mgr inż. Łukasz Dworzak, Politechnika Wrocławska
mgr inż. Sergiusz Ciskowski, Politechnika Wrocławska
prof. dr hab. inż. Tadeusz Mikulczyński, Politechnika Wrocławska
dr inż. Marian Bogdan, Politechnika Wrocławska

SYNTEZA METODĄ GRAFPOL SEKWENCYJNYCH ALGORYTMÓW STEROWANIA

Opisany w normie IEC 1131-3 język SFC (Sequential Function Chart) programowania sterowników PLC, mimo upływu kilkunastu lat od momentu zdefiniowania, nie doczekał się, jak pierwotnie można było oczekiwać, szerokiego zastosowania. Ponadto nawet dziś można się posługiwać jego odpowiednikami jedynie w produktach zaledwie kilku producentów sterowników. Powszechnie stosowanym językiem jest natomiast, wywodzący się jeszcze z języka schematów stykowo-przekaźnikowych, język LD (Ladder Diagram). Oprócz niepodważalnej zalety, jaką są niewielkie wymagania sprzętowe związane z zajmowanym obszarem pamięci, język ten charakteryzuje cykliczne przetwarzanie programu użytkownika w sterowniku. To z kolei utrudnia jego stosowanie i wymusza konieczność użycia pamięci. Opracowane dotychczas metody określania postaci tych pamięci oraz miejsc ich zastosowania w algorytmach sterowania nie w pełni spełniły wymagania branży. W związku z tym postanowiono jeszcze raz przyjrzeć się problematyce pamięci wykorzystywanych w języku LD. Celem prowadzonych prac było stworzenie sposobu szybkiego i jednoznacznego określania postaci pamięci oraz miejsc ich użycia w algorytmie sterowania procedur sekwencyjnych.

SYNTHESIS OF SEQUENTIAL CONTROL ALGORITHMS USING GRAFPOL METHOD

Described in IEC 1131-3 standard SFC (Sequential Function Chart) programming language of PLC controllers still, in spite of several years that passed since it was defined, can't find wide application. Even today SFC language can be used to program only a few PLC controllers. The most common language used to program PLC controllers is LD (Ladder Diagram) language which takes its origin from contact-relay language. Besides its undoubted advantages which are small hardware requirements – due to small storage area needed – this language processes user program cyclically forcing use of memory and making its application difficult. Developed methods of determining memory forms and application places in control algorithm are still not satisfying. In that case problem of memory usage in LD language has been raised again. The aim of carried out scientific research was to develop rapid and unequivocal method to determine memory form and place of its application in control algorithm of sequential procedures.

1. WSTĘP

W algorytmach dyskretnych procesów produkcyjnych wyszczególnić można wiele rodzajów procedur. Niezależnie od stopnia i złożoności algorytmu procesu podstawę, w mniejszym lub większym stopniu, zawsze stanowi procedura sekwencyjny. W związku z tym istotną kwestią jest dysponowanie metodami umożliwiającymi projektowanie algorytmów sterowania dla takich procedur i ich implementację w sterownikach PLC.

Z pozoru zadanie to może wydawać się proste do zrealizowania. Norma IEC 1131-3 określa wszak, prosty w użyciu, język SFC (ang. Sequential Function Chart), którego odpowiedniki funkcjonalne można znaleźć w produktach m.in. Telemecaniquea (PL7-Grafcet), Siemensa (Step7 – Graph7) czy FESTO (FST – STL). Co jednak zrobić w przypadku, gdy nie występuje możliwość użycia tego języka? Projektant zmuszony jest wtedy do zastosowania występującego powszechnie języka LD (ang. Ladder Diagram). To z kolei prowadzi do wzrostu stopnia skomplikowania postawionego zadania. Wynika to z zasadniczej cechy tego języka, a mianowicie jego cyklicznego przetwarzanie w sterowniku.

Cecha ta skłaniała wiele zespołów naukowych do poszukiwania metod umożliwiających przejście od algorytmu procesu do algorytmu sterowania. Wspomnieć tu można o metodzie tablic kolejności łączeń [1]. Metodę tę charakteryzuje jednak ograniczenie co do sumy sygnałów wejściowych i wyjściowych modelowanego procesu.

Podobne prace prowadzono w Laboratorium Podstaw Automatyzacji Instytutu Technologii Maszyn i Automatyzacji Politechniki Wrocławskiej w wyniku których opracowano metodę modelowania procedur sekwencyjnych (Grafpol) [2]. Elementem składowym tej metody był sposób wyznaczania postaci niezbędnych pamięci (Metoda Transformacji Sieci). W związku z pracochłonnością charakteryzującą ten sposób, jest on w praktycznym stosowaniu dość uciążliwy. Wynika to z konieczności graficznego przedstawiania zależności opisujących funkcje zmiennych wyjściowych algorytmu sterowania.

Dlatego też podjęto odpowiednie prace, których celem było opracowanie sposobu wyznaczania postaci pamięci, tj. warunków jej zapisu, kasowania oraz zasad stosowania w algorytmie sterowania. Sposób ten powinien funkcjonować w przypadku algorytmów o dużej liczbie sygnałów wejściowych i wyjściowych modelowanego procesu, bez konieczności graficznego przedstawiania funkcji zmiennych wyjściowych algorytmu procesu. Ponadto, ze względu na wysoki stopień znajomości języka LD wśród automatyków [3], poszukiwane rozwiązanie powinno umożliwiać prostą implementację w sterowniku PLC przy wykorzystaniu tego języka. Rozwiązanie postawionego zadania, w odniesieniu do procesów sekwencyjnych, przedstawiono w niniejszej publikacji na przykładzie napędów pneumatycznych.

2. SYNTEZA POSTACI ZMIENNYCH STERUJĄCYCH I ELEMENTARNYCH KOMÓREK PAMIĘCI

Synteza programu użytkowego sterownika PLC oparta jest o sieć operacyjną i Grafpol GP, reprezentujące algorytm automatyzowanego procesu produkcyjnego. Podczas syntezy programu użytkowego PLC obowiązują następujące zasady:

Zasada 1

Liczba elementarnych komórek pamięci użytych do realizacji automatyzowanego procesu jest równa L i określa ją zależność:

$$L = \frac{\sum_{i=1}^{n} E_i}{2}, \tag{1}$$

gdzie:

n - liczba etapów elementarnych,

E_i - i-ty etap elementarny.

Zasada 2

Zapis komórki pamięci następuje w stanie, w którym zostaje zakończone wykonanie ruchu roboczego poszczególnego napędu z pozycji wyjściowej. Stany te opisują tranzycje t_i, które sygnalizują zakończenie wykonania etapu E_i.

Zależności opisujące zapis elementarnych komórek pamięci mają następujące postaci:

$$M_1(S)=t_1, \tag{2}$$

$$M_j(S)=t_i \cdot M_{j-1} \cdot t_{i-1}, \quad dla \ j=2 \ldots L \ i \ dla \ i=2 \ldots n-1, \tag{3}$$

UWAGA 1
W tranzycjach t_i nie uwzględnia się warunków logicznych zawartych w klatkach warunkowych niezależnych występujących w sieci operacyjnej.

UWAGA 2
W przypadku wielokrotnej pracy tego samego napędu z pozycji wyjściowej każdorazowo jest zapisywana nowa elementarna komórka pamięci.

Zasada 3

Kasowanie wszystkich elementarnych komórek pamięci następuje w ostatnim stanie algorytmu procesu. Zależność opisującą kasowanie wszystkich elementarnych komórek pamięci ma postać:

$$M_{1 \ldots L}(R)=t_n \cdot M_L, \tag{4}$$

Zasada 4

Tranzycja t_i^* określa rozpoczęcie i-tego etapu elementarnego. Stanowi ona iloczyn:

- warunku t_i wynikającego z sieci Grafpol GP,
- warunku sygnalizującego stan wyjściowy danego napędu, który zanika po rozpoczęciu tego ruchu (t_{ZNi}) – jeśli występuje,
- sygnału pamięci (M_j) lub zanegowanego sygnału pamięci $(\overline{M}_j)$ lub iloczynu sygnału pamięci i zanegowanego sygnału pamięci $(M_j \cdot \overline{M}_{j+1} \ lub \ M_j \cdot \overline{M}_{j+2})$.

W sytuacji, gdy w powyższym iloczynie występujące składniki powtarzają się, należy usunąć powtórzenia.

Postać danej tranzycji t_i^* ustalamy na podstawie sieci Grafpol GP i wówczas:

- zanegowany sygnał pamięci występuje we wszystkich tranzycjach poprzedzających tranzycję w której nastąpił zapis pamięci M_j, aż do tranzycji w której zapisywana jest poprzednia elementarna komórka pamięci (M_{j-1}) włącznie.
- sygnał pamięci (M_j) występuje we wszystkich tranzycjach następujących po tranzycji w której nastąpił zapis pamięci M_j do tranzycji w której następuje zapis kolejnej elementarnej komórki pamięci (M_{j+1}) włącznie.

UWAGA 1
Szczególnymi tranzycjami są:

- tranzycja t_0^*, która ma postać:

$$t_0^*=S \cdot t_n \cdot t_{ZN_0} \cdot \overline{M}_1 \tag{5}$$

- tranzycje występujące po zapisie ostatniej elementarnej komórki pamięci (M_L), które mają postać:

$$t_{n-x}^*=t_{n-x} \cdot t_{ZN_{n-x}} \cdot M_L, \quad dla \ x=1,2,.... \tag{6}$$

UWAGA 2
W przypadku napędów sterowanych impulsowo (bistabilnych), w momencie gdy spełniona jest tranzycji t_i^*, zmienna sterująca Y_i ma wartość logiczną 1.

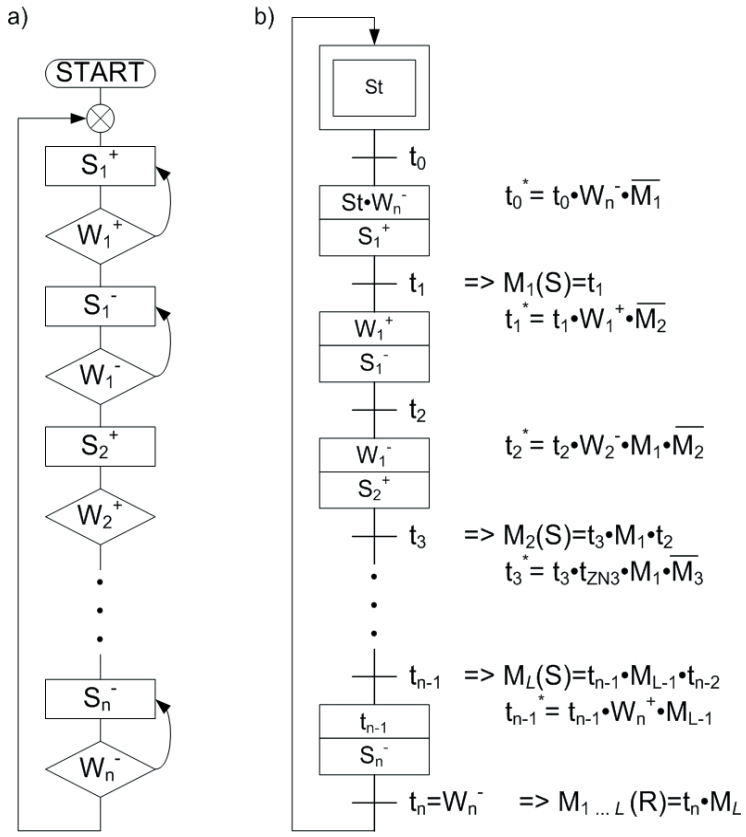
W przypadku napędów sterowanych potencjałowo (monostabilnych) spełnienie tranzycji t_i^*, odpowiadającej za ruch napędu z pozycji wyjściowej, przypisuje zmiennej sterującej Y_i wartość logiczną 1 (tzw. SET). Spełnienie tranzycji t_i^*, odpowiadającej za powrót napędu do pozycji wyjściowej, przypisuje zmiennej sterującej Y_i wartość logiczną 0 (tzw. RESET).

UWAGA 3
W przypadku realizowanych wielokrotnie ruchów przez ten sam napęd pneumatyczny w pojedynczej sekwencji funkcja zmiennej wyjściowej Y_i ma postać:

$$Y_i = Y_{i,1} + Y_{i,2} + \ldots + Y_{i,n} \tag{7}$$

gdzie $Y_{i,n}$ – i-ta zmienna odpowiadająca n-temu powtórzeniu ruchu napędu z pozycji wyjściowej

Powyższe zasady zilustrowano na sieci Grafpol GP (rys. 1b), która powstała w oparciu o przykładowy algorytm pracy napędów pneumatycznych reprezentowany przez sieć operacyjną (rys. 1a).



Rys. 1. Algorytm pracy zapisany w postaci: a) sieci operacyjnej b) sieci Grafpol GP; S_n^+-wysuw tłoczyska n-tego siłownika, S_n^--wsuw tłoczyska n-tego siłownika, W_n^+ - sygnalizacja wysuniętego tłoczyska n-tego siłownika, W_n^- - sygnalizacja wsuniętego tłoczyska n-tego siłownika

Stosując poniższe odwzorowanie etapów elementarnych E_i zmiennymi wyjściowymi układu sterowania Y_i

$$Y_1 \rightarrow S_1^+ \tag{8}$$
$$Y_2 \rightarrow S_1^- \tag{9}$$
$$Y_3 \rightarrow S_2^+ \tag{10}$$
$$\vdots$$
$$Y_n \rightarrow S_n^- \tag{11}$$

otrzymamy następujące równania:

$$Y_1 = St \cdot W_n^- \cdot \overline{M}_{1,} \tag{12}$$
$$Y_2 = W_1^+ \cdot \overline{M}_2, \tag{13}$$
$$Y_3 = W_1^- \cdot W_2^- \cdot M_1 \cdot \overline{M}_2, \tag{14}$$
$$\vdots$$
$$Y_n = t_{n-1} \cdot W_n^+ \cdot M_{L-1}, \tag{15}$$

$$M_1(S) = W_1^+, \tag{16}$$
$$M_2(S) = W_2^+ \cdot M_1 \cdot W_1^-, \tag{17}$$
$$\vdots$$
$$M_{1\ldots L}(R) = W_n^- \cdot M_L \tag{18}$$

Powyżej przedstawione równania są podstawą do implementacji wyznaczonego algorytmu sterowania w sterowniku PLC za pomocą języka LD.

3. PRZYKŁAD

Na rys. 2 pokazano schemat funkcjonalny urządzenia mieszającego elementy A i B przed ich zapakowaniem. W aplikacjach związanych z pakowaniem różnego rodzaju wyrobów często występuje konieczność podania wyrobów do pakowaczki w odpowiedniej kolejności. W rozpatrywanym przypadku pakowanie dotyczy zapełniania opakowania zbiorczego, w którym znaleźć mają się opakowania wyrobów A i B ułożone w sekwencji ABA. Do napędu urządzenia zastosowano dwa napędy pneumatyczne (1A, 2A). Napęd 1A sterowany jest zaworem monostabilnymi (1V), zaś napęd 2A zaworem bistabilnym (2V). Algorytm pracy urządzenia stanowi procedura sekwencyjna składająca się z sześciu etapów elementarnych E1-E6.

ETAP E1: *podanie elementu A*
Realizacja: 1A+ (1Y2+)
Sygnalizacja: 1AS2

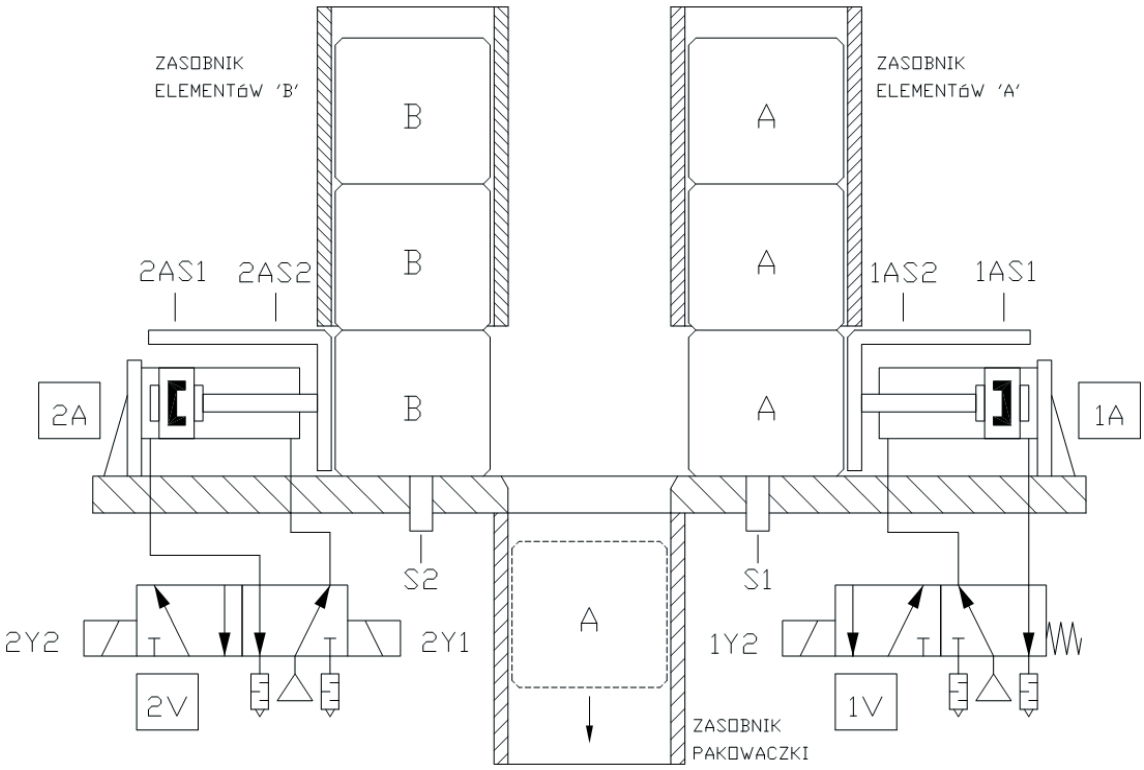
ETAP E2: *wycofanie podajnika elementu A*
Realizacja: 1A- (1Y2-)
Sygnalizacja: 1AS1

ETAP E3: *podanie elementu B*
Realizacja: 2A+ (2Y2)
Sygnalizacja: 2AS2

ETAP E4: *wycofanie podajnika elementu B*
Realizacja: 2A- (2Y1)
Sygnalizacja: 2AS1

ETAP E5: *podanie elementu A*
Realizacja: 1A+ (1Y2+)
Sygnalizacja: 1AS2

ETAP E6: *wycofanie podajnika elementu A*
Realizacja: 1A- (1Y2-)
Sygnalizacja: 1AS1



Rys. 2. Schemat funkcjonalny urządzenia mieszającego podawane do pakowaczki elementy

Ponadto, w trakcie realizacji powyższej procedury, uwzględniane muszą być stany czujników niezależnych od procedury:

- S1 – czy jest element A do podania,
- S2 – czy jest element B do podania.

Powyższy algorytm pracy urządzenia przedstawiono w postaci sieci operacyjnej (rys. 3a) oraz sieci Grafpol GP (rys. 3b).

Zgodnie z Zasadą 1, liczba elementarnych komórek pamięci, niezbędnych do realizacji procesu jest równa L:

$$L = \frac{\sum_{i=1}^{n} E_i}{2} = \frac{6}{2} = 3 \tag{19}$$

Miejsca oraz warunki zapisu i kasowania elementarnych komórek pamięci określono na podstawie sieci Grafpol GP stosując Zasady 2 i 3 (rys. 3b).

Zastosowanie Zasady 4 umożliwiło wyznaczenie postaci tranzycji t_i^* (rys. 3b). Na tej podstawie określono postaci zmiennych wyjściowych układu sterowania Y_i, które odwzorowują etapy elementarne w następujący sposób:

$$Y_{1,1}(S), Y_{1,2}(S) \equiv Q0.0(S) \rightarrow 1\mathrm{Y}2 \rightarrow 1\mathrm{A}+ \tag{20}$$

$$Y_{1,1}(R), Y_{1,2}(R) \equiv Q0.0(R) \rightarrow \overline{1\mathrm{Y}2} \rightarrow 1\mathrm{A}- \tag{21}$$

$$Y_2 \equiv Q0.2 \rightarrow 2\mathrm{Y}2 \rightarrow 2\mathrm{A}+ \tag{22}$$

$$Y_3 \equiv Q0.1 \rightarrow 2\mathrm{Y}1 \rightarrow 2\mathrm{A}- \tag{23}$$

Ostatecznie otrzymano układ równań zmiennych sterujących oraz warunki zapisu i kasowania elementarnych komórek pamięci:

$$\left.\begin{aligned} Y_{1,1}(S) &= St \cdot S1 \cdot 1AS1 \cdot \overline{M_1} \\ Y_{1,2}(S) &= 2AS1 \cdot S1 \cdot 1AS1 \cdot M_2 \cdot \overline{M_3} \end{aligned}\right\} Y_1(S) = St \cdot S1 \cdot 1AS1 \cdot \overline{M_1} + 2AS1 \cdot S1 \cdot 1AS1 \cdot M_2 \cdot \overline{M_3}, \quad (24)$$

$$\left.\begin{aligned} Y_{1,1}(R) &= 1AS2 \cdot \overline{M_2} \\ Y_{1,2}(R) &= 1AS2 \cdot M_2 \end{aligned}\right\} Y_1(R) = 1AS2 \cdot \overline{M_2} + 1AS2 \cdot M_2, \quad (25)$$

$$Y_2 = 1AS1 \cdot S2 \cdot 2AS1 \cdot M_1 \cdot \overline{M_2}, \quad (26)$$

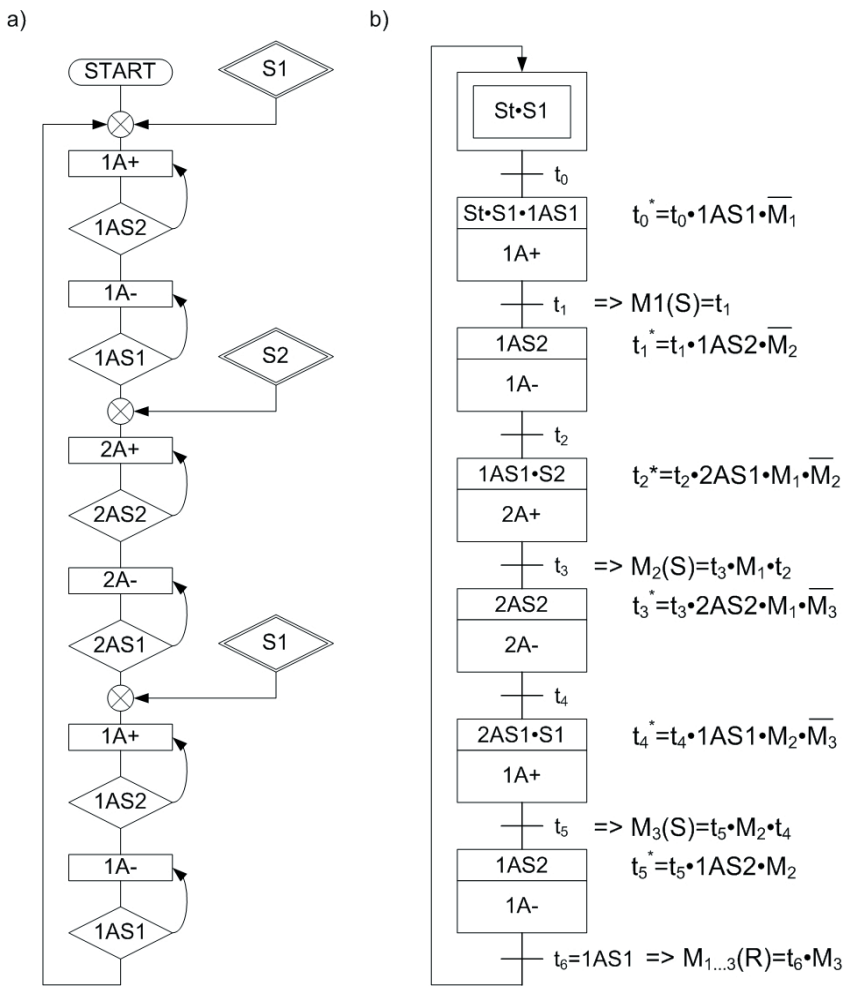
$$Y_3 = 2AS2 \cdot M_1 \cdot \overline{M_3}, \quad (27)$$

$$M_1(S) = 1AS2, \quad (28)$$

$$M_2(S) = 2AS2 \cdot M_1 \cdot 1AS1, \quad (29)$$

$$M_3(S) = 1AS2 \cdot M_2 \cdot 2AS1, \quad (30)$$

$$M_{1...3}(R) = 1AS1 \cdot M_3, \quad (31)$$



Rys. 3. Algorytm pracy urządzenia zapisany w postaci: a) sieci operacyjnej b) sieci Grafpol GP

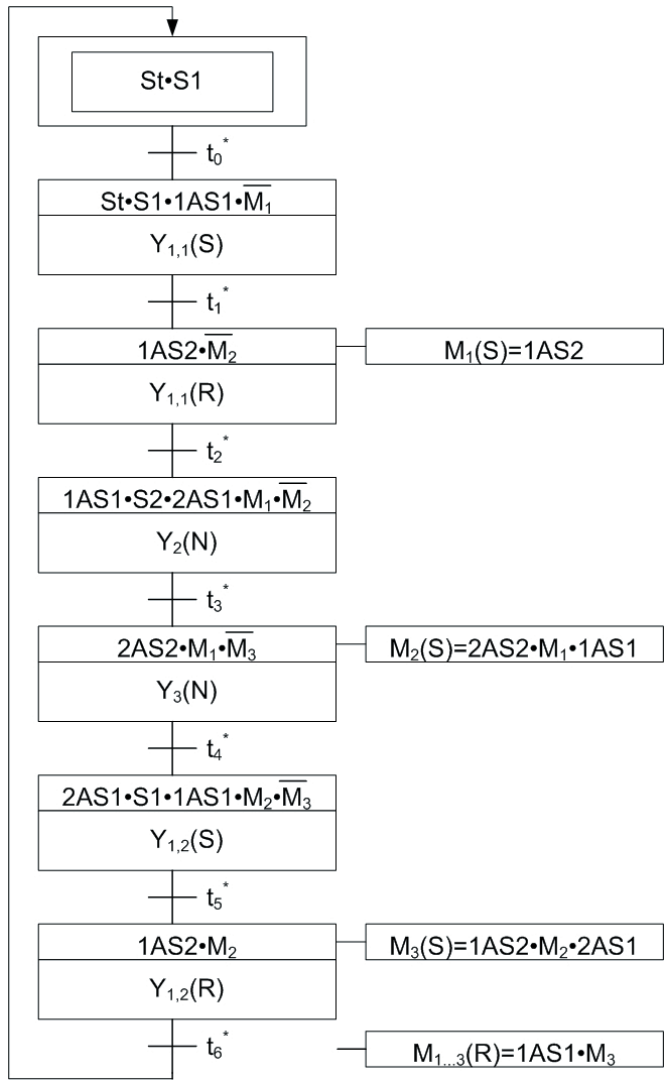
Powyższe równania stanowią podstawę do zapisu algorytmu sterowania w sterowniku PLC za pomocą języka LD. Algorytm ten przedstawiono za pomocą sieci Grafpol GS na rys. 4.
W sieci tej zastosowano kwantyfikatory działań występujące w języku SFC, tj:

N – zmienna Y_i =1 dopóki spełnione są warunki tranzycji,

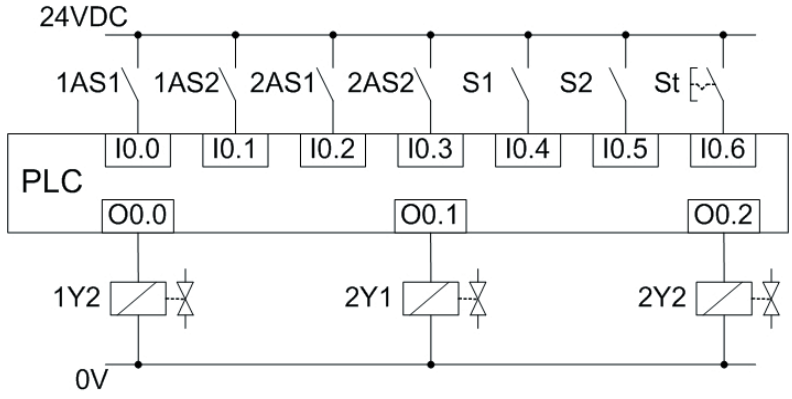
S – zmiennej Y_i lub komórce pamięci M_i przypisywana jest wartość 1 (tzw. set),

R – zmiennej Y_i lub komórce pamięci M_i przypisywana jest wartość 0 (tzw. reset).

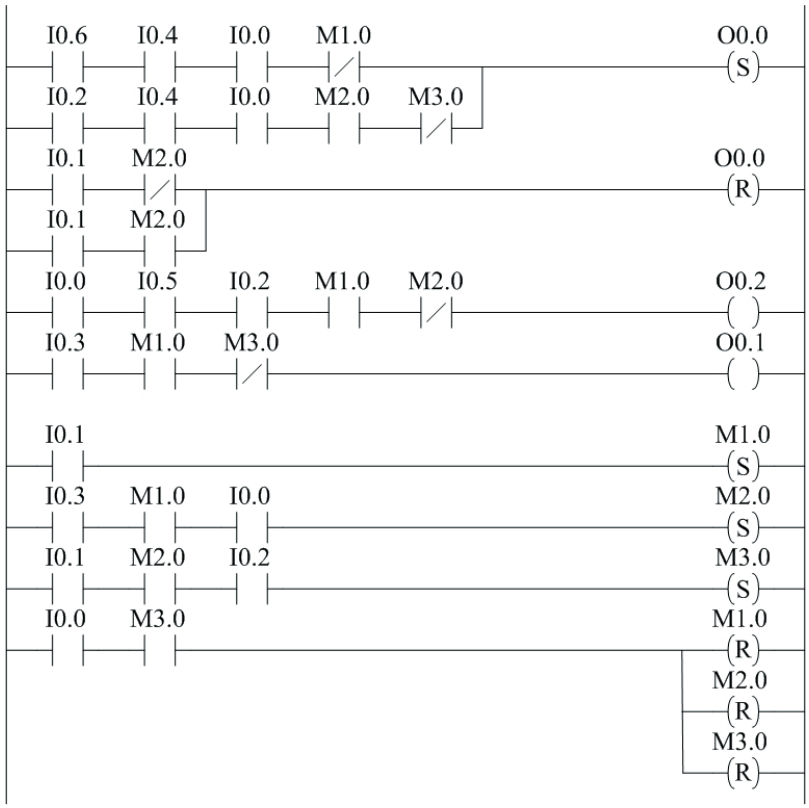
W celu zaprogramowania sterownika zgodnie z uzyskanym algorytmem sterowania niezbędne jest skojarzenie sygnałów wejściowych i wyjściowych procesu z układem sterowania, co przedstawiono na rys. 5. Na postawie tego przyporządkowania otrzymane powyżej równania można zaimplementować w postaci języka LD w sterowniku PLC (rys. 6).



Rys. 4. Opracowany algorytm sterowania przedstawiony za pomocą sieci Grafpol GS



Rys. 5. Przyporządkowanie sygnałów wejściowych i wyjściowych procesu do układu sterowania



Rys. 6. Implementacja otrzymanego algorytmu sterowania za pomocą języka LD w sterowniku PLC

4. BIBLIOGRAFIA

[1] J. Siwiński, *Układy przełączające w automatyce*, Wydawnictwa Naukowo-Techniczne, Warszawa, 1980.

[2] T. Mikulczyński, *Automatyzacja procesów produkcyjnych*, Wydawnictwa Naukowo-Techniczne, Warszawa, 2006.

[3] I. Żylińska, *Polski rynek sterowników PLC*, Control Engineering Polska, nr 8 (51) październik 2008, s. 45.