Ćwiczenie 2

Zasady programowania sterownika PLC Modicon Micro. Podstawowe instrukcje języka drabinkowego Modsoft

1. Cel ćwiczenia

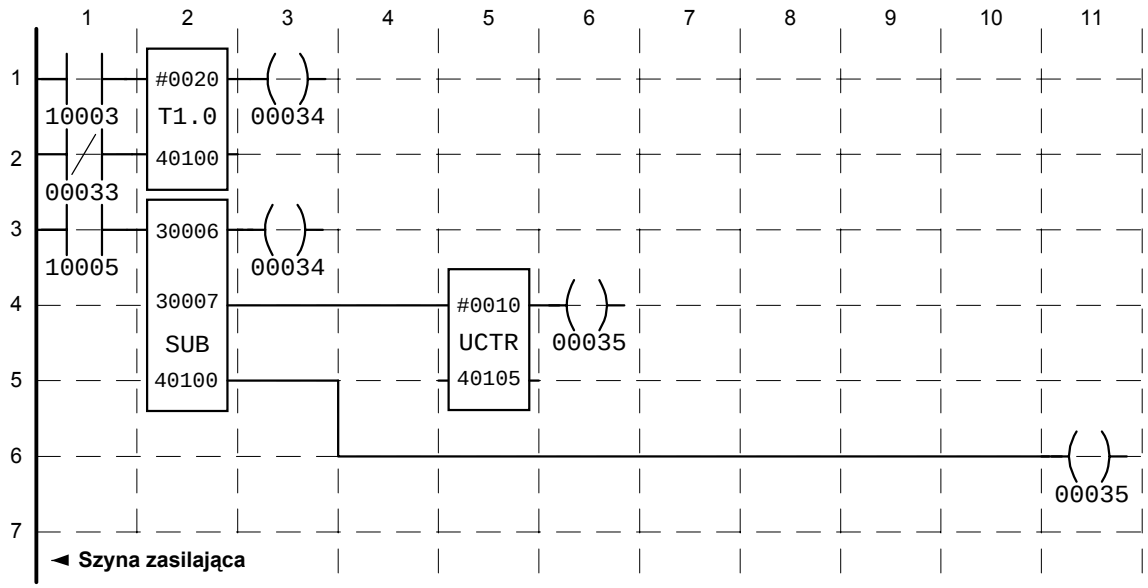
Celem ćwiczenia jest poznanie podstawowych zasad programowania i instrukcji języka drabinkowego Modsoft dla sterownika Modicon Micro oraz napisanie i uruchomienie własnych programów ilustrujących działanie instrukcji.

2. Wprowadzenie

2.1. Zasady wykonywania programu drabinkowego w sterowniku

Programowanie sterownika Modicon Micro 612xx w środowisku uruchomieniowym Modsoft odbywa się przy pomocy języka drabinkowego wspomaganego blokami funkcyjnymi. Podstawowe instrukcje "stykowe" pochodzą z elektrycznych schematów przekaźnikowych i służą do łatwego zapisania prostych operacji logicznych określających warunki zasilania cewek oraz stany wejść logicznych bloków funkcyjnych. Bloki funkcyjne są procedurami o większym stopniu złożoności i służą do przetwarzania danych, np. realizowania działań arytmetycznych, algorytmu PID, operacji na blokach danych, odmierzania czasu czy zliczania.

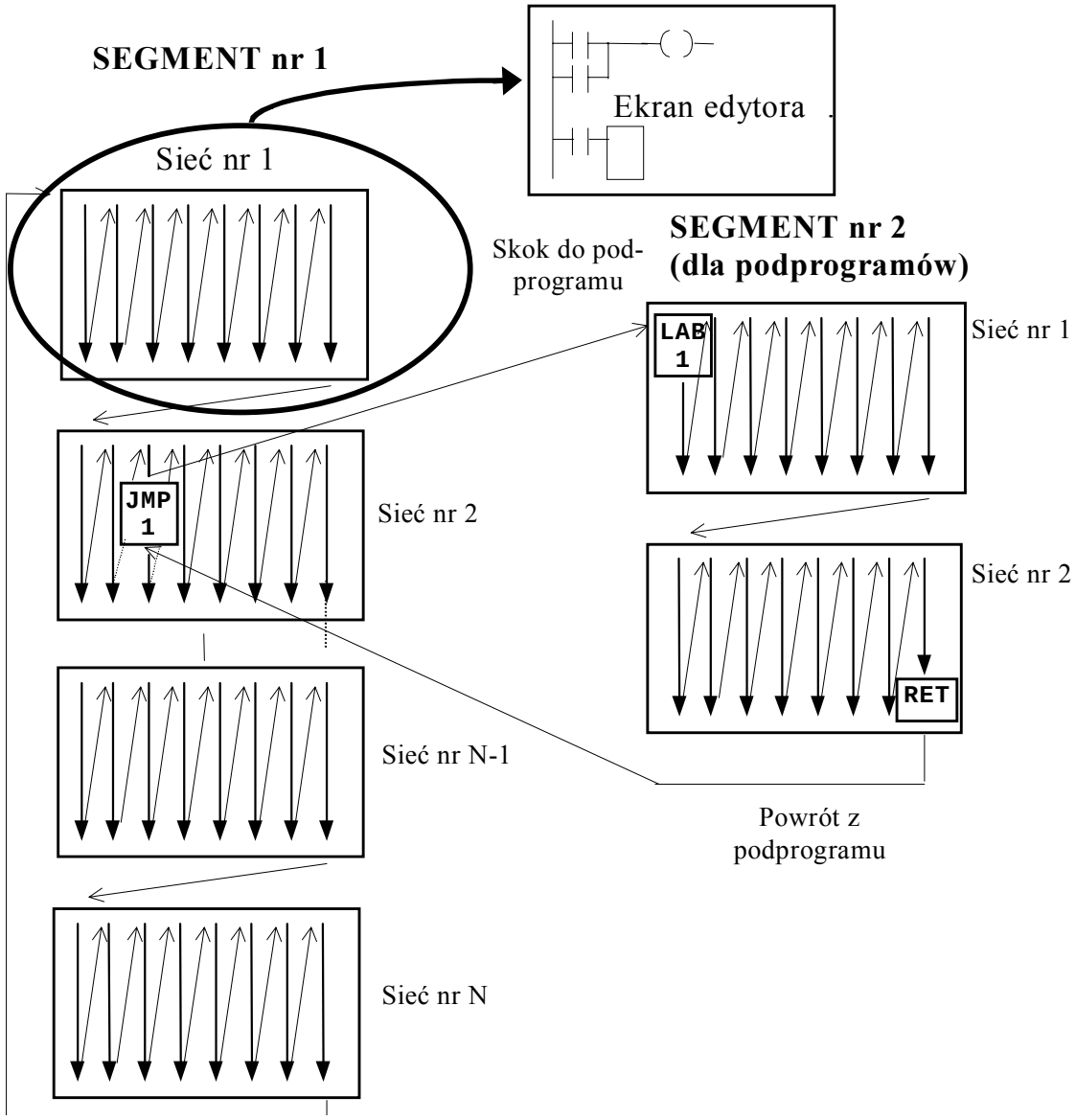
Program sterownika jest podzielony na 2 segmenty. Pierwszy to segment dla programu głównego, drugi - dla podprogramów, o ile takie występują. Każdy segment składa się z tzw. sieci – obszarów drabinki, w których zapisuje się instrukcje (jest to w praktyce fragment programu widoczny na ekranie w oknie edycji).



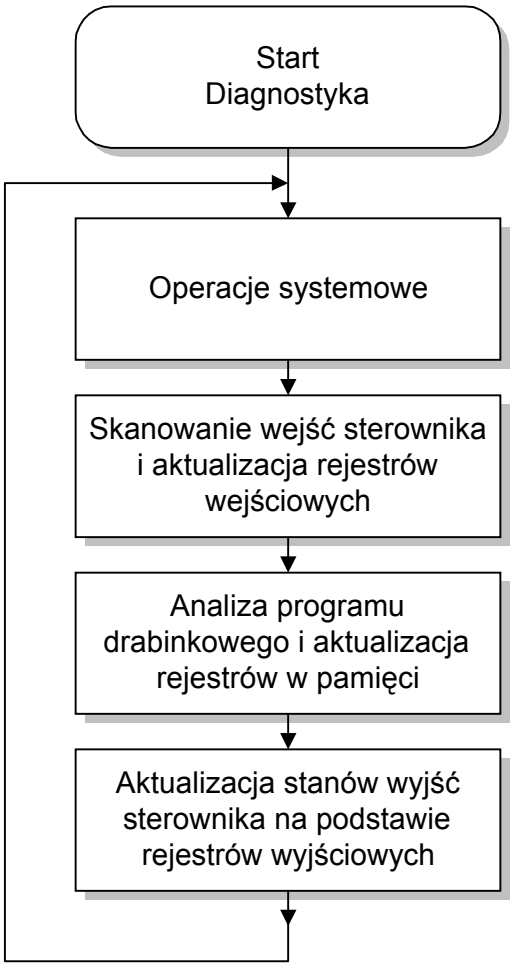
Rys. 1. Struktura pojedynczej sieci drabinki
Punkty przecięcia linii szczebli i kolumn stanowią węzły sieci drabinki.
Instrukcje w sieci wykonywane są w kolejności: 1-1, 1-2-4-5.

Liczba sieci w segmencie zależy od wielkości programu, ponieważ sieć ma ograniczoną pojemność (7 szczebli wysokości i 11 kolumn szerokości, przy czym w kolumnie 11 mogą znajdować się tylko cewki, Rys. 1). Niektóre instrukcje zajmują więcej niż jeden węzeł sieci, bo mają wysokość 2 lub 3 szczebli, zależnie od liczby wejść. Jeżeli w danej sieci brakuje miejsca należy dodać następną i kontynuować program. Rys. 1 pokazuje strukturę i kolejność wykonywania instrukcji w sieci.

Wykonywanie programu przez sterownik odbywa się w sposób sekwencyjny: instrukcja po instrukcji w obrębie sieci oraz sieć po sieci (Rys. 2). Przetwarzanie całego programu jest cykliczne. Każdy cykl jest poprzedzony odczytaniem stanów wejść sterownika i skopiowaniem ich do pamięci RAM danych. Na tych danych wejściowych operuje program, który według zaprogramowanego algorytmu aktualizuje odpowiednie rejestry i flagi w pamięci. Po dojściu do końca drabinki na podstawie zawartości odpowiadających im zmiennych wyjściowych uaktualniane są stany fizycznych wyjść sterownika. Cały proces nazywa się **cyklem skanowania** (Rys. 3) i obejmuje również zaprogramowane na stałe operacje systemowe. Wykonywanie programu przez sterownik odbywa się w sposób sekwencyjny: instrukcja po instrukcji w obrębie sieci oraz sieć po sieci (Rys. 2). Przetwarzanie całego programu jest cykliczne. Każdy cykl jest poprzedzony odczytaniem stanów wejść sterownika i skopiowaniem ich do pamięci RAM danych. Na tych danych wejściowych operuje program, który według zaprogramowanego algorytmu aktualizuje odpowiednie rejestry i flagi w pamięci. Po dojściu do końca drabinki na podstawie zawartości odpowiadających im zmiennych wyjściowych uaktualniane są stany fizycznych wyjść sterownika. Cały proces nazywa się **cyklem skanowania** (Rys. 3) i obejmuje również zaprogramowane na stałe operacje systemowe. Czas cyklu skanowania jest powiązany z liczbą sieci i zawartych w nich instrukcji. Dla sterowników serii 110 CPU 612xx może zawierać się od 10 do maksimum 250ms. Jeśli w takim czasie cykl skanowania nie zostanie zakończony, tzw. „*watchdog timer*” w CPU zatrzymuje program użytkownika i generuje sygnał o błędzie. Zapobiega to niekontrolowanemu zapętlaniu się programu sterownika. Istnieje również możliwość zadania stałego okresu cyklu. Moc obliczeniową sterownika określa parametr będący czasem obliczania samej logiki dla 1K węzłów drabinki. Jego wartość dla sterownika 110CPU612 wynosi 2.5ms.



Rys. 2. Przebieg procesu skanowania sieci programu sterownika
Strzałki przedstawiają kolejność skanowania instrukcji w sieci.



Rys. 3. Przebieg cyklu skanowania w sterowniku

2.2. Stałe i zmienne w programie

Sterownik wykorzystuje dwa typy zmiennych:

- bitowe (BIN)
- całkowite bez znaku (UW - *UNSIGNED WORD*)

Wartości zmiennych umieszczane są w rejestrach sterownika, a każdy z typów adresów odwołań dotyczy ograniczonej liczby rejestrów (Tab. 1). Liczby całkowite bez znaku zapisywane są w postaci UW, tzn. 16 bitów i mogą być traktowane również jako zestaw 16 bitów (jeden rejestr 16-bitowy może zawierać np. stany 16 wejść binarnych sterownika). Dostęp do pojedynczych bitów takiej liczby zapewniają specjalne instrukcje blokowe (patrz Tab. 8). Odwołanie do zmiennej (rejestru) każdego typu następuje przez adres składający się z 5 cyfr. Pierwsza z nich określa charakter zmiennej, a pozostałe jednoznacznie przypisany jej adres w pamięci (Tab. 1).

Tab. 1. Adresowanie rejestrów sterownika

Adres	Maksymalna liczba zmiennych	Typ	Komentarz
`0xxxx`	1536	BIN	wyjścia binarne i flagi bitowe
`1xxxx`	512	BIN	wejścia binarne
`3xxxx`	48	UW	wejścia analogowe i specjalne
`4xxxx`	1872	UW	wyjścia analogowe i ogólnego przeznaczenia

Część adresów jest przeznaczona do obsługi fizycznych wejść i wyjść sterownika oraz do funkcji specjalnych (Tab. 2). Wykorzystanie ich do innych celów niż przeznaczone nie powoduje błędu, ale może spowodować niezamierzone reakcje sterownika.

Tab. 2. Zakresy adresów dla wejść/wyjść sterownika

Zastosowanie	Zakres adresów dla wejść	Zakres adresów dla wyjść
Binarne wejścia i wyjścia	`10001 – 10016`	`00001 – 00016`
Przerwanie / Licznik sprzętowy	`10081 – 10088`	–
Timer / Licznik sprzętowy	`30001`	–
Analogowe wejścia i wyjścia	`30006 – 30010`	`40001 – 40002`

Sterowniki serii Modicon Micro 612xx są wyposażone w 4 wejścia analogowe i 2 wyjścia analogowe. Napięcie na przetwornikach jest reprezentowane w programie przez wartość w odpowiednim 16-bitowym rejestrze pamięci sterownika (Tab. 2). Zakresowi napięcia 0-10V odpowiada zakres liczb całkowitych 0-4095 ($=2^{12}$-1), np.: podanie napięcia 5 V na wejście analogowe nr 2 spowoduje pojawienie się w rejestrze o adresie `30007` wartości 2047, a zapisanie do rejestru o adresie `40001` wartości 1024 spowoduje pojawienie się na wyjściu analogowym nr 1 napięcia 2,5V.

Stałe w programie powinny zawierać się w zakresie 0-9999, a wpisywane są poprzez poprzedzenie liczby określającej wartość stałej znakiem „#”.

2.3. Instrukcje języka drabinkowego

A. Instrukcje stykowe

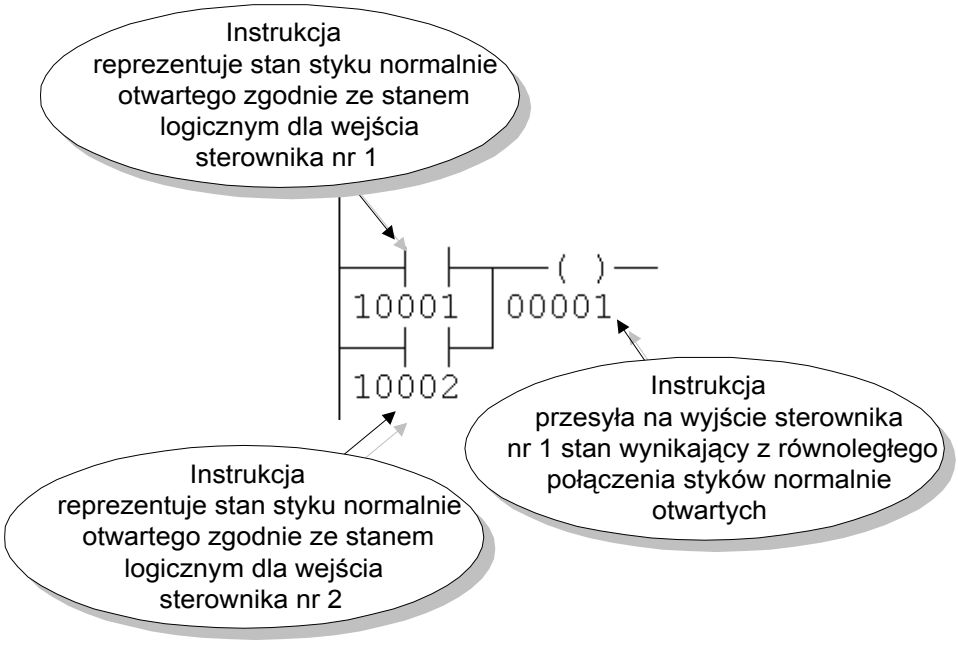
Zestaw instrukcji stykowych dla sterownika Modicon Micro przedstawia Tab. 3. Instrukcje tego typu służą do przeprowadzania operacji logicznych na bitach. W szczególności, ich zadaniem jest umożliwienie komunikacji z wejściami i wyjściami binarnymi sterownika. Dla tego typu instrukcji zarówno argumentami jak i wynikami są zawartości zmiennych (rejestrów) bitowych o adresach typu `1xxxx` i `0xxxx`.

Instrukcje te są najczęściej wykorzystywane do operacji logicznych *OR* (połączenie równoległe styków), *AND* (połączenie szeregowe styków) oraz jako jednobitowe flagi typu *LATCH*.

Tab. 3. Symbole, opis i argumenty instrukcji stykowych

Symbol	Opis elementu (instrukcji)	Argumenty
⊣ ⊢	Styk normalnie otwarty	`Adres zmiennej: 0xxxx, 1xxxx`
⊣/⊢	Styk normalnie zwarty	`Adres zmiennej: 0xxxx, 1xxxx`
⊣P⊢	Styk chwilowej aktywacji zboczem narastającym	`Adres zmiennej: 0xxxx, 1xxxx`
⊣N⊢	Styk chwilowej aktywacji zboczem opadającym	`Adres zmiennej: 0xxxx, 1xxxx`
( )	Cewka normalna, stan OFF przy braku zasilania	`Adres zmiennej: 0xxxx`
(L)	Cewka z pamięcią przez okres jednego cyklu skanowania programu drabinkowego	`Adres zmiennej: 0xxxx`

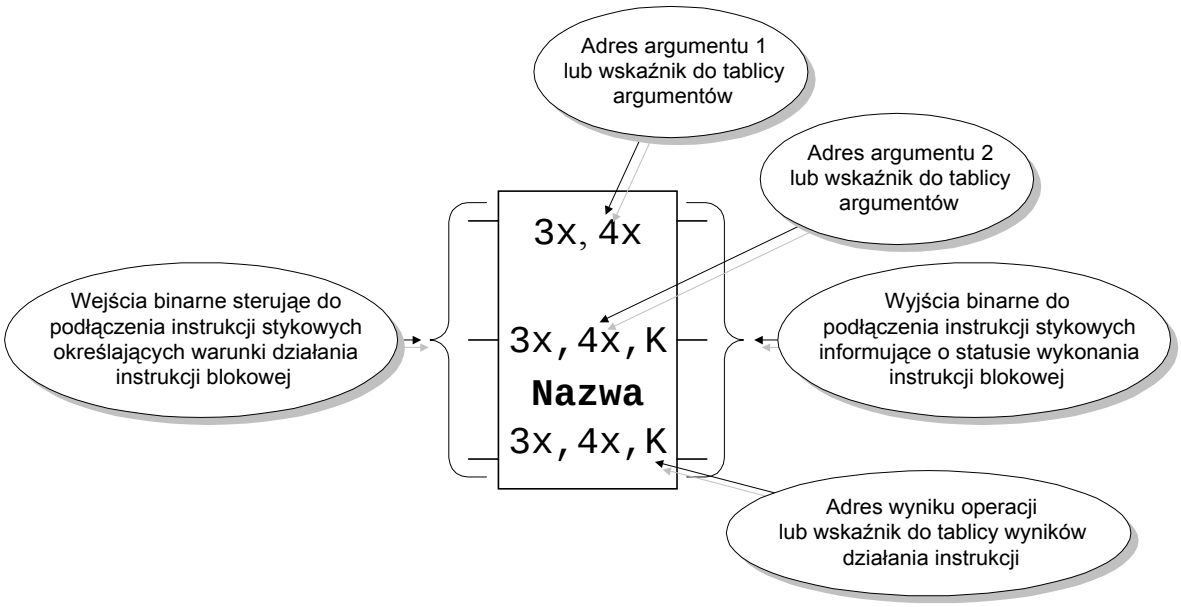
Przykład: Drabinkowa realizacja funkcji *OR*.



B. Instrukcje bloków funkcyjnych

Instrukcje te służą do uproszczenia programowania poprzez sprowadzenie bardziej skomplikowanych operacji lub zadań do jednego bloku, którego warunki działania są określone przez instrukcje stykowe (binarne wejścia bloku). Z punktu widzenia programisty istotna jest wykonywana przez blok operacja oraz typy jego argumentów. Przykładem tego typu bloków mogą być liczniki programowalne oraz *timery* – czyli bloki odmierzające czas. Innym przykładem może być instrukcja regulatora cyfrowego PID, która realizuje skomplikowany algorytm sterowania na podstawie ponad 20 argumentów.

Konstrukcja bloku funkcyjnego może być różnorodna ze względu na sposób działania i liczbę argumentów. Budowę typowego bloku w dialekcie języka drabinkowego dla sterownika Modicon Micro przedstawia Rys. 4.



Rys. 4. Oznaczenie instrukcji blokowej
3X, 4X - oznaczają typy rejestrów mogących być argumentami
K - oznacza, że jako argument dopuszczalna jest wartość stała

Poniżej podany jest tabelaryczny spis instrukcji blokowych z podziałem na grupy. Każda z grup zawiera zestaw instrukcji realizujących podobne typy operacji np.: operacje arytmetyczne, logiczne, przenoszenia danych itd. Do każdej grupy został dołączony przykład wykorzystania instrukcji reprezentatywnej dla danej grupy. Dokładny opis instrukcji jest dostępny w pomocy podręcznej programu Modsoft. Aby uzyskać opis należy wskazać daną instrukcję kursorem i nacisnąć kombinację klawiszy **<ALT> + <H>**.

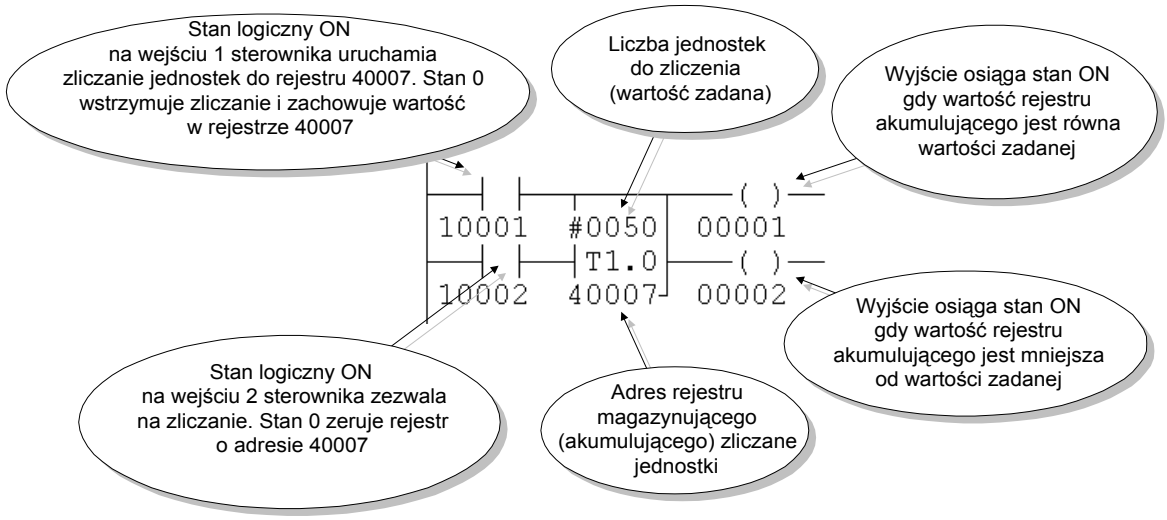
Tab. 4. Instrukcje blokowe liczników

Instrukcje liczników		
Instrukcja	Opis	Argumenty
UCTR	Licznik liczący w górę	4xxxx,K - rejestr lub stała określające próg zliczania 4xxxx - rejestr przechowujący zliczoną liczbę impulsów
DCTR	Licznik zliczający w dół	4xxxx,K - rejestr lub stała określające próg zliczania 4xxxx - rejestr przechowujący zliczoną liczbę impulsów

Tab. 5. Instrukcje blokowe timerów

Instrukcje układów czasowych (timerów)		
Instrukcja	Opis	Argumenty
T1.0	Układ czasowy o skoku liczenia 1 sekundy	4xxxx,K - rejestr lub stała określające zadaną liczbę jednostek (skoków) czasu, 4xxxx - rejestr odmierzający (akumulujący)
T0.1	Układ czasowy o skoku liczenia 0.1 sekundy	4xxxx,K - rejestr lub stała określające zadaną liczbę jednostek (skoków) czasu, 4xxxx - rejestr odmierzający (akumulujący)
T0.01	Układ czasowy o skoku liczenia 0.01 sekundy	4xxxx,K - rejestr lub stała określające zadaną liczbę jednostek (skoków) czasu, 4xxxx - rejestr odmierzający (akumulujący)
T1MS	Układ czasowy o skoku liczenia 0.001 sekundy	4xxxx,K - rejestr lub stała określające zadaną liczbę jednostek (skoków) czasu, 4xxxx - rejestr odmierzający (akumulujący)

Przykład: Układ czasowy o skoku zliczania 1 s. Cewka `00001` jest w stanie ON po odmierzeniu zadanego czasu, cewka `00002` jest w stanie ON w trakcie odmierzania czasu i kiedy timer nie jest aktywny.



Tab. 6. Instrukcje blokowe przesyłania danych

Instrukcje przesyłania danych		
Instrukcja	Opis	Argumenty

R->T	Kopiowanie zawartości rejestru `Arg1` do tablicy o adresie początkowym `Arg2+1` i długości `K`. Wskaźnik do tablicy jest zwiększany o 1 w każdym cyklu aktywności bloku.	`Arg1: 0x, 1x, 3x lub 4x`
		`Arg2: 4x`
		Stała `K`

T->R	Kopiowanie zawartości tablicy o adresie początkowym `Arg1` i długości `K` do rejestru o adresie `Arg2+1.` Wskaźnik do tablicy jest zwiększany o 1 w każdym cyklu aktywności bloku.	`Arg1: 0x, 1x, 3x lub 4x`
		`Arg2: 4x`
		Stała `K`

T->T	Kopiowanie zawartości tablicy o adresie początkowym `Arg1` i długości `K` do tablicy o adresie początkowym `Arg2+1.` Wskaźnik do tablicy jest zwiększany o 1 w każdym cyklu aktywności bloku.	`Arg1: 0x, 1x, 3x lub 4x`
		`Arg2: 4x`
		Stała `K`

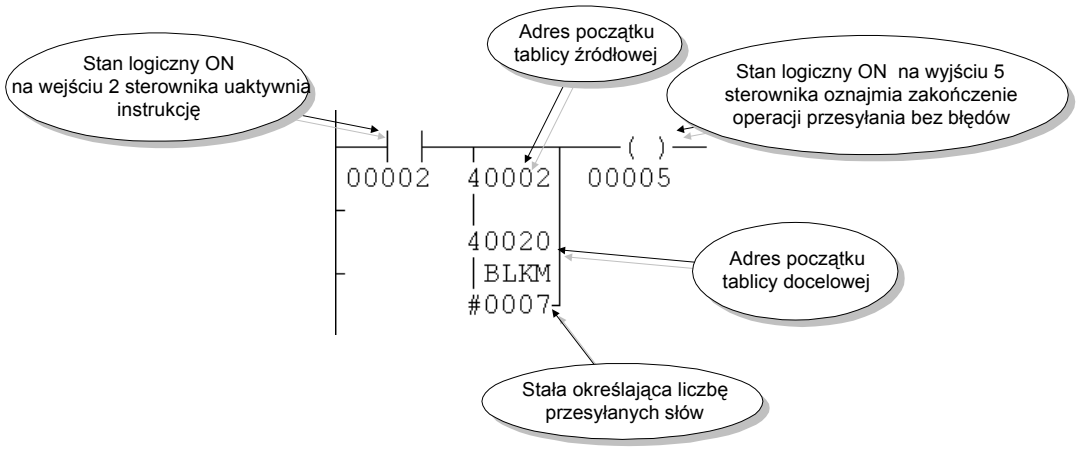
BLKM	Kopiowanie blokowe zawartości tablicy `Arg1` o długości `K` do tablicy `Arg2` w czasie jednego cyklu skanowania.	`Arg1: 0x, 1x, 3x lub 4x`
		`Arg2: 0x, 4x`
		Stała `K`

FIN	Zapełniane stosu o wskaźniku `Arg2` kolejką danych o długości `K` z rejestru źródłowego `Arg1`. Współpracuje z funkcją FOUT.	`Arg1: 0x, 1x, 3x lub 4x`
		`Arg2: 0x, 4x`
		Stała `K`

FOUT	Zdejmowanie kolejki danych o długości `K` ze stosu o wskaźniku `Arg1` (utworzonego przez funkcję FIN) do rejestru `Arg2.`	`Arg1: 4x`
		`Arg2: 0x,4x`
		Stała `K`

SRCH	Przeszukiwanie tablicy o adresie początkowym `Arg1` i długości `K` w poszukiwaniu zadanego wzorca bitowego umieszczonego pod adresem `Arg2+1`. Binarne wyjście statusowe informuje o znalezieniu wzorca.	`Arg1: 3x lub 4x`
		`Arg2: 4x`
		Stała `K`

Przykład: Przesyłanie 7 słów z tablicy do tablicy.



Tab. 7. Instrukcje arytmetyki stałoprzecinkowej

Instrukcje arytmetyczne i porównania		
Instrukcja	Opis	Argumenty

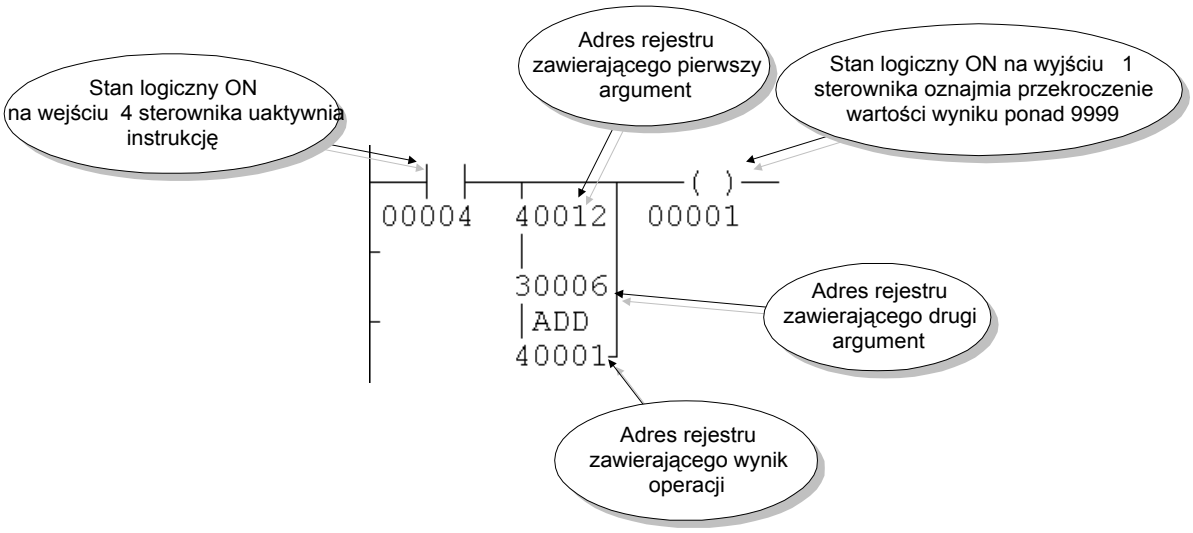
ADD	Dodawanie (**BCD do 9999, bez znaku)** zawartości rejestrów o adresach `Arg1` i `Arg2`. Rezultat operacji umieszczany jest pod adresem `Wynik.` Przepełnienie sygnalizowane na wyjściu statusowym.	`Arg1: 3x,4x,K`
		`Arg2: 3x,4x,K`
		`Wynik: 4x`

SUB	Odejmowanie **(BCD do 9999, bez znaku)** zawartości rejestrów o adresach `Arg1` i `Arg2`. Rezultat operacji umieszczany jest pod adresem `Wynik` (bez znaku). Znak minus sygnalizowany na wyjściu statusowym. **Funkcja służy również do porównywania liczb typu UW.**	`Arg1: 3x,4x,K`
		`Arg2: 3x,4x,K`
		`Wynik: 4x`

MUL	Mnożenie zawartości rejestrów o adresach `Arg1` i `Arg2`. **Argumenty BCD do 9999, bez znaku.** Rezultat operacji umieszczany jest pod adresem `Wynik` (bardziej znaczące bity) oraz `Wynik+1` (mniej znaczące bity)`.` .	`Arg1: 3x,4x,K`
		`Arg2: 3x,4x,K`
		`Wynik: 4x`

DIV	Dzielenie zawartości rejestrów o adresach `Arg1` (bardziej znaczące bity dzielnej) oraz `Arg+1` (mniej znaczące bity dzielnej) i `Arg2`. Rezultat operacji umieszczany jest pod adresem `Wynik` (część całkowita) oraz `Wynik+1` (reszta). **Wynik BCD do 9999, bez znaku.**	`Arg1: 3x,4x,K`
		`Arg2: 3x,4x,K`
		`Wynik: 4x`

Przykład: Dodawanie zawartości 2 rejestrów typu UW.



Tab. 8. Instrukcje logiczne na bitach i blokach danych

Instrukcje operacji logicznych		
Instrukcja	Opis	Argumenty

AND	Operacja AND na odpowiadających sobie bitach 2 tablic o długości K słów i początkowych adresach `Arg1` i `Arg2` . Wynik operacji zapisywany jest w tablicy o adresie `Arg2`.	`Arg1:0x,1x,3x,4x`
		`Arg2: 0x,4x`
		Stała `K`

OR	Operacja OR na odpowiadających sobie bitach 2 tablic o długości K słów i początkowych adresach `Arg1` i `Arg2` . Wynik operacji zapisywany jest w tablicy o adresie `Arg2`.	`Arg1:0x,1x,3x,4x`
		`Arg2: 0x,4x`
		Stała `K`

XOR	Operacja XOR na odpowiadających sobie bitach 2 tablic o długości K słów i początkowych adresach `Arg1` i `Arg2` . Wynik operacji zapisywany jest w tablicy o adresie `Arg2`.	`Arg1:0x,1x,3x,4x`
		`Arg2: 0x,4x`
		Stała `K`

COMP	Negacja bitów tablicy o adresie początkowym `Arg1` i kopiowanie wyniku operacji do tablicy o adresie początkowym `Arg2`. Długość tablic określa stała `K`.	`Arg1:0x,1x,3x,4x`
		`Arg2: 0x,4x`
		Stała `K`

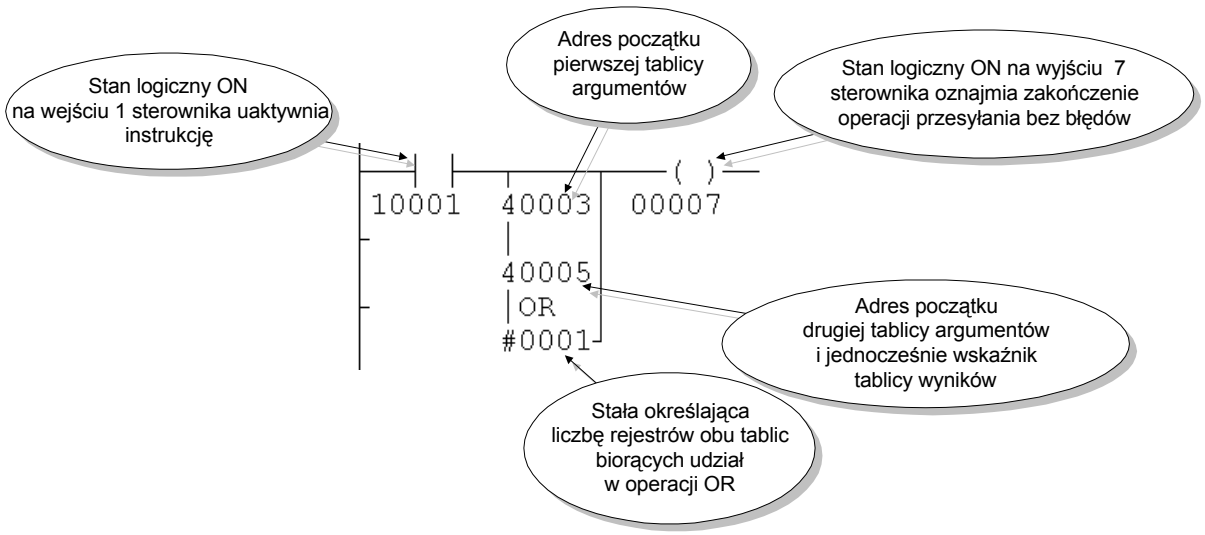
CMPR	Sprawdzanie identyczności zawartości dwóch tablic bitowych o adresach początkowych `Arg1` i `Arg2+1` oraz długości `K`.	`Arg1:0x,1x,3x,4x`
		`Arg2: 0x,4x`
		Stała `K`

MBIT	Zmiana stanu logicznego pojedynczego bitu wskazywanego przez zawartość `Arg1` w tablicy o adresie początkowym `Arg2` i długości `K .`	`Arg1: 3x,4x,K`
		`Arg2: 0x,4x`
		Stała `K`

SENS	Wykrywanie stanu logicznego pojedynczego bitu w tablicy o adresie początkowym `Arg2` i długości `K`. Zawartość `Arg1` wskazuje zadaną pozycję bitu.	`Arg1: 3x,4x,K`
		`Arg2: 0x,4x`
		Stała `K`

BROT	Rotacja lub przesunięcie bitów w tablicy o jedną pozycję w każdym cyklu skanowania pod warunkiem aktywności bloku. `Arg1` określa początek rozpatrywanej tablicy o długości `K`. `Arg2` zawiera adres tablicy, do której kopiowany jest wynik operacji.	`Arg1:0x,1x,3x,4x`
		`Arg2: 0x,4x`
		Stała `K`

Przykład: Realizacja funkcji *OR* na poszczególnych bitach tablic.



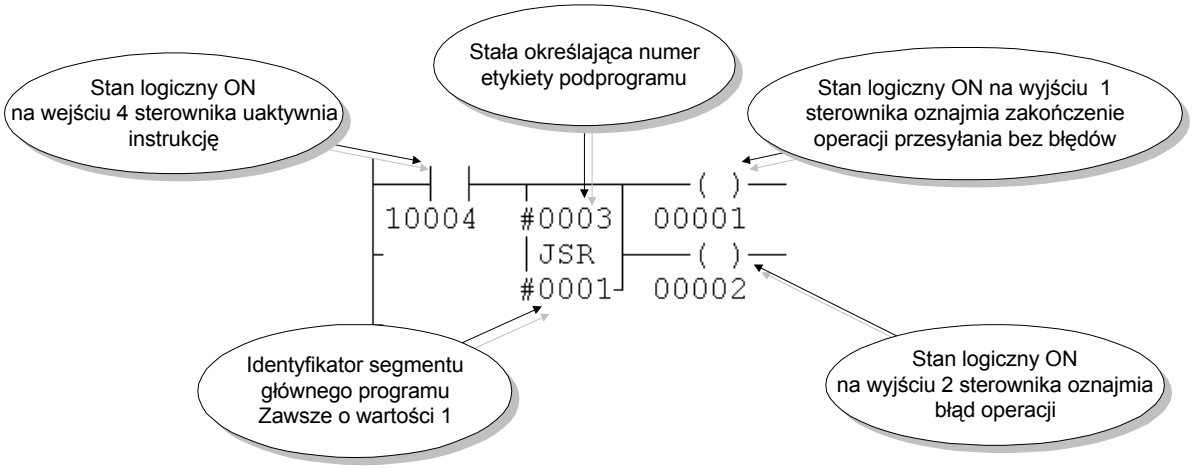
Tab. 9. Instrukcje sterujące

Instrukcje sterowania wykonywaniem programu		
Instrukcja	Opis	Argumenty
JSR	Skok do podprogramu w segmencie 2 do instrukcji LAB wskazanej przez stałą K lub zawartość rejestru 4x.	4x lub K
LAB	Instrukcja początku podprogramu o numerze K w segmencie 2.	K
RET	Instrukcja powrotu z podprogramu (argumentem zawsze jest 1)	1
SKIP	Pominięcie rozwiązywania sieci o podanym numerze K.	K

Tab. 10. Instrukcje specjalne

Instrukcje specjalne		
Instrukcja	Opis	Argumenty
PID2	Regulator cyfrowy PID. Arg1 jest adresem początkowym tabeli 21 rejestrów określających parametry regulatora. Arg2 jest adresem początkowym tabeli 9 rejestrów używanych przez blok do obliczeń. K określa okres próbkowania jako wielokrotność 0.1 s	Arg1:4x Arg2:4x Stała K
EMTH	Rozszerzone operacje matematyczne: operacje zmiennoprzecinkowe, logarytm, pierwiastek kwadratowy itp. Arg1 i Arg2 są adresami argumentów operacji. K określa rodzaj operacji np. K=5 odpowiada pierwiastkowi kwadratowemu.	Arg1:4x Arg2:4x Stała K
COMM	Komunikacja w trybie ASCII poprzez port szeregowy sterownika. Arg1 jest adresem początkowym tabeli 10 rejestrów określających parametry transmisji. Arg2 jest adresem początkowym tabeli o długości K która jest buforem dla danych.	Arg1:4x Arg2:4x Stała K

Przykład: Skok do podprogramu.



3. Zadania do wykonania

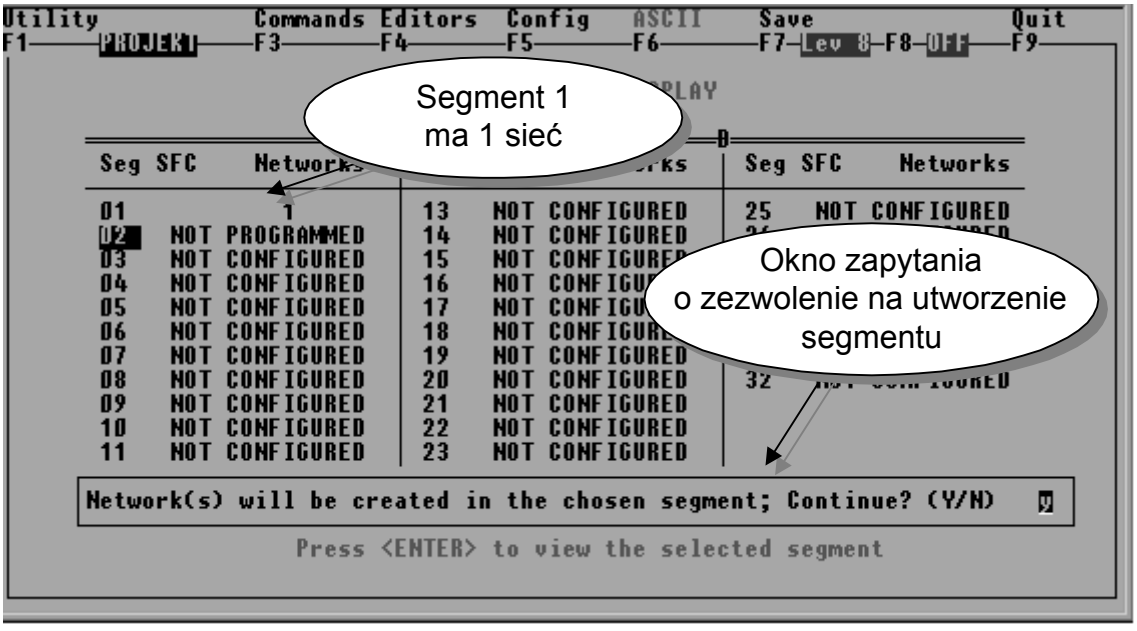
Ćwiczenie obejmuje następujące elementy:

- stworzenie nowego projektu,
- zapoznanie się z edycją i działaniem typowych bloków funkcyjnych,
- rozszerzenie programu głównego o podprogram.

Kolejność wykonywania poszczególnych czynności jest przedstawiona poniżej.

3.1. Etap 1 – tworzenie nowego projektu

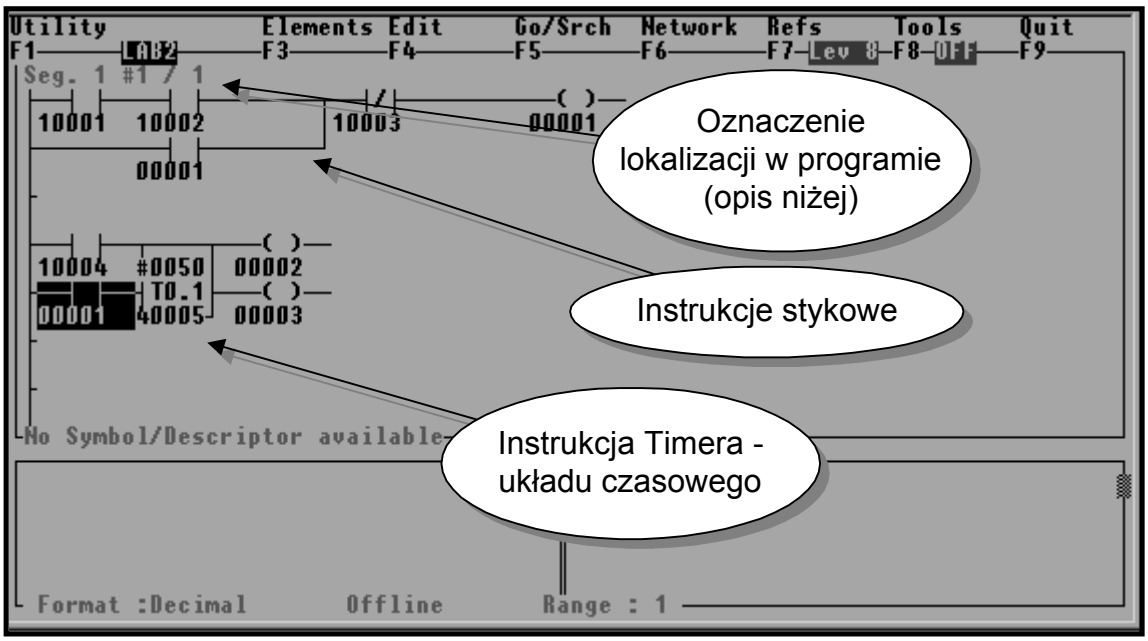
1. Uruchomić program **`modsoft.exe`.**
2. Za pomocą klawisza **<TAB>** i kursorów otworzyć menu *Offline,* a następnie wybrać polecenie *New Program***.**
3. Program poprosi o podanie nazwy projektu i katalogu, w jakim ma zostać umieszczony – należy wpisać odpowiednie dane i zatwierdzić klawiszem **<ENTER>**.
4. Przeprowadzić konfigurację sterownika i otworzyć segment 1 jak w ćwiczeniu 1.
5. Przejść do okna edycji programu i zaprogramować przynajmniej jeden poprawny szczebel logiki drabinkowej (będzie on podstawą do dalszej edycji w trybie *Online*).
6. Przejść do menu głównego programu Modsoft i za pomocą klawisza **<TAB>** i kursorów otworzyć menu *Transfer,* a następnie wybrać polecenie *File To PLC***.** Operacje te spowodują załadowanie utworzonego programu do sterownika.
7. Po zakończeniu ładowania programu należy przejść klawiszem **<ESC>** do głównego ekranu i z menu *Online* wybrać polecenie *Select program*. Z listy przedstawionej przez Modsoft wybrać nazwę aktualnego projektu i po zatwierdzeniu wyboru poczekać na pojawienie ekranu edytora z programem .
8. Za pomocą klawisza **<ESC>** uzyskać ekran konfiguracji segmentów programu jak na Rys. 5. Przejść kursorem do pozycji *02* w kolumnie *Seg* i nacisnąć **<ENTER>.** Na zadane pytanie należy odpowiedzieć **<Y>** i zatwierdzić klawiszem **<ENTER>**. Operacje powyższe spowodują zainicjowanie segmentu 2, który przeznaczony jest tylko dla podprogramów.



Rys. 5. Konfiguracja segmentów

3.2. Etap 2 – edycja programu głównego (Segment 1)

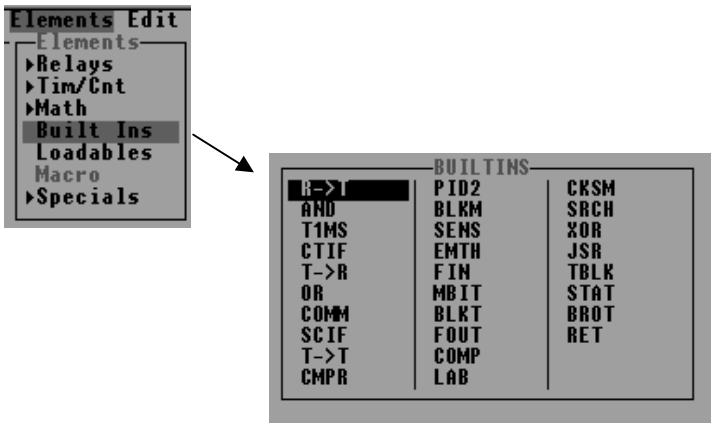
1. Przejść kursorem do pozycji *01* w kolumnie *Seg*, a następnie nacisnąć **<ENTER>**. Operacja ta otwiera edytor dla segmentu 1 (programu głównego) w trybie *Online*.
2. Za pomocą klawisza **<TAB>** należy przejść do menu *Elements*. Opcja ta zawiera listę dostępnych instrukcji i umożliwia wstawianie ich do tworzonego programu.
3. Wprowadzić przykładowy program jak na Rys. 6.



Rys. 6. Sieć 1 programu głównego

Dla programisty istotne jest oznaczenie w lewym górnym rogu ekranu np.: Seg. 1 #1/1 lub Seg. 2 #3/5. Określa ono lokalizację fragmentu edytowanego programu. Pierwsza liczba oznacza nr segmentu, # - jest symbolem sieci, np.: Seg. 2 #3/5 oznacza sieć nr 3 spośród 5 sieci w segmencie 2.

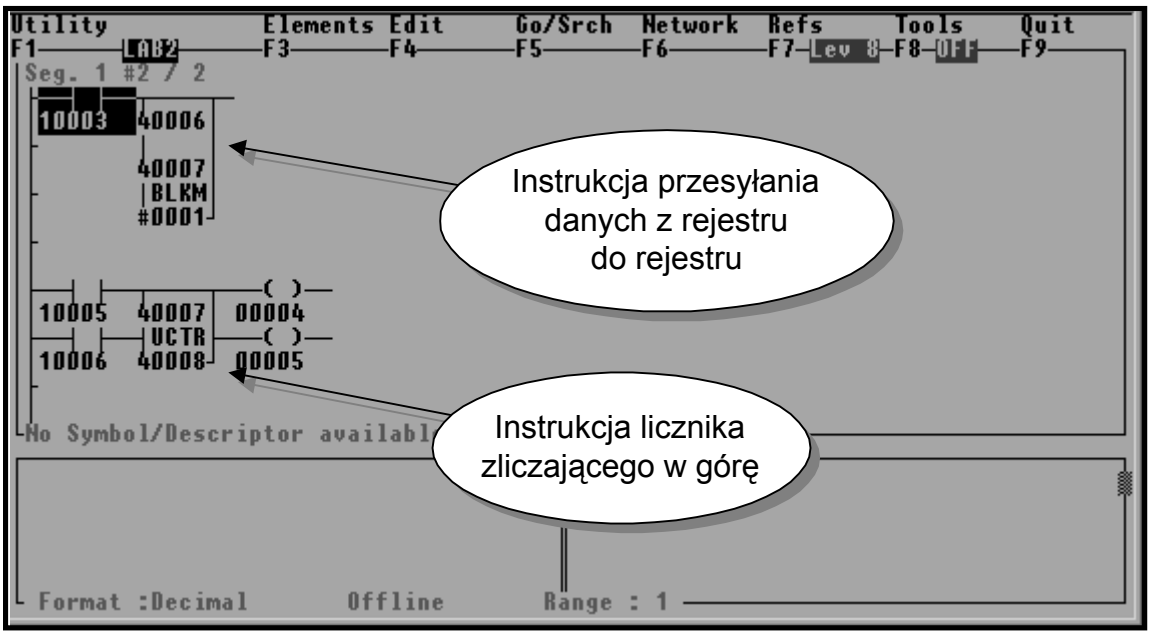
Menu *Elements* pozwala na wybranie z listy instrukcji, którą chce się wprowadzić do programu w miejscu, w którym znajduje się kursor.



Inna metoda wprowadzania polega na użyciu odpowiedniego symbolu klawiszowego (dla instrukcji stykowych) lub wprowadzeniu mnemonicznej nazwy instrukcji (dla instrukcji blo-

kowych) w edytorze np.: LAB, PID2, OR itp. W miejscu nazwy pojawi się dany blok funkcyjny.

4. Posługując się zadajnikiem sprzętowym lub programowym sprawdzić funkcjonowanie wprowadzonego fragmentu programu. **Uwaga:** Wykorzystać okno *Reference Data* do obserwacji i zmian zawartości rejestrów.
5. Za pomocą klawisza **<TAB>** przejść do menu *Network*. Po rozwinięciu menu wybrać opcję *Insert After Network*. Opcja ta pozwala na dodanie nowej sieci programu w danym segmencie. Sieć jest wstawiona za siecią aktualną.
6. W obszarze nowej sieci wprowadzić instrukcje jak na Rys. 7.

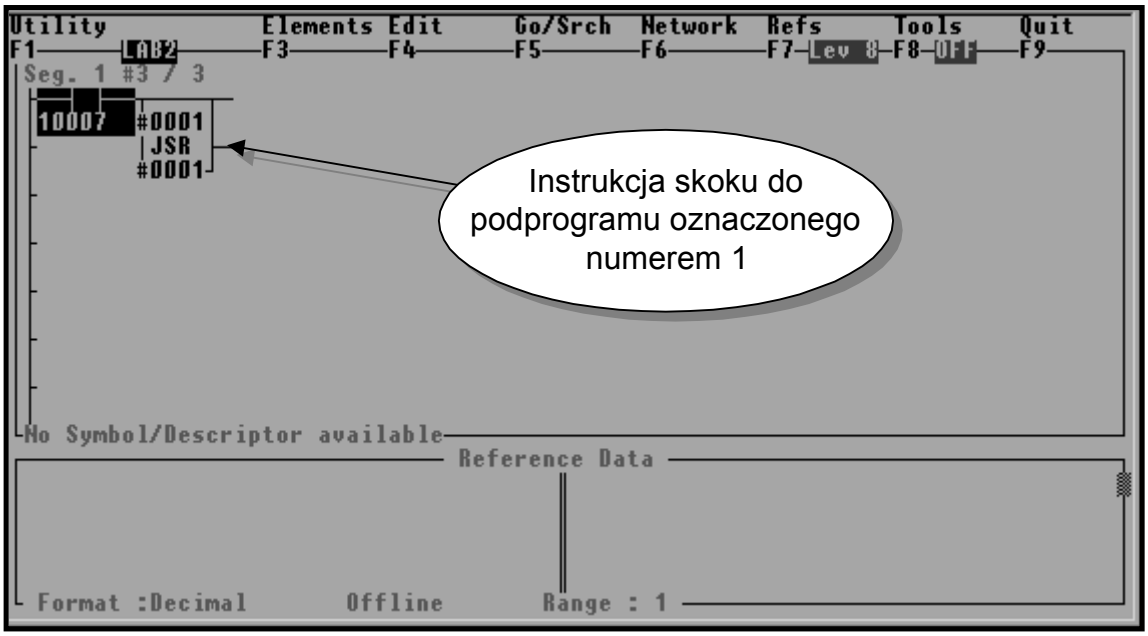


Rys. 7. Sieć 2 programu głównego

7. Posługując się zadajnikiem sprzętowym lub programowym sprawdzić funkcjonowanie wprowadzonych instrukcji. Zwrócić uwagę na funkcje spełniane przez wykorzystywane przez instrukcje rejestry.
8. Przejść do sieci 2 (do przemieszczania się po obszarze segmentu programu służą klawisze **<PgUp>** i <**PgDn>)** .Za pomocą klawisza **<TAB>** przejść do menu i opcji *Network*. Wybrać podopcję *Insert After Network* dodającą kolejną sieć.
9. W obszarze nowej sieci wprowadzić instrukcje jak na Rys. 8.

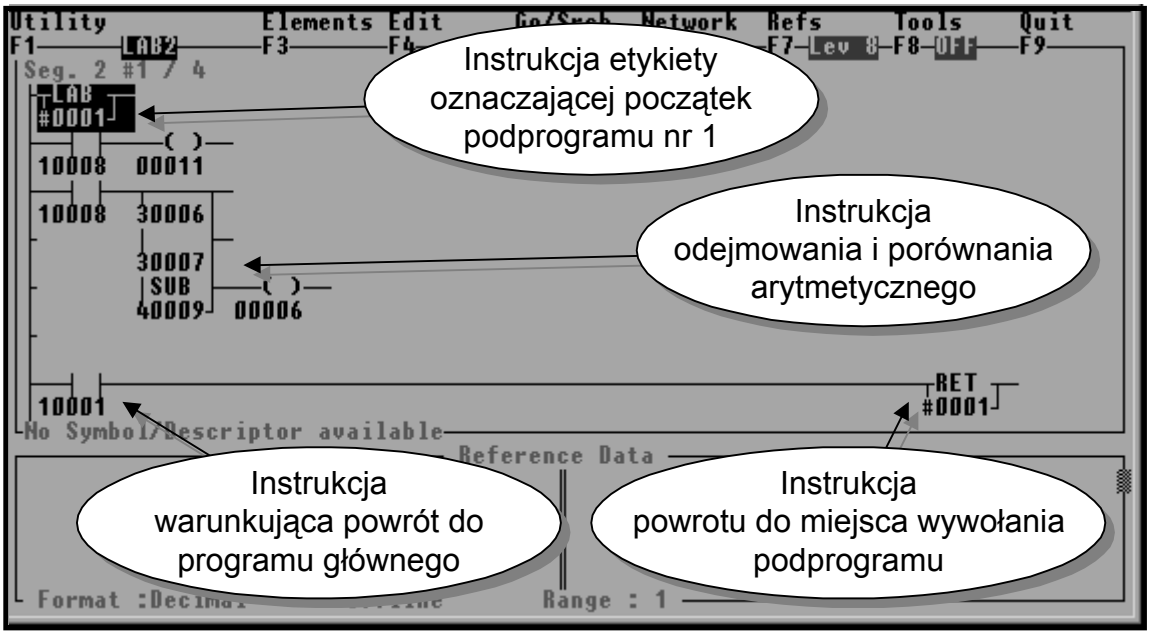
Etap 3 - tworzenie podprogramu (Segment 2)

1. Za pomocą klawisza **<ESC>** przejść do ekranu jak na Rys. 5.
2. Przejść kursorem do pozycji *02* w kolumnie *Seg,* a następnie nacisnąć **ENTER**. Operacja ta otwiera edytor dla segmentu 2 (segmentu podprogramów).
3. W obszarze edycji sieci wprowadzić instrukcje jak na Rys. 9.



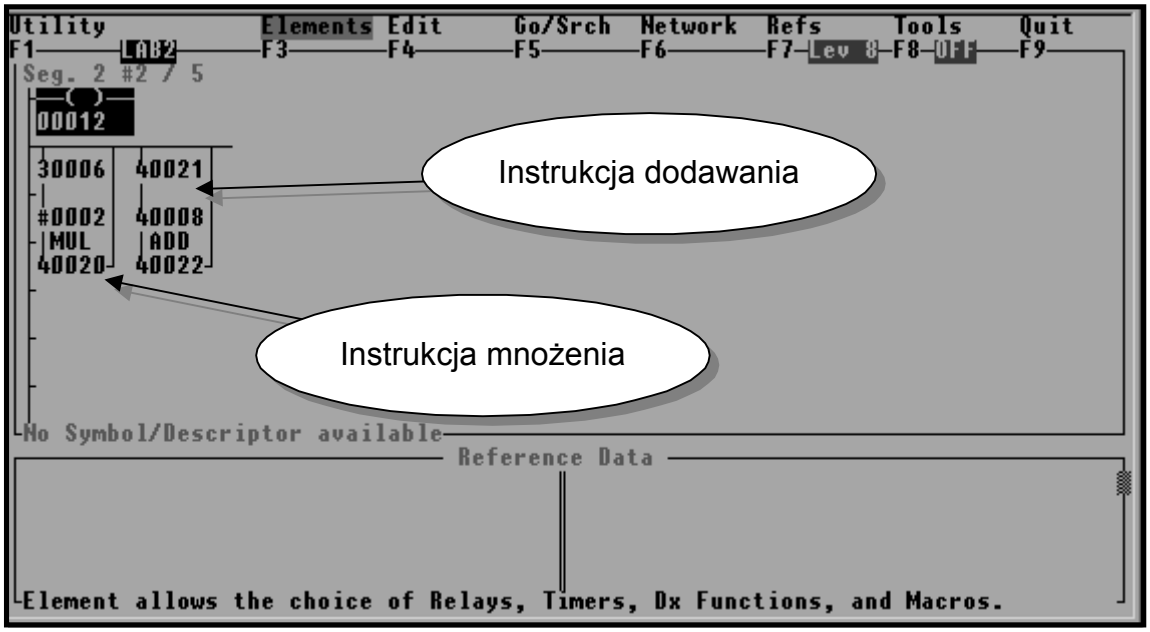
Rys. 8. Sieć 3 programu głównego

Okno edytora jest podzielone na wiersze i kolumny tworzące obszary, w których mogą być wprowadzane instrukcje. Często jednak zachodzi potrzeba modyfikacji sieci programu poprzez dodanie nowej instrukcji, tak aby instrukcje mogły zmieścić się w tej samej sieci. Pomocne wówczas są funkcje z menu Edit: Open Row i Open Column. Pozwalają one na wstawienie całego wiersza lub kolumny między instrukcjami. Operacja tego typu jest szczególnie wygodna dla instrukcji blokowych, mogących zajmować obszar o wysokości 2 lub 3 wierszy.



Rys. 9. Sieć 1 podprogramu

4. Za pomocą klawisza **<TAB>** przejść do menu i opcji *Network*. Wybrać podopcję *Insert After Network*. W ten sam sposób wstawić kolejne 2 sieci w segmencie 2.
5. W obszarze sieci 2 podprogramu wprowadzić instrukcje jak na Rys. 10.



Rys. 10. Sieć 2 podprogramu

10. Korzystając z okna *Reference Data* oraz zadajnika programowego lub sprzętowego zbadać działanie poszczególnych instrukcji. Zbadać warunki wywoływania podprogramu.
11. Za pomocą klawisza **<ECS>** opuścić tryb *Online*.
12. Przejść do ekranu głównego programu Modsoft. Za pomocą klawisza **<TAB>** z menu *Transfer* wybrać podopcję *PLC to File*. Operacje powyższe mają za zadanie przesłanie do komputera PC programu wprowadzonego poprzednio do pamięci sterownika w trybie *Online* i zapisanie go dysku komputera PC.
13. Przejść do ekranu głównego programu Modsoft. Z menu *Offline* wybrać opcję *Select Program*. Wybrać odpowiedni projekt i wczytać do środowiska uruchomieniowego.
14. Przejść do ekranu głównego przy pomocy klawisza **<ESC>.** Wybrać menu *Tools* i polecenie *Print*. Wybrać z listy nazwę projektu.
15. Dobrać parametry dokumentacji zgodnie z zaleceniami prowadzącego.
16. Wydrukować zawartość projektu.

Literatura:

1. Grandek K. Rojek R. *Mikroprocesorowe sterowniki programowalne.* skrypt WSI Opole, 1991.
2. Małysiak H. *Układy przełączające w automatyce przemysłowej - zadania.* WNT Warszawa 1981.
3. Mikulczyński T. Samsonowicz Z. *Automatyzacja dyskretnych procesów produkcyjnych.* WNT Warszawa 1997.
4. Siwiński J. *Układy przełączające w automatyce.* WNT Warszawa 1980.
5. Trybus L. *Regulatory wielofunkcyjne*, WNT, 1992.
6. Modicon Micro 512/612. Sprzęt PLC – *Podręcznik użytkownika*.
7. Modicon Micro. *Przykłady programowania*.