Set Real Time Clock (SET_RTC)**	

3. 14. Funkcje związane ze strukturą programu *(funkcje sterujące)*

STL	Opis	Zmienna
	Przekaźnik skoku warunkowego do podprogramu o adresie "n". Powoduje ona pominięcie części programu sterującego umiesz- czonego pomiędzy instrukcją "JUMP n", a etykietą "LABEL n". Instrukcja zostanie wykonana, gdy do przekaźnika podany zostaje sygnał. **Jump to Label (JMP)**	n = (0 do 63) dla CPU 212 n = (0 do 255) dla CPU 214
	Przekaźnik określający miejsce docelowe "n" skoku wywołanego przekaźnikiem "JUMP n" z tą samą etykietą i powoduje dalszą kontynuację wykonywania programu począwszy od tej etykiety. **Label (LBL)**	n = (0 do 63) dla CPU 212 n = (0 do 255) dla CPU 214

	Przekaźnik powodujący przeniesienie wykonywania programu sterującego do podprogramu o etykiecie "n". Po wykonaniu podprogramu następuje powrót do miejsca bezpośrednio występującego za przekaźnikiem "CALL n". Instrukcja zostanie wykonana, gdy do przekaźnika podany zostaje sygnał. **(CALL)**	n = (0 do 15) dla CPU 212 n = (0 do 63) dla CPU 214
	Przekaźnik wskazujący początek podprogramu "n" wywołanego przekaźnikiem "CALL n". Wszystkie podprogramy muszą być umieszczone na końcu głównego programu sterującego, tzn. za przekaźnikiem końca programu END. **Subroutine (SBR)**	n = (0 do 15) dla CPU 212 n = (0 do 63) dla CPU 214

Uwaga: *Przekaźniki CALL/SBR umożliwiają nakładanie się na siebie zakresów działania w ten sposób, że zakres działania jednej pary przekaźników CALL/SBR zawiera się wewnątrz innej pary przekaźników. Pokrywać się może do 9 takich obszarów.*

	Przekaźnik bezwarunkowego powrotu z podprogramu. Jest ostatnim elementem każdego podprogramu kończąc jego wykonywanie. **Return from Subroutine (RET)**	bez oznaczeń
	Przekaźnik warunkowego powrotu z podprogramu. Przekaźnik może być wykorzystany do opuszczania danego podprogramu. Może również wystąpić przed przekaźnikiem bezwarunkowego zakończenia podprogramu "RET". Instrukcja zostanie wykonana, gdy do przekaźnika podany zostaje sygnał. **Return from Subroutine (RET)**	bez oznaczeń
	Przekaźnik zakończenia wykonywania części logicznej programu. Przekaźnik spowoduje zatrzymanie wykonywania programu w miejscu, w którym występuje i rozpoczęcie cyklu skanowania od początku. Instrukcja zostanie wykonana, gdy do przekaźnika podany zostaje sygnał. **(END)**	bez oznaczeń
	Przekaźnik o działaniu bezwarunkowym. Jest ostatnim elementem programu sterującego. Powoduje rozpoczęcie przez sterownik skanowania programu od początku. **(END)**	bez oznaczeń
	Przekaźnik ten kończy wykonywanie programu i powoduje natychmiastowe przejście sterownika w tryb STOP. Instrukcja zostanie wykonana, gdy do przekaźnika podany zostaje sygnał. **(STOP)**	bez oznaczeń
	Blok funkcyjny powodujący wielokrotne powtórzenie fragmentu programu sterującego zawartego między instrukcjami "FOR" i "NEXT". Ilość tych powtórzeń określają parametry - początkowy "INITIAL" oraz końcowy "FINAL". Parametr "INDEX" określa aktualną liczbę powtórzeń (wykonanych pętli), gdyż jego zawartość wzrasta o jeden po zakończeniu każdej pętli (powtórzenia). Jeśli aktualna wartość parametru "INDEX" będzie większa od wartości parametru końcowego "FINAL", nastąpi zakończenie wykonywania pętli (opuszczenie pętli). Instrukcja	INDEX: VW, T, C, IW, (słowo) QW, MW, SMW, AC, INITIAL: VW, T, C, IW, (słowo) QW, MW, AC, SMW, AIW, K FINAL: VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K

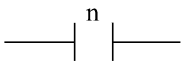
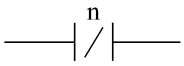
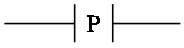
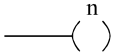
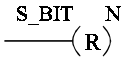
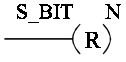
	wykonywania pętli (opuszczenie pętli). Instrukcja zostanie wykonana, gdy do wejścia "EN" podany zostaje sygnał. **(FOR)** np: gdy wartość INITIAL = 1, a FINAL = 10, to fragment programu zawarty pomiędzy blokami "FOR - NEXT" zostanie powtórzony 10 razy.	SMW, AC, AIW, K
⊢——(NEXT)	Przekaźnik, o działaniu bezwarunkowym, określa koniec obszaru programu sterującego, który podlega wielokrotnemu "zapętleniu". Ustawia pierwszy bit stosu na "**1**". **(NEXT)**	bez oznaczeń

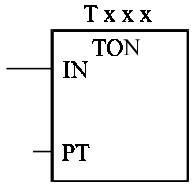
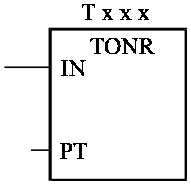
Uwaga: *Instrukcje FOR/NEXT umożliwiają nakładanie się na siebie zakresów działania w ten sposób, że zakres działania jednej pary instrukcji FOR/NEXT zawiera się wewnątrz innej pary instrukcji. Pokrywać się może do 9 takich obszarów.*

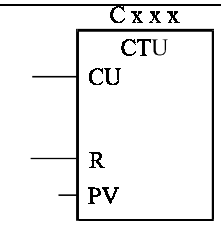
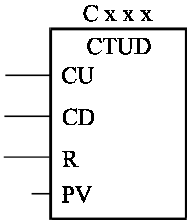
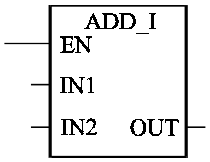
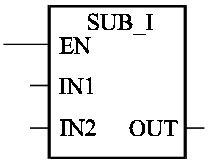
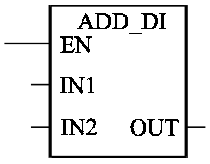
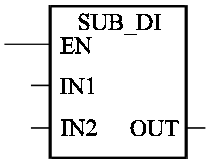
4. Załącznik

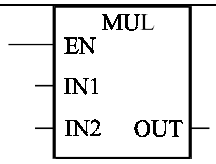
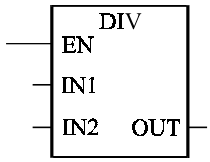
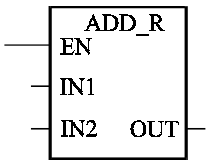
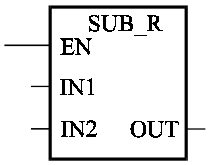
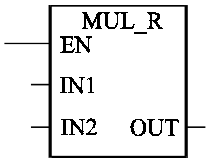
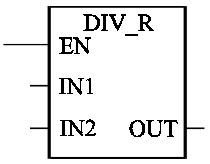
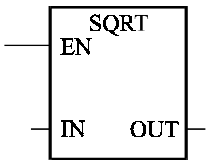
zmienne systemowe	Przeznaczenie
Zmienne systemowe - tylko do odczytu	
SM0.0	Bit ten jest zawsze ustawiony na "**1**" w każdym cyklu skanowania.
SM0.1	Bit ten jest ustawiony na "**1**" tylko w czasie pierwszego cyklu skanowania.
SM0.2	Bit ustawiany na "**1**" w pierwszym cyklu skanowania, gdy zmienne z pamięcią stanu w pamięci RAM zostały skasowane.
SM0.3	Bit ustawiany na "**1**" w czasie pierwszego cyklu skanowania po przejściu w tryb RUN po ponownym doprowadzeniu napięcia zasilającego.
SM0.4	Bit sterowany impulsami zegarowymi. Zmienia swój stan co 30 sekund.
SM0.5	Bit sterowany impulsami zegarowymi. Zmienia swój stan co 0,5 sekundy.
SM0.6	Bit zmieniający stan na przeciwny w kolejnych cyklach zegarowych.
SM0.7	Bit odzwierciedlający pozycję zewnętrznego przełącznika trybu pracy (pozycja TERM/STOP - bit ustawiony na "**0**", pozycja RUN - bit ustawiony na "**1**").
Uwaga: *Stan powyższych bitów jest uaktualniany na końcu każdego cyklu skanowania*	
SM1.0	Bit ustawiany na "1", gdy efektem określonej operacji jest cyfra "0".
SM1.1	Bit ustawiany na "1", gdy efektem określonej operacji jest przekroczenie zakresu lub niedozwolona wartość parametru.
SM1.2	Bit ustawiany na "**1**", gdy wynikiem operacji matematycznej jest liczba ujemna.
SM1.3	Bit ustawiany na "**1**", gdy wystąpiła próba dzielenia przez zero.
SM1.4	Bit ustawiany na "**1**", gdy wystąpi próba wczytania zmiennej do zapełnionej tablicy danych.
SM1.5	Bit ustawiany na "**1**", gdy wystąpi próba czytania zmiennej z pustej tablicy danych przy użyciu instrukcji LIFO lub FIFO.
SM1.6	Bit ustawiany na "**1**", gdy wystąpi próba konwersji niewłaściwej zmiennej wejściowej w bloku BDC_I.
SM1.7	Bit ustawiany na "**1**", gdy nastąpi próba konwersji niewłaściwej zmiennej wejściowej w bloku ATH.
SMB6	Rejestr identyfikujący: parametry jednostki CPU sterownika, rodzaj oraz liczbę wejść i wyjść. Dla jednostki centralnej CPU-212 nr id. ma postać: 00000101, a dla CPU-214 wynosi: 00101010.
SMB7	Rejestr przechowujący informacje o ewentualnych błędach, które wystąpiły w obw. wejść/wyjść sterownika PLC.
SMB8-21	Rejestry o adresach parzystych (8,10,12,14,16,18,20) identyfikujące odpowiednio parametry modułów rozszerzeń o numerach (0-6) dotyczące: typu modułu, rodzaju i liczby wejść/wyjść. W drugiej grupie rejestrów o adresach nieparzystych (9,11,13,15, 17,19,21) przechowywane są informacje związane z błędami, które wystąpiły w poszczególnych modułach. Czyli każdy z modułów opisany jest parą rejestrów. Dla CPU-212 bajty od SM12 do SM21 są niewykorzystane, gdyż jednostka ta może być rozbudowana max. o dwa moduły zewnętrzne.
SMW22	Rejestr 16 - bitowy zapisuje czas ostatniego cyklu skanowania w [ms].
SMW24	Rejestr 16 - bitowy zapisuje minimalny czas cyklu skanowania liczony od momentu uruchomienia sterownika w [ms].
SMW26	Rejestr 16 - bitowy zapisuje maksymalny czas cyklu skanowania liczony od momentu uruchomienia sterownika w [ms].
SMB28-29	Rejestry reprezentujące aktualną pozycję potencjometrów analogowych 0 i 1 sterownika.

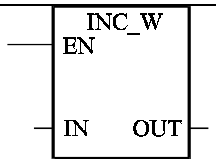
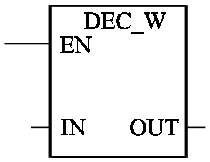
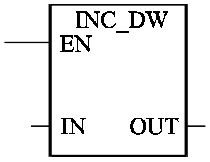
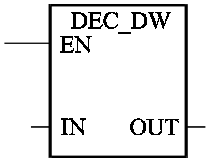
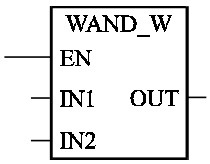
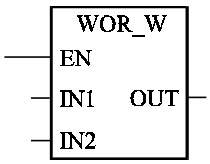
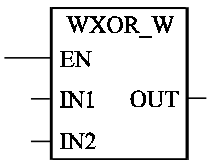
Zmienne systemowe - typu odczyt/zapis	
SMB30	Rejestr umożliwiający konfigurowanie portu komunikacyjnego w trybie swobodnym (Freeport). Przez ustawienie odpowiednich wartości bitów tego rejestru można wybrać parytet parzystości, szybkość transmisji danych, odpowiedni protokół komunikacyjny (PPI/Freeport)
SMB31	Rejestr umożliwiający zapisywanie zmiennych z obszaru adresowego (0-199) dla CPU-212 lub (0-1023) dla CPU-214 w nieulotnej pamięci sterownika EEPROM. Pierwszy bit bajtu SMB31 wskazuje na rozkaz zachowania wartości, a dwa ostatnie określają format zmiennej (bit, bajt, słowo, zmienna 32-bitowa) .
SMW32	W pierwszych dziesięciu bitach tego bajtu podaje się adres zmiennej przeznaczonej do zapamiętania w nieulotnej pamięci EEPROM: 000000aaaaaaaaaa.
SM34-35	Rejestry określające przedział czasowy dla przerwań programowych w zakresie od 0 do 255ms z rozdzielczością co 1ms.
SMB36-65	Rejestry te służą do monitorowania i sterowania funkcjami szybkich liczników (HSC).
SMB66-85	Rejestry te służą do monitorowania i sterowania wyjściami impulsowymi o stałym 50% wypełnieniu - PTO lub z modulacją szerokości impulsu (PWM).

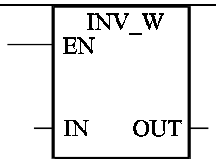
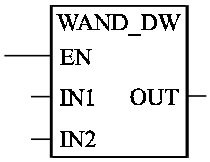
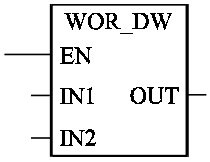
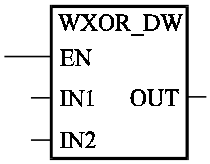
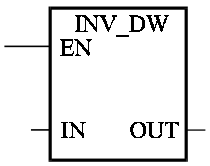
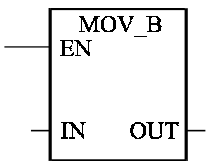
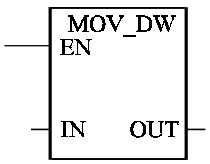
LAD	STL	Opis	str.
	LD n	Styk normalnie otwarty	19
	LDN n	Styk normalnie zamknięty	19
	NOT	Styk negacji	19
	EU	Styk impulsowy reagujący na zbocze narastające	19
	ED	Styk impulsowy reagujący na zbocze opadające	19
	= n	Przekaźnik wyjściowy o stykach normalnie otwartych	19
	S S_BIT, N	Przekaźnik ustawialny "SET"	19
	R S_BIT, N	Przekaźnik ustawialny "RESET"	19
	B = n1, n2	Styki komparatorów 8 bitowych	20
	B > = n1, n2		
	B < = n1, n2		

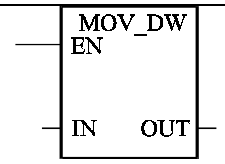
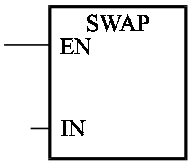
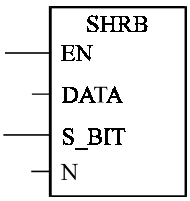
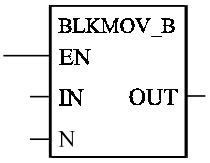
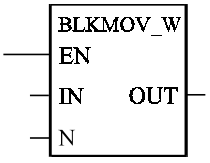
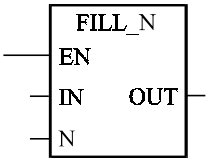
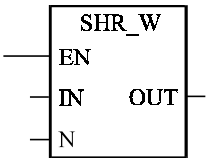
	W = n1, n2	Styki komparatorów 16 bitowych	20
	W > = n1, n2		
	W < = n1, n2		
	D = n1, n2	Styki komparatorów 32 bitowych	20
	D > = n1, n2		
	D < = n1, n2		
	TON Txxx, PT	Timer bez pamięci	21
	TONR Txxx, PT	Timer z pamięcią	21
	CTU Cxxx, PV	Licznik zliczający w górę (do przodu)	22

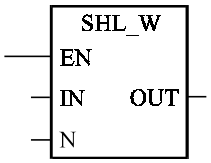
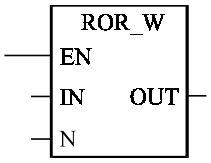
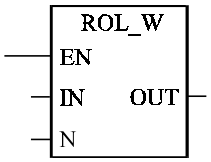
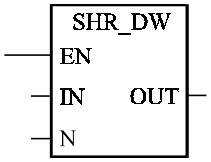
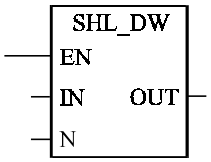
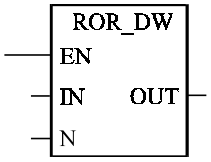
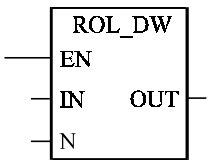
			
	CTUD Cxxx, PV	Licznik zliczający w dół (do tyłu)	22
	+I IN1, IN2	Operacja dodawania dwóch zmiennych 16 bitowych	22
	-I IN1, IN2	Operacja odejmowania dwóch zmiennych 16 bitowych	22
	+D IN1, IN2	Operacja dodawania dwóch zmiennych 32 bitowych	22
	-D IN1, IN2	Operacja odejmowania dwóch zmiennych 32 bitowych	23

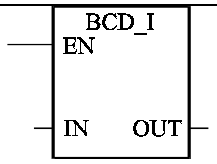
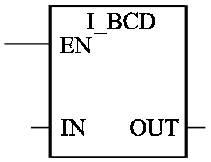
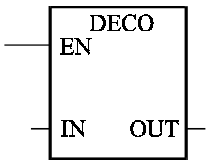
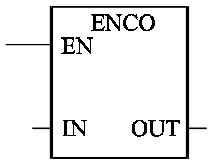
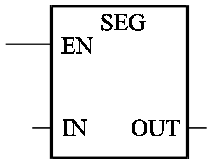
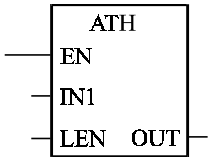
	MUL IN1, IN2	Operacja mnożenia dwóch zmiennych 16 bitowych	23
	DIV IN1, IN2	Operacja dzielenia dwóch zmiennych 16 bitowych	24
	+R IN1, IN2	Operacja dodawania dwóch zmiennych rzeczywistych 32-bitowych	23
	-R IN1, IN2	Operacja odejmowania dwóch zmiennych rzeczywistych 32-bitowych	23
	*R IN1, IN2	Operacja mnożenia dwóch zmiennych rzeczywistych 32-bitowych	23
	/R IN1, IN2	Operacja dzielenia dwóch zmiennych rzeczywistych 32-bitowych	24
	SQRT IN, OUT	Pierwiastek kwadratowy ze zmiennej 32 bitowej	24

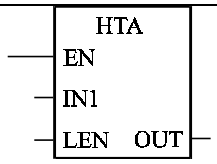
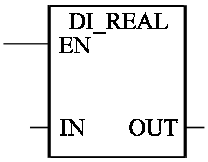
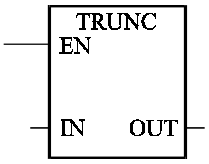
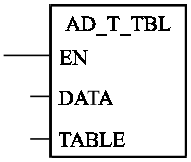
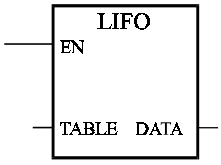
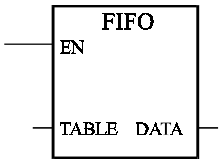
	INCW IN	Zwiększenie wartości zmiennej 16 bitowej o jeden	24
	DECW IN	Zmniejszenie wartości zmiennej 16 bitowej o jeden	24
	INCD IN	Zwiększenie wartości zmiennej 32 bitowej o jeden	24
	DECD IN	Zmniejszenie wartości zmiennej 32 bitowej o jeden	24
	ANDW IN1, IN2	Iloczyn logiczny dwóch zmiennych 16 bitowych	25
	ORW IN1, IN2	Suma logiczna dwóch zmiennych 16 bitowych	25
	XORW IN1, IN2	XOR (Wyłącznie - LUB) dwóch zmiennych 16 bitowych	25

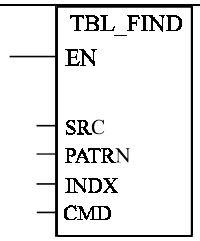
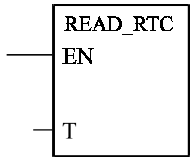
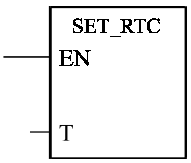
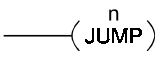
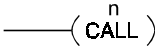
	INVW IN	Inwersja zmiennej 16 bitowej	25
	ANDD IN1, IN2	Iloczyn logiczny dwóch zmiennych 32 bitowych	25
	ORD IN1, IN2	Suma logiczna dwóch zmiennych 32 bitowych	26
	XORD IN1, IN2	XOR (Wyłącznie - LUB) dwóch zmiennych 32 bitowych	26
	INVD IN	Inwersja zmiennej 32 bitowej	26
	MOVB IN, OUT	Przeniesienie zmiennej wejściowej 8 bitowej na wyjście OUT	26
	MOVW IN, OUT	Przeniesienie zmiennej wejściowej 16 bitowej na wyjście OUT	27
	MOVD IN, OUT	Przeniesienie zmiennej wejściowej 32 bitowej na wyjście OUT	27

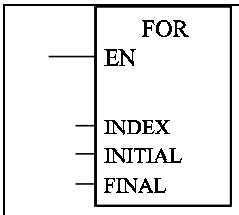
			
	SWAP IN	Zamiana miejscami dwóch bajtów słowa 16 bitowego	27
	SHRB DATA, S_BIT, N	Rejestr szeregowy	27
	BMB IN, OUT, N	Przemieszczenie zmiennych 8-bitowych w liczbie "N"	28
	BMW IN, OUT, N	Przemieszczenie zmiennych 16-bitowych w liczbie "N"	28
	FILL IN, OUT, N	Wypełnienie zmiennymi 16-bitowymi w liczbie "N"	28
	SRW IN, N	Przesunięcie zmiennej 16 bitowej o wyszczególnioną liczbę miejsc w prawo	28

	SLW IN, N	Przesunięcie zmiennej 16 bitowej w lewo o wyszczególnioną liczbę miejsc	28
	RRW IN, N	Rotacja zmiennej 16 bitowej w prawo o wyszczególnioną liczbę miejsc	29
	RLW IN, N	Rotacja zmiennej 16 bitowej w lewo o wyszczególnioną liczbę miejsc	29
	SRD IN, N	Przesunięcie zmiennej 32 bitowej o wyszczególnioną liczbę miejsc w prawo	29
	SLD IN, N	Przesunięcie zmiennej 32 bitowej o wyszczególnioną liczbę miejsc w lewo	29
	RRD IN, N	Rotacja zmiennej 32 bitowej w prawo o wyszczególnioną liczbę miejsc	29
	RLD IN, N	Rotacja zmiennej 32 bitowej w lewo o wyszczególnioną liczbę miejsc	29

	BCDI IN	Konwersja danych zapisanych w kodzie BCD-4 na liczby całkowite	30
	IBCD IN	Konwersja danych zapisanych jako liczby całkowite na zapis w BCD-4	30
	DECO IN, OUT	Ustawianie wartości określonego bitu zmiennej 16 bitowej na "1"	30
	ENCO IN, OUT	Lokalizacja w zmiennej 16 bitowej pierwszego bitu o wartości "1"	30
	SEG IN, OUT	Blok sterujący pracą wyświetlacza 7 - segmentowego	30
	ATH IN, OUT, LEN	Konwersja znaków w formacie ASCII na format heksadecymalny	31
	HTA IN, OUT, LEN	Konwersja znaków w formacie heksadecymalnym na ciąg znaków ASCII	31

			
	DTR IN, OUT	Przetwarzanie zmienne 32-bitowej na 32-bitową liczbę rzeczywistą	31
	TRUNC IN, OUT	Przetwarzanie zmiennej rzeczywistej na zmienną 32-bitową ze znakiem	31
	ATT DATA, TABLE	Wpisywanie zmiennych 16-bitowych do tablicy zmiennych	32
	LIFO TABLE, DATA	Wyprowadzanie zmiennych 16-bitowych z tablicy danych	32
	FIFO TABLE, DATA	Wyprowadzanie zmiennych 16-bitowych z tablicy danych	32
	FND xx SRC, PATRN,	Przeszukiwanie zawartości tablicy danych	32

	INDX		
	TODR T	Odczyt aktualnej wartości czasu i daty	33
	TODW T	Ustawianie aktualnej wartości czasu i daty	33
	JMP n	Skok warunkowy	33
	LBL n	Etykieta skoku "JUMP"	33
	CALL n	Wywołanie podprogramu	34
	SBR n	Etykieta wywołania podprogramu "CALL"	34
	RET	Bezwarunkowy powrót z podprogramu	34
	CRET	Warunkowy powrót z podprogramu	34
	END	Warunkowe zakończenie programu	34
	MEND	Bezwarunkowe zakończenie programu	34
	STOP	Warunkowe zakończenie programu	34
	FOR INDEX INITIAL	Wielokrotne powtarzanie wykonywania fragmentu programu	34

	FINAL		
	NEXT	Koniec obszaru fragmentu programuwielokrotnie powtarzanego	35