S P I S T R E Ś C I

1. Język programowania STL ..... 2 str.
2. Pojęcie stosu ..... 3 str.

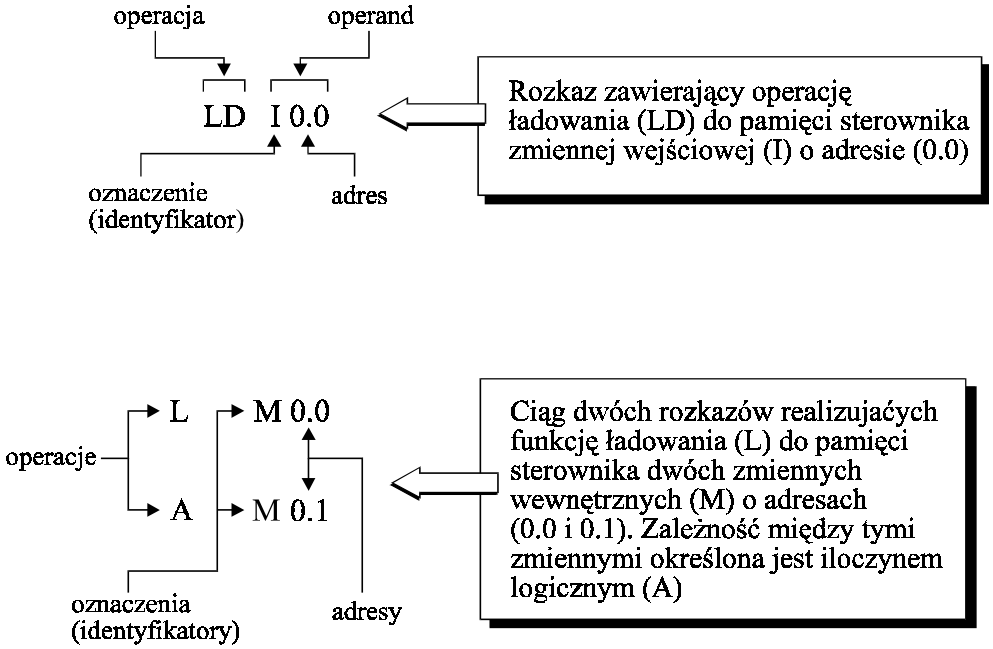
   2. 1. Funkcja ładuj - LD ..... 4 str.
   2. 2. Iloczyn i suma logiczna - AND/OR ..... 4 str.
   2. 3. Funkcja ALD oraz OLD ..... 5 str.
   2. 4. Funkcja przyporządkowania ..... 5 str.
   2. 5. Funkcje logiczne - LPS/LRD/LPP ..... 6 str.
   2. 6. Operacje logiczne - NOT/EU/ED ..... 7 str.

3. Związki funkcyjne pomiędzy stosem a rozkazami programu sterującego ..... 8 str.
4. Przykłady identyfikatorów zmiennych ..... 12 str.
5. Przestrzeń adresowa jednostek centralnych ..... 13 str.
6. Opis podstawowych rozkazów oprogramowania Micro/WIN ..... 16 str.

   6. 1. Styki ..... 16 str.
   6. 2. Przekaźniki ..... 16 str.
   6. 3. Operacje logiczne na stosie ..... 17 str.
   6. 4. Komparatory ..... 17 str.
   6. 5. Timery ..... 19 str.
   6. 6. Liczniki ..... 19 str.
   6. 7. Operacje matematyczne ..... 20 str.
   6. 8. Operacje logiczne ..... 24 str.
   6. 9. Funkcje przemieszczenia ..... 26 str.
   6. 10. Funkcje na blokach danych ..... 27 str.
   6. 11. Funkcje związane z tablicami danych ..... 28 str.
   6. 12. Funkcje przesunięcia ..... 31 str.
   6. 13. Funkcje konwersji ..... 34 str.
   6. 14. Funkcje związane z zegarem czasu rzeczywistego ..... 35 str.
   6. 15. Funkcje związane ze strukturą programu ..... 36 str.

1. Język programowania STL (Statement List)

Jest to zapis symboliczny stanowiący ciąg kolejno ułożonych jedna pod drugą instrukcji (rozkazów). Jest to zapis bardziej zwarty operujący na skrótach literowych symbolizujących np: wejścia/wyjścia sterownika, zmienne wewnętrzne, operacje logiczne i matematyczne oraz inne bloki funkcyjne. Rozkaz jest najmniejszą jednostką programu i składa się z operacji i operandu. Operacja określa rodzaj czynności jaka w danym rozkazie ma być wykonana, natomiast operand składa się z oznaczenia i parametru. Oznaczeniem jest skrót literowy określonej zmiennej binarnej, rejestru lub innego bloku funkcyjnego, a parametrem jest adres zmiennej (zmiennych) podlegającej danej operacji w bieżącym rozkazie, wykonywanego przez sterownik, programu sterującego.

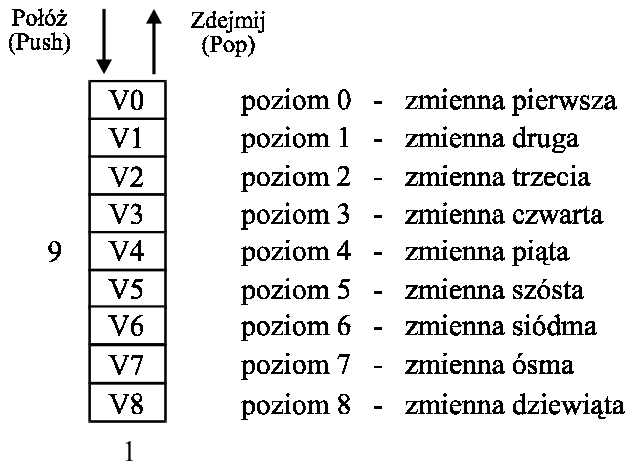


Rys.1 Budowa i sposób zapisu przykładowych rozkazów w STL obowiązujących w oprogramowaniu STEP7.

Sterownik mikroprocesorowy wykonując dany program analizuje każdy pojedynczy rozkaz osobno krok po kroku w kolejności ich występowania począwszy od rozkazu pierwszego, a skończywszy na ostatnim (z góry na dół). Po realizacji ostatniego rozkazu mikroprocesor rozpoczyna proces analizy od początku. Proces ten nazywa się cyklicznym wykonywaniem programu, a czas potrzebny na przeanalizowanie wszystkich rozkazów danego programu nazywa się czasem jednego cyklu.

2. Stos

W celu dokładniejszego zrozumienia idei pisania programów w języku STL oraz umiejętności poprawnej analizy zapisu takiego programu ważna jest świadomość tego, w jaki sposób jednostka CPU sterowników S7 - 200 wykorzystuje tzw. STOS. Stos stanowi specyficzny sposób organizacji przechowywania danych. Reprezentację graficzną stosu przedstawia poniższy rysunek:

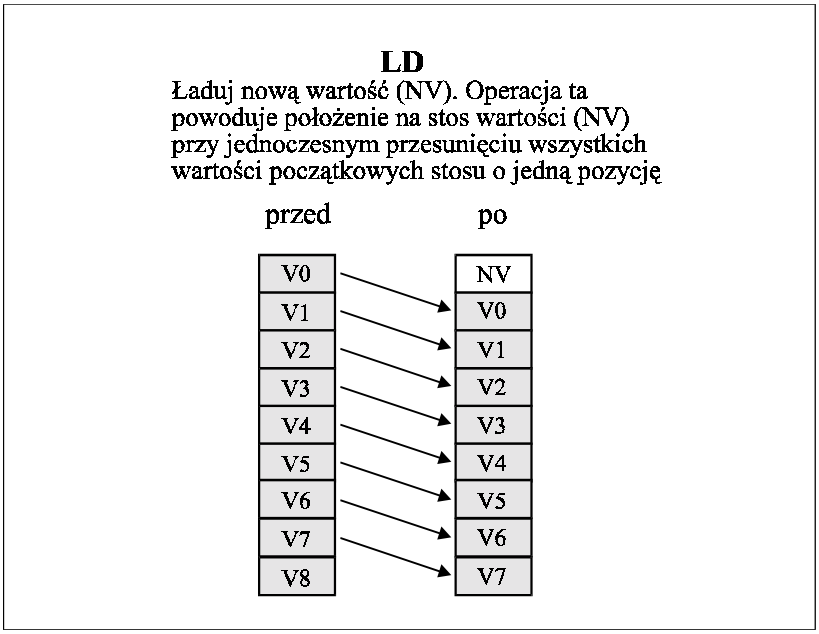


Rys.2. Stos o wymiarze 9 x 1 bitów

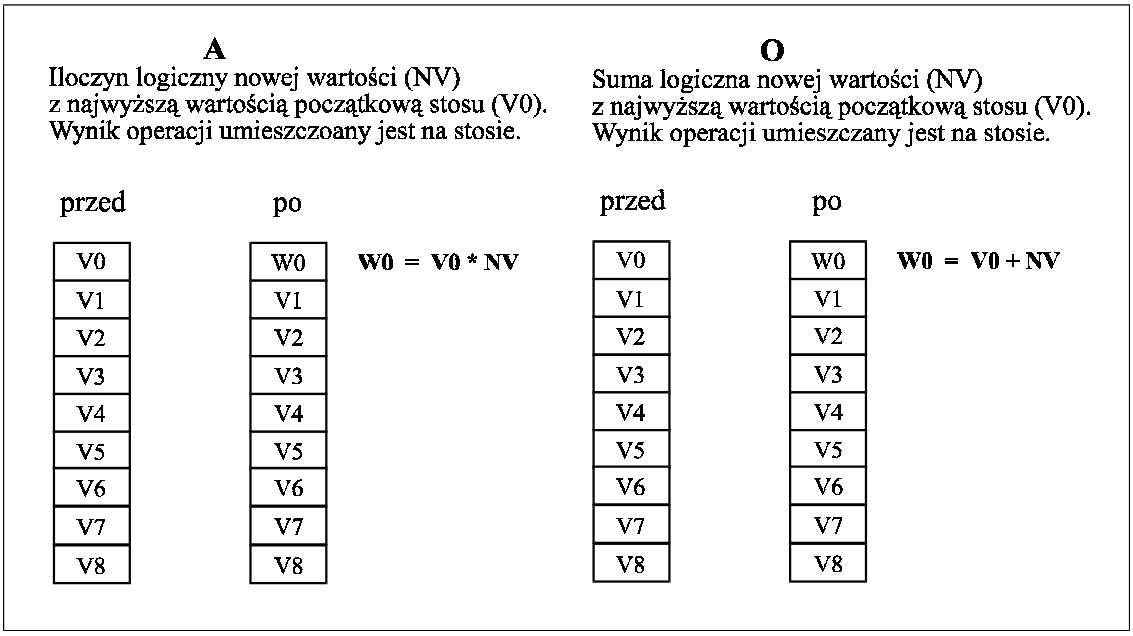
Stos przypomina strukturę wykorzystującą 9 bitów ułożonych jeden na drugim. Podstawową zasadą wykorzystania stosu jest zdejmowanie (Pop) ze stosu danych w odwrotnej kolejności do ich układania (Push). Praktycznie każda operacja logiczna i blok funkcyjny w języku STEP7 są bezpośrednio związane są z funkcją STOS-u. Przykłady przedstawione na następnej stronie omawiają sposób
wykorzystania STOS-u:

2. 1. Funkcja ładuj - Load (LD)

Funkcja może być wykorzystana w programie wielokrotnie. Zapoczątkowuje określone fragmenty, z których składa się cały program sterujący. Operacja "load" zawsze poprzedza blok funkcyjny. Ładowana zmienna (bit) wprowadzana jest na sam wierzch stosu:

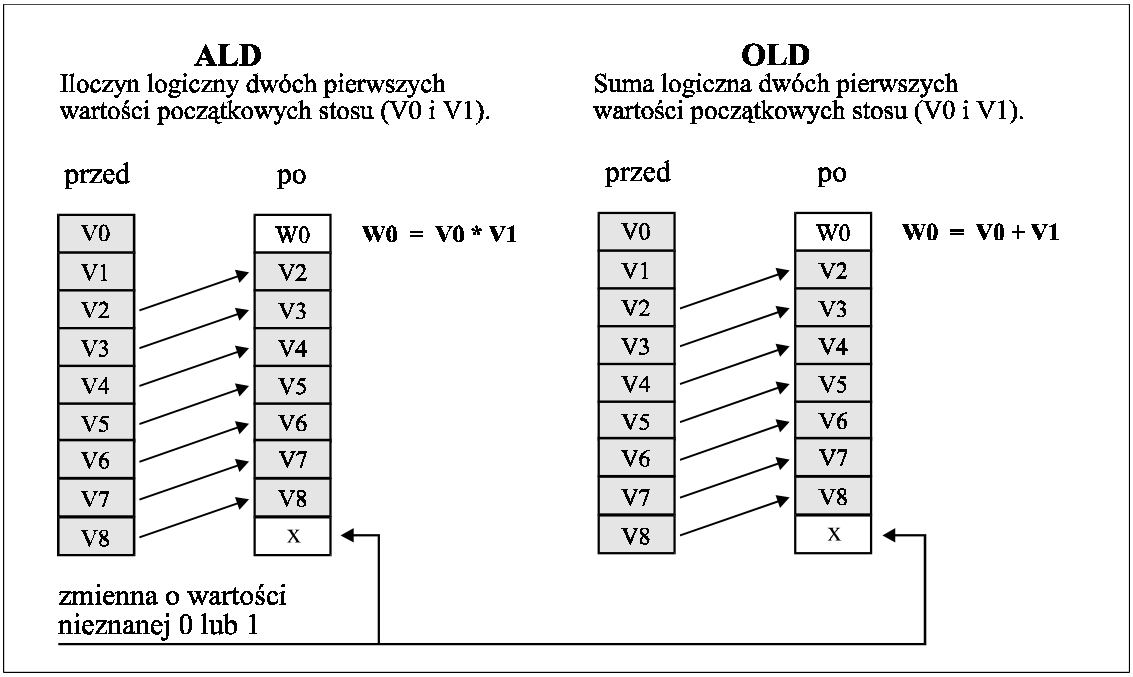


2. 2. Iloczyn i suma logiczna zmiennych dyskretnych - And (A) i Or (O)



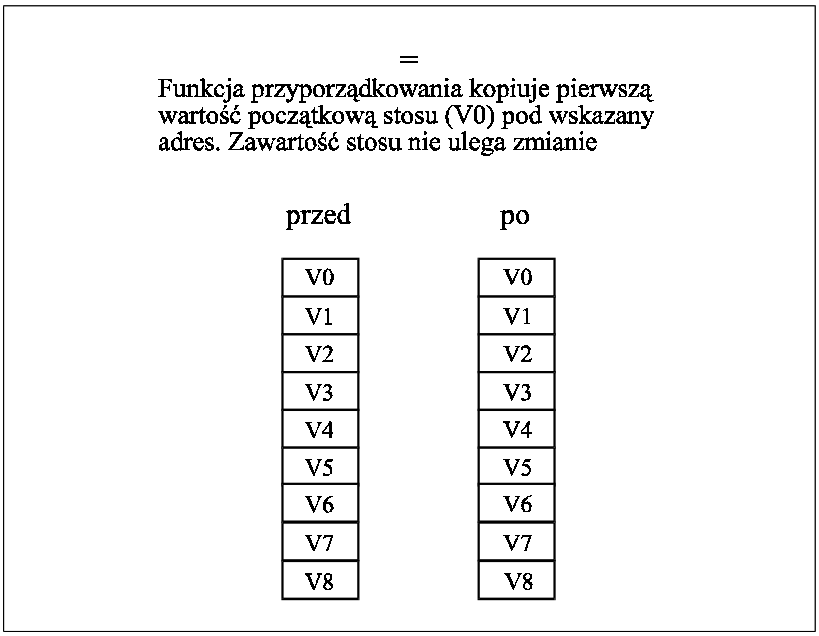
2. 3. Funkcja And Load (ALD) oraz Or Load (OLD)

Funkcje logiczne (ALD) i (OLD) operują na dwóch pierwszych bitach wierzchołka stosu (V0 i V1). Wynik logiczny tych działań lokalizowany jest na samym wierzchu stosu:

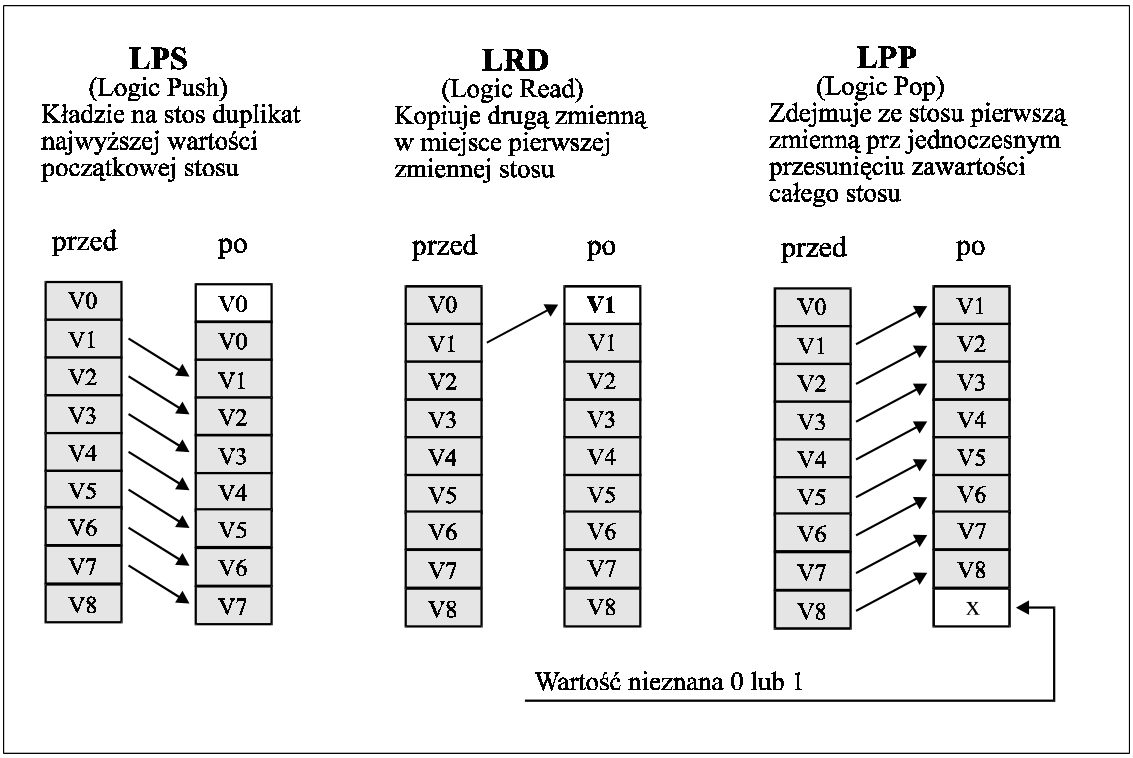


2. 4. Funkcja przyporządkowania Output (=)

Operacja ta kopiuje pierwszy bit stosu (V0) w miejsce określone adresem umieszczonym obok operacji (=). Zawartość stosu po operacji nie ulega zmianie:

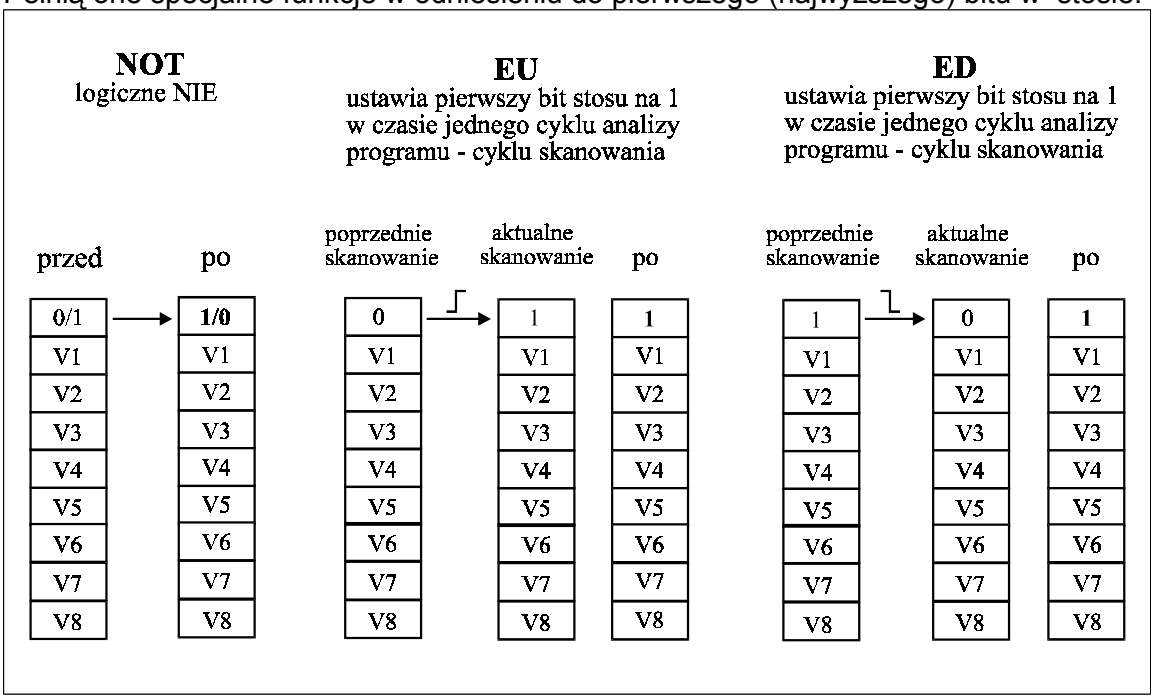


2. 5. Funkcje logiczne Logic Push/Read/Pop



2. 6. Operacje logiczne NOT/EU/ED

Pełnią one specjalne funkcje w odniesieniu do pierwszego (najwyższego) bitu w stosie.



Funkcja NOT powoduje za każdym razem zmianę stanu bitu pierwszego. Operacje EU oraz ED wykrywają zmianę stanu najwyższego bitu w stosie między kolejnymi dwoma cyklami analizy programu sterującego. W efekcie ustawiają najwyższy bit stosu na 1. Operacja EU (Edge Up) wykrywa zmianę stanu pierwszego bitu stosu z 0 na 1, natomiast ED (Edge Down) zmianę stanu pierwszego bitu stosu z 1 na 0.

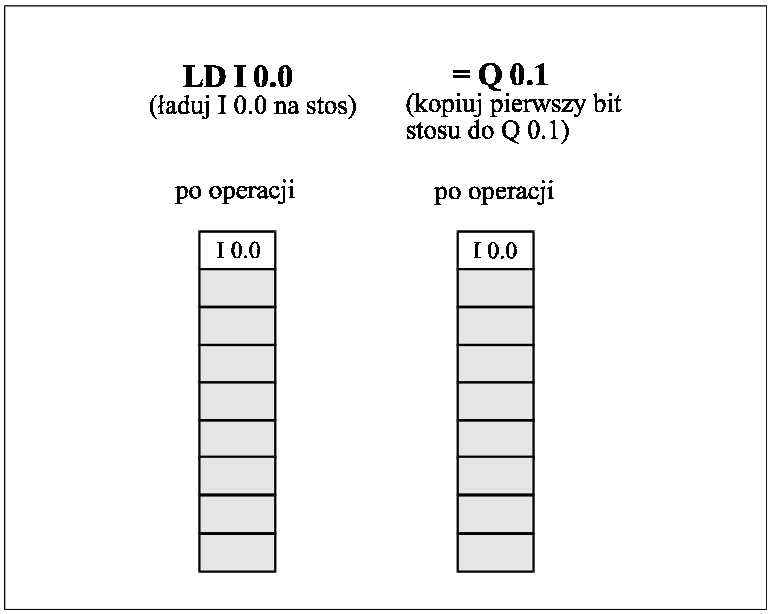
Poniższe przykłady ilustrują sytuację zmiany zawartości stosu podczas wykonania fragmentów programu:

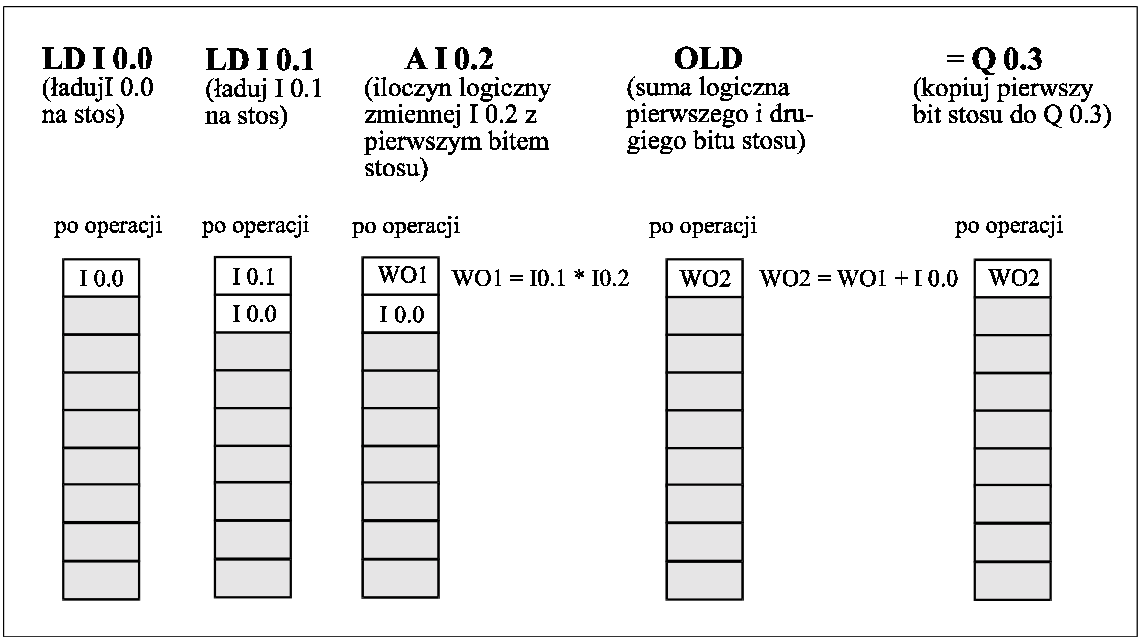
Przykład 1:

```
1. LD    I 0.0
2. =     Q 0.1
```

Przykład 2:

```
1. LD    I 0.0
2. LD    I 0.1
3. A     I 0.2
4. OLD
5. =     Q 0.3
```





3. Związki funkcyjne między stosem a rozkazami programu sterującego

Rozkazy oprogramowania STEP-7 do sterowników Simatic S7 - 200 można podzielić na dwie zasadnicze grupy:

1) rozkazy, których wykonanie jest niezależne od aktualnej wartości pierwszego (najwyższego) bitu stosu oraz
2) rozkazy, których realizacja jest uzależniona od aktualnej wartości pierwszego bitu położonego na stosie. Warunkiem wykonania tych operacji jest to, by wartość

tego bitu była równa **1**.

Kolejne dwie tabele dzielą rozkazy, w porządku alfabetycznym, na wyżej wymienione kategorie. W tabelach zawarte są kompletne oznaczenia rozkazów w języku STL oraz zawierają krótką interpretację pełnionych przez nie funkcji.

Rozkazy realizowane niezależnie od aktualnej zawartości stosu

Oznaczenie	Charakterystyka
= n	Pierwszy bit stosu kopiowany w miejsce określone adresem zmiennej n
A n	Iloczyn logiczny zmiennej n z wartością pierwszego bitu stosu
AB < = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 < = n2
AB = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 = n2
AB > = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 > = n2
AD < = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 < = n2
AD = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 = n2
AD > = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 > = n2
AR < = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 < = n2 (tylko CPU214)
AR = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 = n2 (tylko CPU214)
AR > = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 > = n2 (tylko CPU214)
ALD	Iloczyn logiczny pierwszego i drugiego bitu stosu. Wynik operacji jest pierwszym bitem stosu
AN n	Iloczyn logiczny pierwszego bitu stosu z zanegowaną wartością zmiennej n
AW < = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 < = n2
AW = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 = n2
AW > = n1, n2	Iloczyn logiczny pierwszego bitu stosu z **1**, gdy n1 > = n2
CTU Cxxx, PV	Licznik zliczający impulsy wejściowe (drugi bit stosu) w górę. Sygnał kasujący zawartość licznika jest pierwszym bitem stosu.
Rozkazy realizowane niezależnie od aktualnej zawartości stosu - c.d.	
CTUD Cxxx, PV	Licznik zliczający impulsy w górę (trzeci bit w stosie) lub w dół (drugi bit w stosie). Sygnał kasujący zawartość licznika jest pierwszym bitem stosu
ED	Ustawia pierwszy bit stosu na **1**, gdy w dwóch kolejnych cyklach skanowania bit ten zmienił wartość z 0 na 1. (impuls)
EU	Ustawia pierwszy bit stosu na **1**, gdy w dwóch kolejnych cyklach skanowania bit ten zmienił wartość z 1 na 0. (impuls)
LBL xxx	Początek podprogramu określony etykietą xxx
LD n	Ładowanie na stos wartości zmiennej n
LDB < = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 < = n2
LDB = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 = n2
LDB > = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 > = n2
LDD < = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 < = n2
LDD = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 = n2
LDD > = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 > = n2
LDN n	Ładowanie na stos zanegowanej wartości wyszczególnionej

	adresem n
LDR < = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 < = n2 (tylko PU214)
LDR = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 = n2 (tylko CPU214)
LDR > = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 > = n2 (tylko CPU214)
LDW < = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 < = n2
LDW = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 = n2
LDW > = n1, n2	Ładowanie na stos bitu o wartości **1**, gdy n1 > = n2
LPP	Zdejmowanie ze stosu pierwszego bitu. Drugi bit stosu po przesunięciu staje się bitem pierwszym
LPS	Połóż na stos duplikat pierwszego bitu tego stosu
LRD	Kopiuj wartość drugiego bitu w stosie na pozycję pierwszą
MEND	Zakończenie cyklu skanowania (analizy programu sterującego)
NOT	Zanegowanie wartości pierwszego bitu w stosie
O n	Suma logiczna pierwszego bitu w stosie ze zmienną o adresie n
OB < = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 < = n2
OB = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 = n2
OB > = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 > = n2

Rozkazy realizowane niezależnie od aktualnej zawartości stosu - c.d.

OD < = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 < = n2
OD = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 = n2
OD > = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 > = n2
OLD	Suma logiczna pierwszego i drugiego bitu stosu. Wynik operacji jest pierwszym bitem stosu
ON n	Suma logiczna pierwszego bitu w stosie i zanegowanej zmiennej n
OR < = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 < = n2 (tylko CPU214)
OR = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 = n2 (tylko 214)
OR > = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 > = n2 (tylko CPU214)
OW < = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 < = n2
OW = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 = n2
OW > = n1, n2	Suma logiczna pierwszego bitu stosu z **1**, gdy n1 > = n2
RET	Zakończenie wykonywania podprogramu

Uwaga: Dodatkowe oznaczenia literowe uzupełniające właściwe operacje to:

B (Byte) - bajt (zmienna 8 bitowa),
W (Word) - słowo (zmienna 16 bitowa)
I (Integer) - liczba całkowita ze znakiem (zmienna 16 bitowa),
D (Double Integer) - liczba całkowita ze znakiem (zmienna 32 bitowa),
R (Real) - liczba rzeczywista (zmienna 32 bitowa).

Rozkazy realizowane, gdy pierwszy bit w stosie ma wartość **1**. (Top of Stack = 1)

+ D IN1, IN2	Dodawanie dwóch liczb 32 bitowych IN1 i IN2 ze znakiem
+ I IN1, IN2	Dodawanie dwóch liczb 16 bitowych IN1 i IN2 ze znakiem
- D IN1, IN2	Odejmowanie dwóch liczb 32 bitowych IN1 i IN2 ze znakiem
- I IN1, IN2	Odejmowanie dwóch liczb 16 bitowych IN1 i IN2 ze znakiem
+ R IN1, IN2	Dodawanie dwóch liczb rzeczywistych IN1 i IN2 (tylko CPU214)
- R IN1, IN2	Odejmowanie dwóch liczb rzeczywistych IN1 i IN2 (tylko CPU214)
* R IN1, IN2	Mnożenie dwóch liczb rzeczywistych IN1 i IN2 (tylko CPU214)
/ R IN1, IN2	Dzielenie dwóch liczb rzeczywistych IN1 i IN2 (tylko CPU214)
ANDD IN1, IN2	Iloczyn logiczny na zmiennych 32 bitowych ze znakiem
ANDW IN1, IN2	Iloczyn logiczny na zmiennych 16 bitowych
ATH IN, OUT, LEN	Konwersja znaków ASCII na system szesnastkowy (HEX)
Rozkazy realizowane, gdy pierwszy bit w stosie ma wartość **1**. - c.d. (Top of Stack = 1)	
ATT DATA, TABLE	Wpisywanie zmiennej 16 bitowej do tablicy danych (tylko CPU214)
BCDI IN	Konwersja zmiennej IN w kodzie BCD na zmienną 16 bitową
BMB IN, OUT, N	Wielokrotne przemieszczanie bajtów z przestrzeni IN do przestrzeni OUT
BMW IN, OUT, N	Wielokrotne przemieszczanie słów z przestrzeni IN do przestrzeni OUT
CALL n	Przejście do podprogramu
CRET	Zakończenie wykonywania podprogramu
DECD IN	Zwiększenie wartości zmiennej IN 32 bitowej o **1**
DECO IN, OUT	Ustawianie wybranego bitu zmiennej 16 bitowej na **1**
DECW IN	Zwiększenie wartości zmiennej 16 bitowej IN o **1**
DIV IN1, IN2	Dzielenie dwóch zmiennych 16 - bitowych
ENCO IN, OUT	Lokalizacja w zmiennej 16 bitowej pierwszego bitu o wartości **1**
END	Zatrzymanie cyklu skanowania (analizy programu sterującego)
FIFO TABLE, DATA	Wyprowadzanie zmiennych 16 bitowych z tablicy danych w kolejności zgodnej do ich wpisywania
FILL IN, OUT, N	Zapełnianie przestrzeni OUT zmienną wejściową IN w liczbie N
FND x SCR, PATRN, INDX	Wyszukiwanie wszystkich zmiennych 16 bitowych w tabeli danych według określonego kryterium: x = " < " lub " < > " lub " = " lub " > ".
FOR INDEX, INITIAL, FINAL	Rozpoczęcie wykonywania pętli programu sterującego
HTA IN, OUT, LEN	Konwersja zmiennej zapisanej w kodzie HEX na format ASCII
IBCD IN	Konwersja liczby całkowitej IN (max. 9999)na zmienną 16 bitową
INCD IN	Zmniejszenie wartości zmiennej IN 32 bitowej o **1**
INCW IN	Zmniejszenie wartości zmiennej 16 bitowej IN o **1**
INVD IN	Zamiana stanów bitów zmiennej 32 bitowej IN na przeciwny (inwersja)
INVW IN	Zamiana stanów bitów zmiennej 16 bitowej IN na przeciwny (inwersja)
JMP xxx	Instrukcja skoku do podprogramu wskazanego etykietą LBL xxx
LIFO TABLE, DATA	Wyprowadzanie zmiennych 16 bitowych z tablicy danych w kolejności odwrotnej do ich wpisywania
MOVB IN, OUT	Przesuwanie zmiennej 8 bitowej IN w miejsce wskazane adresem OUT
MOVD IN, OUT	Przesuwanie zmiennej 32 bitowej IN w miejsce wskazane adresem OUT
MOVR IN, OUT	Przesuwanie zmiennej rzeczywistej 32 bitowej w miejsce o adresie OUT (tylko CPU214)

MOVW IN, OUT	Przesuwanie zmiennej 16 bitowej IN w miejsce wskazane adresem OUT
Rozkazy realizowane, gdy pierwszy bit w stosie ma wartość **1**. - c.d. (Top of Stack = 1)	
MUL IN1, IN2	Mnożenie liczb 16 bitowych IN1 i IN2 ze znakiem
ORD IN1, IN2	Suma logiczna zmiennych 32 bitowych IN1 i IN2
ORW IN1, IN2	Suma logiczna dwóch zmiennych 16 bitowych IN1 i IN2
R S_BIT, n	Kasowanie kolejnych n bitów począwszy od adresu S_BIT
RLD IN, N	Rotacja zmiennej 32 bitowej IN o N pozycji w lewo
RLW IN, N	Rotacja zmiennej 16 bitowej IN o N pozycji w lewo
RRD IN, N	Rotacja zmiennej 32 bitowej IN o N pozycji w prawo
RRW IN, N	Rotacja zmiennej 16 bitowej IN o N pozycji w prawo
S S_BIT, n	Ustawianie kolejnych n bitów począwszy od adresu S_BIT
SEG IN, OUT	Dekoder sterujący wyświetlaczem 7 - segmentowym
SHRB DATA, S_BIT, N	Rejestr szeregowy dwukierunkowy o zmiennej długości (+/-N)
SLD IN, N	Przesuwanie zmiennej 32 bitowej IN o N pozycji w lewo
SLW IN, N	Przesuwanie zmiennej 16 bitowej IN o N pozycji w lewo
SQRT IN, OUT	Pierwiastek kwadratowy z 32 bitowej liczby rzeczywistej (tylko CPU214)
SRD IN, N	Przesuwanie zmiennej 32 bitowej IN o N pozycji w prawo
SRW IN, N	Przesuwanie zmiennej 16 bitowej IN o N pozycji w prawo
STOP	Zatrzymanie cyklu skanowania i przejście w tryb STOP
SWAP IN	Zamiana miejscami dwóch bajtów zmiennej 16 bitowej
TODR T	Odczyt aktualnej daty i czasu z zegara systemowego (tylko CPU214)
TODW T	Ustawianie aktualnej daty i czasu (tylko CPU214)
TON Txxx, PT	Timer o numerze xxx bez pamięci
TONR Txxx, PT	Timer o numerze xxx z pamięcią
XORD IN1, IN2	Wyłącznie - LUB dwóch zmiennych 32 bitowych IN1 i IN2
XORW IN1, IN2	Wyłącznie - LUB dwóch zmiennych 16 bitowych IN1 i IN2

Uwaga: Dodatkowe oznaczenia literowe uzupełniające właściwe operacje to:

B (Byte) - bajt (zmienna 8 bitowa),
W (Word) - słowo (zmienna 16 bitowa)
I (Integer) - liczba całkowita ze znakiem (zmienna 16 bitowa),
D (Double Word) - liczba całkowita ze znakiem (zmienna 32 bitowa),
R (Real) - liczba rzeczywista (zmienna 32 bitowa).

4. Przykłady 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	-

5. Przestrzeń adresowa jednostek centralnych

Przy pisaniu programów obok oznaczenia literowego identyfikatora należy podać odpowiednią cyfrę (liczbę) określającą miejsce (adres) w 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 … 0 LSB*	MSB* 7 … 0 LSB*
Zmienne pamięciowe - odczyt/zapis (V - memory)	V0	V0
	Pamięć nieulotna od V0 do V199	Pamięć nieulotna od V0 do V1023
	V199	V1023
Zmienne pamięciowe - odczyt/zapis (V - memory)	V200	V1024
	. . .	. . .
	V1023	V4095
	I0.7... I0.0	I0.7... I0.0
	.	.

Rejestry wejściowe - odczyt/zapis	. . I7.7... I7.0	. . I7.7... I7.0
Rejestry wyjściowe - odczyt/zapis	Q0.7 ... Q0.0 . . . Q7.7 ... Q7.0	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 ... SM29.0	SM0.7 ... SM0.0 . . . SM29.7 ... SM29.0
Wewnętrzne zmienne specjalne (systemowe) - odczyt/zapis	SM30.7 ... SM30.0 . . . SM45.7 ... SM45.0	SM30.7 ... SM30.0 . . . SM85.7 ... SM85.0

	CPU - 212	CPU - 214
	MSB* 15 … LSB* 0	MSB* 15 … LSB* 0
Timery - odczyt/zapis	T0	T0
	. . .	. . .
	T63	T127
Liczniki - odczyt/zapis	C0	C0
	. . .	. . .
	C63	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

6. Opis podstawowych rozkazów oprogramowania Micro/WIN

6. 1. Styki

STL	Opis	Zmienna
LD n	Instrukcja **Load** ładuje na stos wartość zmiennej n.	n: I, Q, M, SM, T, C, V (bit)
LDN n	Instrukcja **Load Not** ładuje na stos zanegowaną wartość n.	bez dodatkowych oznaczeń
NOT	Zmiana wartości pierwszego bitu stosu na przeciwną **-** negacja.	
EU	Instrukcja **Edge Up** powoduje ustawienie, na czas jednego cyklu, pierwszego bitu stosu na "1", gdy w dwóch kolejnych cyklach skanowania programu bit ten zmienił wartość z "0" na "1". Limit ilości styku impulsowego dla CPU 212 wynosi 128, a dla CPU 214 256.	
ED	Instrukcja **Edge Down** powoduje chwilowe ustawienie, na czas jednego cyklu, pierwszego bitu stosu na "1", gdy w dwóch kolejnych cyklach skanowania programu bit ten zmienił wartość z "1" na "0". Limit ilości styku impulsowego dla CPU 212 wynosi 128, a dla CPU 214 256.	
= n	Instrukcja **Output** powoduje kopiowanie pierwszego bitu stosu do zmiennej określonej adresem n.	n: I, Q, M, SM, T, C, V (bit)

6. 2. Przekaźniki

S S_BIT, N	Instrukcja **SET** ustawia na "1" kilka zmiennych dyskretnych, których liczbę określa parametr "N". Parametr S_BIT określa adres początkowy tych zmiennych. Ich stan nie zmienia się nawet w przypadku wyłączenia zasilania sterownika lub po przejściu sterownika w tryb STOP. Wartość "1" jest utrzymywana do chwili wywołania instrukcji "RESET".	S_BIT: I, Q, M, SM, T, (bit) C, V N: IB, QB, MB, SMB, (bajt) VB, AC, K
R S_BIT, N	Instrukcja **RESET** jest identyczna do "SET" z tą różnicą, że powoduje ustawienie zmiennych dyskretnych na "0".	

6. 3. Operacje logiczne na stosie

ALD	Wynikiem działania instrukcji **And Load** jest iloczynlogiczny pierwszego i drugiego bitu w stosie. Wynik staje się pierwszym bitem w stosie.	bez dodatkowych oznaczeń
OLD	Wynikiem działania instrukcji **Or Load** jest suma logiczna pierwszego i drugiego bitu w stosie. Wynik staje się pierwszym bitem w stosie.	
LPS	Instrukcja **Logic Push** powoduje położenie na stos duplikatu pierwszego bitu w tym stosie.	
LRD	Instrukcja **Logic Read** kopiuje drugi bit stosu w miejsce pierwszego.	
LPP	Instrukcja **Logic Pop** powoduje zdjęcie ze stosu pierwszego bitu. Jednocześnie bit drugi staje się pierwszym w tym stosie.	

6. 4. Komparatory

STL	Opis	Zmienna
LDB = n1, n2 LDW = n1, n2 LDD = n1, n2	Instrukcje powodujące załadowanie (LD) bitu na stos, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D./1 Bit wynikowy porównania wynosi "**1**", gdy n1 = n2.	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
AB = n1, n2 AW = n1, n2 AD = n1, n2	Instrukcje wykonujące operacje iloczynu logicznego (A) pierwszego bitu stosu z bitem, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 = n2.	
OB = n1, n2 OW = n1, n2 OD = n1, n2	Instrukcje wykonujące operacje sumy logicznej (O) pierwszego bitu stosu z bitem, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 = n2.	
LDB > = n1, n2 LDW > = n1, n2 LDD > = n1, n2	Instrukcje powodujące załadowanie (LD) bitu na stos, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 > = n2.	

AB > = n1, n2 AW > = n1, n2 AD > = n1, n2	Instrukcje wykonujące operacje iloczynu logicznego (A) pierwszego bitu stosu z bitem, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 > = n2.	
OB > = n1, n2 OW > = n1, n2 OD > = n1, n2	Instrukcje wykonujące operacje sumy logicznej (O) pierwszego bitu stosu z bitem, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 > = n2.	
LDB < = n1, n2 LDW < = n1, n2 LDD < = n1, n2	Instrukcje powodujące załadowanie (LD) bitu na stos, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 < = n2.	
AB < = n1, n2 AW < = n1, n2 AD < = n1, n2	Instrukcje wykonujące operacje iloczynu logicznego (A) pierwszego bitu stosu z bitem, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 < = n2.	
OB < = n1, n2 OW < = n1, n2 OD < = n1, n2	Instrukcje wykonujące operacje sumy logicznej (O) pierwszego bitu stosu z bitem, który jest wynikiem porównania dwóch zmiennych "n1" i "n2" określonego typu: B, W, D. /1 Bit wynikowy porównania wynosi "**1**", gdy n1 < = n2.	

UWAGA: /1 *Skróty B, W, D oznaczają:*
B bajt (zmienna 8-bitowa) - Byte,
W słowo (zmienna 16-bitowa) - Word Integer,
D podwójne słowo (zmienna 32-bitowa) - Double Word Integer.

6. 5. Timery

STL	Opis	Zmienna
TONR T xxx, PT np: LD I0.0 TONR T65, 15 (Txxx) (PT)	Timer zlicza jednostki czasu, gdy pierwszym bitem stosu jest "**1**" i zatrzymuje naliczoną wartość, gdy bit ten równa się "**0**". Zliczanie czasu jest kontynuowane, gdy pierwszy bit stosu ponownie będzie miał wartość "**1**". Zakres zliczanych jednostek czasu określony parametrem "PT" wynosi max: +32767 jednostek czasu. Gdy wartość bieżąca zrówna się z wartością zadaną "PT" timer zostaje uaktywniony. (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 - 31, 64 - 95 PT: VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K
TON T xxx, PT np: LD I0.1 TON T90, VW260 (Txxx) (PT)	Timer zlicza jednostki czasu, gdy pierwszym bitem stosu jest "**1**". Zostaje wyzerowany, gdy bit ten będzie równy "**0**". Zakres zliczanych jednostek czasu określony parametrem "PT" wynosi max: +32767. Gdy wartość bieżąca zrówna się z wartością zadaną "PT" timer zostaje uaktywniony. (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

6. 6. Liczniki

STL	Opis	Zmienna
CTU C xxx, PV np: LD I0.0 - zliczanie w górę LD I0.2 -	Licznik zliczający w górę. Zlicza impulsy wejściowe (zbocze narastające) pochodzące od zmian wartości drugiego bitu w stosie. Zakres licznika "PV" wynosi 32767 impulsów. Licznik jest wyzerowany tak długo, jak długo pierwszy bit stosu równa się "**1**". Przy zrównaniu się liczby zliczanych impulsów z wartością zadaną "PV" licznik zostaje uaktywniony. (bit	Cxxx: CPU 212: 0 - 63 (słowo) CPU 214: 0 - 127, PV: VW, T, C, IW, QW,

kasowanie CTU C63, 10 (Cxxx) (PV)	wyjściowy licznika "Cxxx"ustawia się na "1"). **(Count Up)**	(słowo) MW, SMW, AC, AIW, K
CTUD C xxx, PV np: LD I0.0 - zliczanie w górę LD I0.1 - zliczanie w dół LD I0.2 - kasowanie CTUD C48, 4 (Cxxx) (PV)	Licznik zliczający w górę i w dół. Zlicza impulsy wejściowe (zbocze narastające) pochodzące od zmian wartości drugiego (zliczanie w dół) i trzeciego (zliczanie w górę) bitu w stosie. Zakres zliczanych impulsów "PV" wynosi: -32768 do +32767. Licznik jest wyzerowany tak długo, jak długo pierwszy bit stosu równa się "**1**". Przy zrównaniu się liczby zliczanych impulsów z wartością zadaną "PV" licznik zostaje uaktywniony. (bit wyjściowy licznika "Cxxx" ustawia się na "1"). **(Count Up/Down)**	Cxxx: CPU 212: 0 - 63 (słowo) CPU 214: 0 - 127, PV: VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K

6. 7. Operacje matematyczne

STL	Opis	Zmienna
+I IN1, IN2 np: LD I0.0 +I VW50, VW60 /* (IN1) (IN2) czyli: VW50 + VW60 = VW60	Sumowanie dwóch liczb całkowitych 16 bitowych ze znakiem (Signet Integer - IN1 i IN2) w zakresie: -32768 do +32767. Wynik jest zmienną 16 - bitową ze znakiem IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu będzie równy "**1**". /1,2,3 **Add Integer (+I):** IN1 + IN2 = IN2	IN1: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K IN2: VW, T, C, IW, (słowo) QW, MW, SMW, AC
-I IN1, IN2 np: LD I0.0 MOVW VW0, VW4 -I VW2, VW4 (IN1) (IN2) czyli: VW0 - VW2 = VW4	Odejmowanie dwóch liczb całkowitych 16 bitowych ze znakiem (Signet Integer - IN1 i IN2) w zakresie: -32768 do +32767. Wynik jest zmienną 16 - bitową ze znakiem IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu będzie równy "**1**". /1,2,3 **Subtract Integer (-I):** IN2 - IN1 = IN2	
+D IN1, IN2 np: LD I0.1 +D VD4, VD8 /* (IN1) (IN2) czyli: VD4 + VD8 = VD8	Dodawanie dwóch liczb całkowitych 32 bitowych ze znakiem (Signet Integer - IN1 i IN2) w zakresie: -2147483648 do +2147483647. Wynik jest zmienną 32 - bitową ze znakiem IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu będzie równy "**1**". /1,2,3 **Add Double Integer (+D):** IN1 + IN2 = IN2	IN1: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K

-D IN1, IN2 np: LD I0.1 MOVD +25, VD8 -D +15, VD8 (IN1) (IN2) czyli: 25 - 15 = VD8	Odejmowanie dwóch liczb całkowitych 32 bitowych ze znakiem (Signet Integer - IN1 i IN2) w zakresie: -2147483648 do +2147483647. Wynik jest zmienną 32 - bitową ze znakiem IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu będzie równy "**1**". /1,2,3 **Subtract Double Integer (-D):** IN2 - IN1 = IN2	IN2:: VD, ID, QD, MD, (32 bity) SMD, AC
MUL IN1, IN2 np: LD I0.0 MUL VW100, QD0 /* (IN1) (IN2) czyli: VW100 * QD0 = QD0	Mnożenie 16 bitowej liczby całkowitej ze znakiem IN1 w zakresie: -32768 do +32767 z mniej znaczącym słowem 16 bitowym liczby 32 bitowej ze znakiem IN2. Wynik działania IN2 jest zmienną 32 bitowa. Operacja zostanie wykonana, gdy pierwszy bit stosu równa się "**1**". /2,3 **Multiply Integer (MUL):** IN1 * IN2 = IN2	IN1: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K IN2: VD, ID, QD, MD, (32 bity) SMD, AC <u>Uwaga</u>: *Część całkowita ilorazu prze- chowywana jest w pierwszych 16 mniej znaczących bitach, a reszta w kolejnych 16 bardziej znaczących bitach zmiennej wyjściowej 32 bitowej IN2.*
DIV IN1, IN2 np: LD I0.1 MOVW VW0, VW6 DIV VW2, VD4 (IN1) (IN2) czyli: VW0 : VW2 = VD4	Operacja dzielenia mniej znaczącego słowa 16 bitowego ze znakiem, w zakresie: -32768 do +32767, zmiennej 32-bitowej IN2 ze zmienną 16-bitową ze znakiem IN1. Wynik działania IN2 jest zmienną 32 bitowa. Operacja zostanie wykonana, gdy pierwszy bit stosu równa się "**1**". /1,2,3,4 **Divide Integer (DIV):** IN2 : IN1 = IN2	
SQRT IN, OUT np: LD I0.0 SQRT VD0, VD4 (IN1) (OUT)	Pierwiastek kwadratowy 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 pierwszy bit stosu równa się "**1**". /1,2, 3 **Square Root (SQRT)**	IN: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K OUT: VD, ID, QD, MD, (32 bity) SMD, AC
+R IN1, IN2 np: LD I0.0 +I VD10, VD60 /* (IN1) (IN2) czyli: VD10 + VD60 = VD60	Operacja dodawania dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i zajmuje miejsce IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu równa się "**1**". /1,2,3 **Add Real (+R):** IN1 + IN2 = IN2	IN1: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K IN2: VD, ID, QD, MD, (32 bity) SMD, AC

-R IN1, IN2 np: LD I0.1 MOVR 35.56700, VD8 -D 10.00000, VD8 (IN1) (IN2) czyli: 35,567 - 10,0 = VD8	Operacja odejmowania dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i zajmuje miejsce IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu równa się "**1**". /1,2,3 **Subtract Real (-R):** IN2 - IN1 = IN2	
*R IN1, IN2 np: LD I0.0 *R VD100, QD0 /* (IN1) (IN2) czyli: VD100 * QD0 = QD0	Operacja mnożenia dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i zajmuje miejsce IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu równa się "**1**". /1,2,3 **Multiply Real (*R):** IN1 * IN2 = IN2	
/R IN1, IN2 np: LD I0.0 /R VD100, VD0 /* (IN1) (IN2) czyli: VD0 / VD100 = VD0	Operacja dzielenia dwóch 32-bitowych liczb rzeczywistych IN1 oraz IN2 w arytmetyce zmiennoprzecinkowej (floating point). Wynik operacji jest liczbą rzeczywistą 32-bitową i zajmuje miejsce IN2. Operacja zostanie wykonana, gdy pierwszy bit stosu równa się "**1**". /1,2,3,4 **Divide Real (/R):** IN2 / IN1 = IN2	
INCW IN np: LD I0.5 MOVW VW0, VW5 INCW VW5 (IN) czyli: VW0 + 1 = VW5	Instrukcja powoduje zwiększenie wartości słowa wejściowego 16 bitowego (IN) ze znakiem o zakresie: -32768 do +32767 o "1", gdy pierwszy bit stosu równa się "**1**". /1,2,3 **Increment Word (INCW):** IN + 1 = IN	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC,

DECW IN np: LD I0.0 DECW VW100 /* (IN) czyli: VW100 - 1 = VW100	Instrukcja powoduje zmniejszenie wartości słowa wejściowego 16 bitowego (IN) ze znakiem o zakresie: -32768 do +32767 o "1", gdy pierwszy bit stosu równa się "**1**". /1,2,3 **Decrement Word (DECW):** IN - 1 = IN	
INCD IN np: LD I0.5 INCD VD5 /* (IN) czyli: VD5 + 1 = VD5	Instrukcja zwiększająca wartość słowa wejściowego 32 bitowego (IN) ze znakiem o zakresie: -2147483648 do +2147483647 o "1", gdy pierwszy bit stosu równa się "**1**". /1,2,3 **(Increment Double Word (INCD):** IN + 1 = IN	IN: VD, ID, QD, MD, (32 bity) SMD, AC
DECD IN np: LD I0.3 MOVD VD55, VD70 INCD VD70 (IN) czyli: VD55 - 1 = VD70	Instrukcja zmniejszająca wartości słowa wejściowego 32 bitowego (IN) ze znakiem o zakresie: -2147483648 do +2147483647 o "1", gdy pierwszy bit stosu równa się "**1**". /1,2,3 **Decrement Double Word (DECD):** IN - 1 = IN	

UWAGI:

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

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

/3 *W przypadku, gdy wynik operacji IN 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",*

/* *Przy takim sposobie zapisu funkcji, aby sprawdzić jej działanie korzystając z tablicy statusu sterownika, należy spowodować jednokrotne wykonanie danej operacji*
W tym celu należy zastosować dodatkowo styk impulsowy EU, który spowoduje ustawienie na czas jednego cyklu, zmiennej wejściowej I (pierwszego bitu stosu) rozpoczynającej wykonanie danej instrukcji.

Oto przykłady notacji wybranych bloków funkcyjnych uwzględniających powyższą uwagę:

```
LD    I0.0                LD    I0.5
EU                        EU
+I    VW50, VW60          INCD  VD5

LD    I0.1                LD    I0.0
EU                        EU
+D    VD4, VD8            DECW  VW100
```

```
LD      I0.0
EU
MUL     VW100, QD0
```

6. 8. Operacje logiczne

STL	Opis	Zmienna
ANDW IN1, IN2 np: LD I0.0 MOVW VW0, VW4 ANDW VW2, VW4 (IN1) (IN2)	Koniunkcja dwóch słów (IN1 i IN2) 16 bitowych bez znaku (zakres od 0 do 65535). Instrukcja zostaje wykonana, gdy pierwszy bit stosu wynosi "**1**". Wynik zajmuje miejsce liczby IN2. /1 **AND Word (ANDW)**	IN1: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K IN2: VW, T, C, IW, AC (słowo) QW, MW, SMW,
ORW IN1, IN2 np: LD I0.1 ORW VW10, QW0 /* (IN1) (IN2)	Suma logiczna dwóch słów (IN1 i IN2) 16 bitowych bez znaku (zakres od 0 do 65535). Instrukcja zostaje wykonana, gdy pierwszy bit stosu wynosi "**1**". Wynik zajmuje miejsce liczby IN2. /1 **OR Word (ORW)**	
XORW IN1, IN2 np: LD I0.2 XORW VW100, VW10 /* (IN1) (IN2)	Wyłącznie - LUB dwóch słów (IN1 i IN2) 16 bitowych bez znaku (zakres od 0 do 65535). Instrukcja zostaje wykonana, gdy pierwszy bit stosu wynosi "**1**". Wynik zajmuje miejsce liczby IN2. /1 **XOR Word (XORW)**	
INVW IN np: LD I0.4 INVW VW99 (IN)	Funkcja negacji (NOT) powoduje zmianę stanu każdego bitu słowa 16 bitowego ze znakiem na przeciwny. Operacja zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". Wynik zajmuje miejsce zmiennej IN. /1 **Invert Word (INVW)**	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC,

ANDD IN1, IN2 np: LD I0.0 MOVD VD0, VD8 ANDD VD4, VD8 (IN1) (IN2)	Koniunkcja dwóch słów (IN1 i IN2) 32 bitowych bez znaku (zakres od 0 do 4294967295). Instrukcja zostaje wykonana, gdy pierwszy bit stosu wynosi "**1**". Wynik zajmuje miejsce liczby IN2. /1 **AND Doble Word (ANDD)**	IN1: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K IN2: VD, ID, QD, MD, (32 bity) SMD, AC
ORD IN1, IN2 np: LD I0.0 MOVD 125, VD150 ORD 3570, VD150 (IN1) (IN2)	Suma logiczna dwóch słów (IN1 i IN2) 32 bitowych bez znaku (zakres od 0 do 4294967295). Instrukcja zostaje wykonana, gdy pierwszy bit stosu wynosi "**1**". Wynik zajmuje miejsce liczby IN2. /1 **OR Doble Word (ORD)**	
XORD IN1, IN2 np: LD I0.1 MOVD VD0, VD8 XORD VD4, VD8 (IN1) (IN2)	Wyłącznie - LUB dwóch słów (IN1 i IN2) 32 bitowych bez znaku (zakres od 0 do 4294967295). Instrukcja zostaje wykonana, gdy pierwszy bit stosu wynosi "**1**". Wynik zajmuje miejsce liczby IN2. /1 **XOR Double Word (XORD)**	
INVD IN np: LD I0.6 INVD VW199 (IN)	Funkcja negacji (NOT) powoduje zmianę stanu każdego bitu słowa 32 bitowego ze znakiem na przeciwny. Operacja zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". Wynik zajmuje miejsce zmiennej IN. /1 **Invert Double Word (INVD)**	IN: VD, ID, QD, MD, (32 bity) SMD, AC,

UWAGA: /1 *W przypadku, gdy wynik operacji IN2 wyniesie zero bit specjalny SM1.0 ustawia się na "1",*

/* *Przy takim sposobie zapisu funkcji, aby sprawdzić jej działanie korzystając z tablicy statusu sterownika, należy spowodować jednokrotne wykonanie danej operacji.*
W tym celu należy zastosować dodatkowo styk impulsowy EU, który spowoduje ustawienie na czas jednego cyklu, zmiennej wejściowej I (pierwszego bitu stosu) rozpoczynającej wykonanie danej instrukcji.

6. 9. Funkcje przemieszczenia

STL	Opis	Zmienna

MOVB IN, OUT
 	
np:
LD I0.0
MOVB VB5, VB15
(IN) (OUT)
	Instrukcja przemieszczająca zmienną 8 bitową (IN) bez znaku o zakresie do 255 w miejsce docelowe określone adresem "OUT". Bajt wejściowy nie ulega zmianie po wykonaniu przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy pierwszym bitem stosu jest "**1**".
	Move Byte (MOVB)
		IN: VB, IB, QB, MB, (bajt) SMB, AC,' K
		
		OUT: VB, IB, QB, MB, (bajt) SMB, AC

MOVW IN, OUT
 	
np:
LD I0.1
MOVW VW300, VW45
(IN) (OUT)
	Instrukcja przemieszczająca zmienną 16 bitową (IN) bez znaku o zakresie do 65535 w miejsce docelowe określone adresem "OUT". Słowo wejściowe nie ulega zmianie po wykonaniu przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy pierwszym bitem stosu jest "**1**".
	Move Word (MOVW)
		IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K
		OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AQW

MOVD IN, OUT
 	
np:
LD I0.4
MOVD VD5, VD500
(IN) (OUT)
	Instrukcja przemieszczająca zmienną 32 bitową (IN) bez znaku
	o zakresie do 4294967295 w miejsce docelowe określone adresem "OUT". Zmienna wejściowa nie ulega zmianie po wykonaniu przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy pierwszym bitem stosu jest "**1**".
	Move Double Word (MOVD)
		IN: VD, ID, QD, MD, (32 bity) SMD, AC, HC, K
		
		OUT: VD, ID, QD, MD, (32 bity) SMD, AC

MOVR IN, OUT
 	
np:
LD I0.4
MOVR 21.14527, VD500
(IN) (OUT)
	Instrukcja przemieszczająca zmienną rzeczywistą 32 bitową (IN)
	w miejsce docelowe określone adresem "OUT". Zmienna wejściowa (IN) nie ulega zmianie po wykonaniu przemieszczenia. Operacja przemieszczenia zostanie wykonana, gdy pierwszym bitem stosu jest "**1**".
	Move Real (MOVR)	

SWAP IN np: LD I0.3 SWAP VW450 /* (IN)	Blok funkcyjny powodujący zamianę miejscami dwóch bajtów (zmiennych 8 bitowych) słowa 16 bitowego - mniej (LSB) z bardziej (MSB) znaczącym. Operacja zamiany zostanie wykonana, gdy pierwszym bitem stosu jest "**1**". **Swap (SWAP)**	IN: VW, T, C, IW, AC, (słowo) QW, MW, SMW,
SHRB DATA, S_BIT, N np: LD I0.2 SHRB I0.1, Q0.3, 2 /* (DATA) (S_BIT) (+N)	Rejestr szeregowy wpisujący o długości określonej parametrem "N" (max 64). Kierunek wpisywania danych "DATA" określa znak przy parametrze "N" (+N - wpisywanie danych z prawej strony w kierunku bitów o wagach rosnących lub -N - wpisywanie z lewej strony w kierunku bitów o wagach malejących). Parametr "S_BIT" określa pierwszy (w przypadku +N) lub ostatni (w przypadku -N) bit rejestru szeregowego. Operacja wpisywania zostanie wykonana, gdy pierwszym bitem stosu jest "**1**". /[1] **(Shift Register Bit)**	DATA, S_BIT: I, Q, M, SM, T, C, W, V N: VB, IB, QB, MB, SMB, AC, K

UWAGA: */** *W tym przypadku, aby sprawdzić działanie bloku funkcyjnego*

korzystając z tablicy statusu sterownika, należy wymusić jednokrotne wykonanie danej operacji. W tym celu należy zastosować dodatkowo styk impulsowy EU, który spowoduje ustawienie na czas jednego cyklu, zmiennej wejściowej I (pierwszego bitu stosu) rozpoczynającej wykonanie danej operacji, np:

LD I0.3
EU
SHRB I0.1, Q0.3, 2

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

6. 10. Funkcje na blokach danych

STL	Opis	Zmienna
BMB IN, OUT, N np: LD I0.3 BMB VB200, VB100, 3 (IN) (OUT) (N)	Instrukcja powodująca przemieszczenie określonej liczby bajtów (zmiennych 8-bitowych) określonej parametrem "N" 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). Instrukcja zostanie wykonana, gdy pierwszym bitem stosu jest "**1**". **Block Move Byte (BMB)**	IN: VB, MB, SMB, (bajt) IB, QB, OUT: VB, IB, QB, MB, (bajt) SMB, N: VB, IB, QB, MB, (bajt) SMB, AC, K
BMW IN, OUT, N np: LD I0.2 BMW VW200, VW100, 9 (IN) (OUT) (N)	Instrukcja powodująca przemieszczenie pewnej liczby słów (zmiennych 16-bitowych) określonej parametrem "N" z przestrzeni o adresie początkowym "IN" do miejsca o adresie początkowym zdefiniowanym przez parametr "OUT". Ilość przemieszczanych słów "N" zawiera się w przedziale: (1 - 255). Instrukcja zostanie wykonana, gdy pierwszym bitem stosu jest "**1**". **Block Move Word (BMW)**	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC, OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AQW N: VB, IB, QB, MB, (bajt) SMB, AC, K
FILL IN, OUT, N	Instrukcja wypełniająca przestrzeń adresową począwszy od adresu początkowego "OUT" słowami	IN: VW, T, C, IW, (słowo) QW, MW,

np: LD I0.4 FILL VW4, VW6, VB10 (IN) (OUT) (N)	(zmiennymi 16-bitowymi) o wartości określonej parametrem "IN". Ilość przemieszczanych słów "N" zawiera się w przedziale: (1 - 255). Instrukcja zostanie wykonana, gdy pierwszym bitem stosu jest "**1**". **Memory Fill (FILL_N)**	SMW, AIW, K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AQW N: VB, IB, QB, MB, (bajt) SMB, AC, K

6. 11. Funkcje związane z tablicami danych

STL	Opis	Zmienna
ATT DATA, TABLE np: LD I0.0 ATT VW50, VW100/* (DATA) (TABLE)	Instrukcja umożliwiająca wpisywanie słów (danych 16 bitowych) - DATA do tablicy o adresie początkowym (TABLE). Pod tym adresem umieszczone jest słowo określające zadaną pojemność tablicy (od 0 do 99 słów). Kolejny adres zawiera zmienną stanowiącą licznik wejściowy (EC). Zawiera on informację o liczbie umieszczonych słów w tablicy. Zawartość licznika (EC) wzrasta o jeden po każdym kolejnym wpisaniu słowa. Następny adres jest adresem początkowym wskazującym umiejscowienie pierwszego z wpisywanych kolejno słów. Operacja wpisywania jest powtarzana w kolejnych cyklach za każdym razem, gdy pierwszym bitem stosu będzie "**1**". /1 **Add To Table (ATT)**	DATA: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K TABLE: VW, T, C, IW, (słowo) QW, MW, SMW

LIFO TABLE, DATA np: LD I0.0 LIFO VW100, VW300/ (TABLE) (DATA)	Instrukcja pozwalająca na wyprowadzanie słów (zmiennych 16 bitowych) w miejsce wyspecyfikowane parametrem (DATA) z tablicy o adresie początkowym (TABLE) w kolejności odwrotnej w stosunku do operacji wprowadzania. Zawartość licznika wej-ściowego (EC) maleje o jeden po każdym kolejnym wyprowadzeniu zmiennej z tablicy. Operacja wyprowadzania jest powtarzana w kolejnych cyklach za każdym razem, gdy pierwszym bitem stosu będzie "**1**". /2 **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
FIFO TABLE, DATA np: LD I0.0 FIFO VW100, VW300/* (TABLE) (DATA)	Instrukcja pozwalająca na wyprowadzanie danych 16 bitowych w miejsce wyspecyfikowane parametrem (DATA) z tablicy o adresie początkowym (TABLE) w kolejności zgodnej w stosunku do operacji wprowadzania. Zawartość licznika wejściowego (EC) maleje o jeden po każdym kolejnym wyprowadzeniu zmiennej z tablicy. Operacja wyprowadzania jest powtarzana w kolejnych cyklach za każdym razem, gdy pierwszym bitem stosu będzie "**1**". /2 **First-In-First-Out (FIFO)**	
FND= SRC, PATRN, INDX np: LD I0.1 FND= VW102, 16#2500, VW0 FND<> SRC, PATRN, INDX FND> SRC, PATRN, INDX FND< SRC, PATRN, INDX	Instrukcje pozwalają 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 (=, <>, <, 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 (EC). **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

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.*

Przed operacją wpisania ATT

VW300	2345	
VW100	0004	TABLE - pojemność tablicy
VW102	0002	EC - licznik wejściowy
VW104	1245	dana - 0
VW106	8945	dana - 1
VW108	xxxx	brak danej
VW110	xxxx	brak danej

Po operacji wpisania ATT

VW100	0004	TABLE - pojemność tablicy
VW102	0003	EC - licznik wejściowy
VW104	1245	dana - 0
VW106	8945	dana - 1
>VW108	2345	dana - 2
VW110	xxxx	brak danej

Przed operacją LIFO

VW100	0004	TABLE - pojemność tablicy
VW102	0003	EC - licznik wejściowy
VW104	1245	dana - 0
VW106	8945	dana - 1
VW108	2345	dana - 2
VW110	xxxx	brak danej

Po operacji LIFO

>VW300	2345	
VW100	0004	TABLE - pojemność tablicy
VW102	0002	EC - licznik wejściowy
VW104	1245	dana - 0
VW106	8945	dana - 1
VW108	xxxx	brak danej
VW110	xxxx	brak danej

Przed operacją FIFO

VW100	0004	TABLE - pojemność tablicy
VW102	0002	EC - licznik wejściowy
VW104	1245	dana - 0
VW106	8945	dana - 1
VW108	xxxx	brak danej
VW110	xxxx	brak danej

Po operacji FIFO

>VW300	1245	
VW100	0004	TABLE - pojemność tablicy
VW102	0001	EC - licznik wejściowy
VW104	8945	dana - 0
VW106	xxxx	brak danej
VW108	xxxx	brak danej
VW110	xxxx	brak danej

W przypadku przeszukiwania tablicy zmienna VW100 określająca jej początek nie jest wymagana. Przy pierwszym przeszukiwaniu elementu tablicy równego 2500 wartość parametru INDX wyniesie 1, a po zwiększeniu tego parametru o jeden i ponownym przeszukaniu tablicy wartość indeksu INDX będzie równa 3.

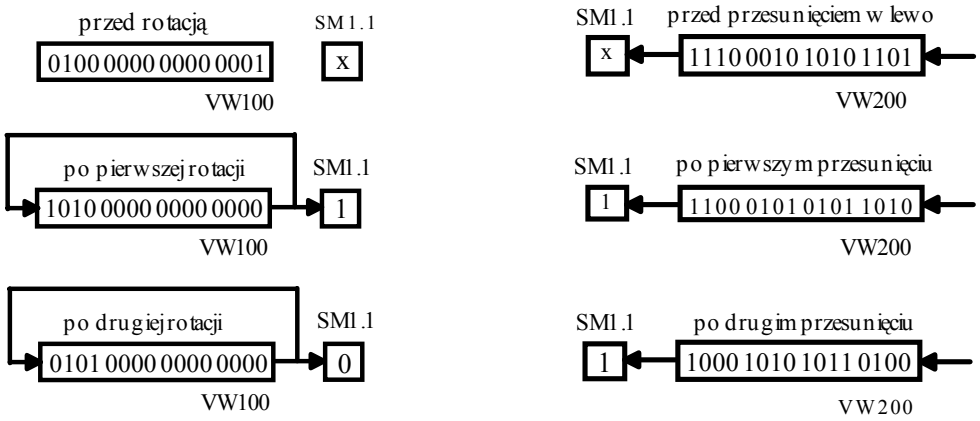
VW102	0004	EC - licznik wejściowy
VW104	1245	dana - indeks 0
VW106	**2500**	dana - indeks 1
VW108	9976	dana - indeks 2
VW110	**2500**	dana - indeks 3

6. 12. Funkcje przesunięcia

Przedstawione rysunki ilustrują realizację przykładowych instrukcji przesunięcia (Shift), które zostały opisane w poniższej tabeli:

Rotacja zmiennej 16-bitowej bitowej w prawo (RRW)

Przesunięcie zmiennej 16- o jedną pozycję (N=1): w lewo (SLW) o jedną pozycję (N=1):



STL	Opis	Zmienna
SRW IN, N np: LD I0.0 SRW VW500, VB1 /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (16 bitowego) bez znaku 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 "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /[1,2 ,3] **Shift Right Word (SRW)**	IN: VW, T, C, IW, (słowo) QW, MW, SMW, N: VB, IB, QB, MB, (bajt) SMB, AC, K
SLW IN, N np: LD I0.0 SLW VW200, 1 /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (16 bitowego) bez znaku w lewo w kierunku bitów o rosnących wagach, o wyszczególnioną parametrem "N" liczbę miejsc. W puste miejsca zostają wpisane zera. Operacja, której wynik jest przypisany do "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /[1,2,3] **Shift Left Word (SLW)**	
RRW IN, N np: LD I0.0 RRW VW100, 1 /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (16 bitowego) bez znaku, 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 "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /[1,2,3] **Rotate Right Word (RRW)**	
RLW IN, N np: LD I0.0 RLW VW200, 4 /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (16 bitowego) bez znaku, 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 "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /[1,2,3] **Rotate Left Word (RLW)**	
SRD IN, N np: LD I0.4 SRD VD250, 1 /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (32 bitowego) bez znaku 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 "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /[1,2,3] **Shift Right Double Word (SRD)**	IN: VD,ID,QD,MD, (32bity) SMD,AC

SLD IN, N np: LD I0.5 SLD VW220, 2 /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (32 bitowego) bez znaku w lewo w kierunku bitów o rosnących wagach, o wyszczególnioną parametrem "N" liczbę miejsc. W puste miejsca zostają wpisane zera. Operacja, której wynik jest przypisany do "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /1,2,3 **Shift Left Double Word (SLD)**	N: VB, IB, QB, K, (bajt) MB, SMB, AC,
RRD IN, N np: LD I0.6 RRD VW225, VB /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (32 bitowego) bez znaku, 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 "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /1,2,3 **Rotate Right Double Word (RRD)**	
RLD IN, N Np: LD I0.0 RLD VW200, 1 /* (IN) (N)	Instrukcja przesuwająca wszystkie bity słowa "IN" (32 bitowego) bez znaku, 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 "IN", zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /1,2,3 **Rotate Left Double Word (RLD)**	

UWAGI:

/1 *Bit specjalny SM1.0 ustawia się na "1", gdy wynik danej operacji wynosi IN ,*

/2 *Bit specjalny SM1.1 zawiera ostatni bit podlegający operacji przesunięcia lub rotacji,*

/3 *Stopień przesunięcia/rotacji określony parametrem (N) zawiera się w przedziale: (0 - 15) dla rejestru 16-bitowego (W) oraz (0 - 31) dla rejestru 32-bitowego (D). W przypadku przekroczenia przez ten parametr wartości maksymalnej przesunięcie wynosić będzie odpowiednio: N = 16 (dla słowa - W) lub N=32 (dla podwójnego słowa - D),*

/* *Aby sprawdzić działanie powyższych funkcji korzystając z tablicy statusu sterownika, należy zastosować dodatkowo styk impulsowy EU, który spowoduje ustawienie na czas jednego cyklu, zmiennej wejściowej I (pierwszego bitu stosu) inicjującej daną operację.*

6. 13. Funkcje konwersji

LAD	Opis	Zmienna
BCDI IN np: LD I0.0 BCDI VW250 (IN)	Konwersja danych "IN" zapisanych w kodzie BCD-4 na dane zapisane jako liczby całkowite (Integer). Operacja, której wynik jest przypisany do "IN" zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /1 **BCD to Integer (BCDI)**	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC
IBCD IN np: LD I0.5 IBCD VW250 (IN)	Konwersja danych "IN" zapisanych jako liczba całkowita (max. 9999) na równoważną liczbę zapisaną w kodzie BCD. Operacja, której wynik jest przypisany do "IN" zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". /1 **Integer to BCD (IBCD)**	
DECO IN, OUT np: LD I0.3 DECO VB250, QW0 (IN) (OUT)	Instrukcja ustawiająca na "1" 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 wykonana, gdy pierwszym bitem stosu jest "**1**". **Decode (DECO)**	IN: VB, IB, QB, MB, (bajt) SMB, AC, K OUT: VW, T, C, IW, (słowo) QW, MW, SMW, AC,
ENCO IN, OUT np: LD I0.3 ENCO VW250, VB100 (IN) (OUT)	Instrukcja lokalizująca w słowie 16 bitowym wejściowym "IN" pierwszy bit o wartości "1". Operacja zostaje wykonana, gdy pierwszym bitem stosu jest "**1**". Lokalizowanie bitu rozpoczyna się od najniższego bitu słowa wejściowego "IN". Numer pozycji zlokalizowanego bitu, w zapisie dwójkowym, wyprowadzony jest na wyjście "OUT". **Encode (ENCO)**	IN: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K OUT: VB, IB, QB, MB, (bajt) SMB, AC
SEG IN, OUT np: LD I0.4 SEG +5, QB0 (IN) (OUT)	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: liczbie 16 odpowiada 0, 17 - 1, ... , 31-15 itd. Operacja konwersji zachodzi, gdy pierwszym bitem stosu jest "**1**". **Segment (SEG)**	IN: VB, IB, QB, MB, (bajt) SMB, AC, K OUT: VB, IB, QB, MB, (bajt) SMB, AC
HTA IN, OUT, LEN np: LD I0.4 HTA VB45, VB100, 30 (IN) (OUT) (LEN)	Przetwarzanie ciągu liczb zapisanych w formacie heksadecymalnym o długości "LEN"" (1 do 255) rozpoczynających się od adresu początkowego "IN" na ciąg znaków ASCII o adresie początkowym określonym parametrem "OUT". W przypadku LEN=0 proces konwersji nie jest realizowany. Operacja konwersji zachodzi, gdy pierwszym bitem stosu jest "**1**". **HEX to ASCII (HTA)**	IN: VB, IB, QB, MB, (bajt) SMB,

ATH IN, OUT, LEN np: LD I0.4 ATH VB50, VB100, 30 (IN) (OUT) (LEN)	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. Operacja konwersji zachodzi, gdy pierwszym bitem stosu jest "**1**". /3 **ASCII to HEX (ATH)** 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)	OUT: VB, IB, QB, MB, (bajt) SMB, LEN: VB, IB, QB, MB, (bajt) SMB, AC, K
DTR IN, OUT np: LD I0.5 DTR VD5, VD100, (IN) (OUT)	Instrukcja przetwarzająca zmienną 32-bitową ze znakiem (signet integer) na 32-bitową liczbę rzeczywistą o adresie "OUT". Operacja konwersji zachodzi, gdy pierwszym bitem stosu jest "**1**". **Double Word Integer to Real (DTR)**	IN: VD, ID, QD, MD, (bajt) SMD, AC, HC, K OUT: VD, ID, QD, MD, (bajt) SMD, AC
TRUNC IN, OUT np: LD I0.5 TRUNC VD60, VD90, (IN) (OUT)	Instrukcja przetwarzająca 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)**	

UWAGA: /1 *W przypadku, gdy do wejścia "IN" bloku IBCD lub BCDI zostanie doprowadzona zmienna, której wartość jest nieprawidłowa, bit specjalny SM1.6 ustawia się na "1".*

/2 *Gdy wynik operacji przekroczy wartość dopuszczalną bit specjalny SM1.1 ustawi się na "1".*

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

6. 14. Funkcje związane z zegarem czasu rzeczywistego

W punkcie tym zostaną opisane funkcje odczytu i ustawiania zegara wewnętrznego, w który wyposażony jest sterownik z jednostką CPU 214. Przy ustawianiu daty należy pamiętać o tym, że sterownik nie weryfikuje prawidłowości wpisywanych danych i na przykład zapis: 30 lutego będzie akceptowany. Wszystkie wartości odnoszące się do czasu i daty należy przedstawiać w formacie BCD (na przykład: 16#95 odnosi się do roku 1995). Początkowe ustawienia zegara są następujące:

- Data: 01 - stycznia - 90
- Czas 00 : 00 : 00
- Dzień tygodnia Niedziela

STL	Opis	Zmienna
TODR T np: LD I0.0 TODR VB40	Instrukcja umożliwiająca 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. **Time of Day Read (TODR)**	T: VB, IB, QB, MB, (bajt) SMB
TODW T np: LD I0.0 TODW VB40	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. **Time of Day Write (TODW)**	

bajt początkowy - pierwszy	VB400	99	Rok	od 0 do 99
bajt drugi	VB401	01	Miesiąc	od 1 do 12
bajt trzeci	VB402	15	Dzień	od 1 do 30
bajt czwarty	VB403	21	Godzina	od 0 do 23
bajt piąty	VB404	25	Minuty	od 0 do 59
bajt szósty	VB405	39	Sekundy	od 0 do 59
bajt siódmy	VB406	00		
bajt ósmy	VB407	06	Dzień tygodnia	od 1 do 7 (1 = niedziela)
Dane zawarte w buforze zegarowym wskazują na: 15 stycznia 1999r., godzina: 21:25:39 - piątek				

Przykład bufora zegarowego o adresie początkowym T = VB400

6. 15. Funkcje związane ze strukturą programu *(funkcje sterujące)*

STL	Opis	Zmienna
JMP n	Instrukcja skoku warunkowego do podprogramu o adresie "n". Powoduje ona pominięcie części programu sterującego umieszczonego pomiędzy instrukcją "JUMP n", a etykietą "LABEL n". Instrukcja zostanie wykonana, gdy pierwszy bit stosu wynosi "**1**". **Jump to Label (JMP)**	n = (0 do 63) dla CPU 212 n = (0 do 255) dla CPU 214

LBL n	Etykieta LABEL oznacza miejsce docelowe "n" skoku wywołanego przez funkcję "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
CALL n	Instrukcja powodująca przeniesienie wykonywania programu sterującego do podprogramu o etykiecie "n". Po wykonaniu podprogramu następuje powrót do miejsca bezpośrednio występującego za instrukcją "CALL n". Instrukcja zostanie wykonana, gdy pierwszy bit stosu wynosi "**1**". **CALL**	n = (0 do 15) dla CPU 212 n = (0 do 63) dla CPU 214
SBR n	Instrukcja wskazująca początek podprogramu "n" wywołanego procedurą "CALL n". Wszystkie podprogramy muszą być umieszczone na końcu głównego programu sterującego, tzn. po instrukcji MEND. **Subroutine (SBR)**	n = (0 do 15) dla CPU 212 n = (0 do 63) dla CPU 214
Uwaga:	*Instrukcje CALL/SBR umożliwiają nakładanie się na siebie zakresów działania w ten sposób, że zakres działania jednej pary instrukcji CALL/SBR zawiera się wewnątrz innej pary instrukcji. Pokrywać się może do 9 takich obszarów.*	
RET	Instrukcja bezwarunkowego powrotu z podprogramu umieszczana na końcu danego podprogramu. Kończy wykonywanie danego podprogramu. Musi znajdować się na końcu każdego podprogramu. **Return from Subroutine (RET)**	bez oznaczeń
CRET	Instrukcja warunkowego powrotu z podprogramu. Instrukcja może być wykorzystana do opuszczania danego podprogramu. Może również wystąpić przed instrukcją zakończenia podprogramu "RET". Instrukcja zostanie wykonana, gdy pierwszy bit stosu wynosi "**1**". **Conditional Return from Subroutine (CRET)**	bez oznaczeń
END	Warunkowe zakończenie wykonywania części logicznej programu. Zastosowanie tej instrukcji spowoduje zatrzymanie wykonywania programu w miejscu jej występowania i rozpoczęcie cyklu skanowania od początku. Instrukcja zostanie wykonana, gdy pierwszy bit stosu wynosi "**1**". **END**	bez oznaczeń
MEND	Instrukcja bezwarunkowa jako ostatnia w programie sterującym kończy wykonywanie programu i cały cykl pracy sterownika zaczyna się od początku. **Main program End (MEND)**	bez oznaczeń
STOP	Instrukcja warunkowa Stop kończy wykonywanie programu i powoduje natychmiastowe przejście sterownika w tryb STOP. Instrukcja zostanie wykonana, gdy pierwszy bit stosu wynosi "**1**". **STOP**	bez oznaczeń
FOR INDEX, INITIAL, FINAL	Instrukcja powoduje wielokrotne powtórzenie fragmentu programu sterującego zawartego między instrukcjami "FOR" i "NEXT". Ilość tych powtórzeń określają parametry - początkowy "INITIAL" oraz	INDEX: VW, T, C, IW, (słowo) QW, MW, SMW, AC

np: LD I0.0 FOR VW250, 1, 45 (IND.) (INIT.) (FIN.)	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 zostanie wykonana, gdy pierwszy bit stosu wynosi "**1**". **FOR**	INITIAL: VW, T, C, IW, (słowo) QW, MW, SMW, AC, AIW, K FINAL: VW, T, C, IW, QW, (słowo) MW, SMW, AC, AIW, K
NEXT	Instrukcja bezwarunkowa określa koniec obszaru programu sterującego, który podlega wielokrotnemu "zapętleniu". **NEXT**	
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.*		