INSTYTUT TECHNOLOGII MASZYN I AUTOMATYZACJI
POLITECHNIKI WROCŁAWSKIEJ

PRACA DOKTORSKA

NOWE ZASADY SYNTEZY METODĄ GRAFPOL SEKWENCYJNYCH UKŁADÓW STEROWANIA

Łukasz Dworzak

Słowa kluczowe: **procedura sekwencyjna, procedura współbieżna, programowanie PLC, metoda Grafpol**

Promotor: prof. dr hab. inż. Tadeusz Mikulczyński

Wrocław 2012

SPIS TREŚCI

WSTĘP

W dobie światowego kryzysu ekonomicznego, oszczędnego wytwarzania (ang. *Lean Manufacturing*) oraz coraz krótszego cyklu życia produktów, jedną z cech przedsiębiorstw produkcyjnych, pozwalającą na uzyskanie przewagi konkurencyjnej, jest zdolność do szybkiego i elastycznego przezbrajania produkcji. Nierozerwalnie wiąże się to z koniecznością przeprogramowania układów sterowania maszyn produkcyjnych. To z kolei wymusza konieczność poszukiwania szybkich, niezawodnych i prostych, w codziennym stosowaniu, metod umożliwiających opracowywanie algorytmów sterowania dla nowych procesów technologicznych.

Klasyczna metoda syntezy algorytmów sterowania umożliwia realizację układów sterowania w oparciu o elektryczne elementy stykowo-przekaźnikowe lub logiczne elementy pneumatyki. W odniesieniu do powszechnie stosowanych dzisiaj mikroprocesorowych układów sterowania, których głównym reprezentantem są swobodnie programowalne sterowniki logiczne PLC (ang. *Programmable Logic Controller*), podejście to nie spełnia swoich funkcji. Dlatego na początku lat 90-tych XX wieku Tadeusz Mikulczyński podjął prace mające na celu opracowanie metody znajdującej zastosowanie w modelowaniu zautomatyzowanych procesów produkcyjnych i programowaniu sterowników PLC.

Opracowana metoda MTS [32] i Grafpol [33], jak też i inne podobne rozwiązania, charakteryzują się pewnymi niedogodnościami. Dlatego też w niniejszej pracy doktorskiej postanowiono udoskonalić metodę prof. Tadeusza Mikulczyńskiego eliminując utrudnienia i sprawiając by metoda ta stała się prostsza i użyteczniejsza w codziennym stosowaniu. W wyniku przeprowadzonych prac usprawniono metodę Grafpol, pozwalającą teraz na syntezę algorytmów sterowania w odniesieniu do procedur sekwencyjnych, współbieżnych i złożonych bez konieczności analizy graficznej przebiegu sygnałów wejściowych układu sterowania. Metoda Grafpol znajduje zastosowanie w sterowaniu napędami pneumatycznymi, hydraulicznymi i elektrycznymi. Ponadto, syntezowane przy zastosowaniu nowych zasad metody Grafpol równanie schematowe, stanowiące analityczny zapis algorytmu sterowania wraz z komórkami pamięci, może być zrealizowane za pomocą elementów pneumatycznych, układów stykowo-przekaźnikowych oraz sterowników PLC. Program użytkowy sterownika PLC może zostać zapisany za pomocą jednego dowolnego języka, określonego w normie PN-EN 61131-3.

Praca składa się z dwóch części. Pierwszą stanowią rozdziały 1-6, drugą zaś rozdziały 7-10. W rozdziale 1 skrótowo przedstawiono typy układów sterowania, w tym istotne z punktu widzenia pracy układy sekwencyjne. Zaprezentowano także różnice w sterowaniu napędami elektrycznymi, pneumatycznymi i hydraulicznymi, wyjaśniając pojęcia monostabilności oraz bistabilności. Rozdział 2 zawiera opis Metody Transformacji Sieci (MTS) służącej syntezie równań schamatowych i projektowaniu pamięci w sekwencyjnych układach sterowania, jej zalety i wady. MTS stosowana jest w metodzie Grafpol opisanej w rozdziale 4. W rozdziale 3 przedstawiono najpowszechniej znane, niemal bliźniacze, metody Grafcet i SFC. Ze względu na fakt, iż obecnie układy sterowania najczęściej realizowane są za pomocą sterowników PLC, w rozdziale 5 scharakteryzowano sterowniki pod względem budowy oraz zaprezentowano cykl przetwarzania informacji w sterowniku. Uzupełnienie rozdziału 5 stanowi rozdział 6, w którym przedstawiono składnię języków programowania sterowników PLC według normy PN-EN 61131-3. Opisy zawarte w rozdziałach 1-6 mają na celu skrótowe zaprezentowanie dotychczasowych osiągnięć w dziedzinie syntezy algorytmów sterowania i stanowią wprowadzenie do drugiej, zasadniczej, części niniejszej pracy.

W rozdziale 7 zdefiniowano założenia programowe pracy, określając jej cel i tezy. W rozdziale 8, stanowiącym najistotniejszą część pracy, zaprezentowano opracowane nowe zasady realizacji pamięci w metodzie Grafpol. Poprawność opracowanych zasad zweryfikowano na przykładowych algorytmach procesu przedstawionych w rozdziale 9.

1. WYBRANE UKŁADY AUTOMATYKI

Zagadnienia modelowania procesów dyskretnych oraz programowania układów sterowania nie są ze sobą bezwzględnie powiązane. Za każdym razem można bowiem wyszczególnić proces tworzenia modelu na podstawie którego realizowana jest pamięć oraz syntezowane jest równanie schematowe i implementację opracowanego równania w układzie sterowania. Choć nie jest to połączenie bezwzględne, to jednak proces modelowania zależy choćby od możliwości układu sterowania w którym chcemy zaimplementować równanie. Inaczej sytuacja wygląda w przypadku procesu modelowania. Ze względu na fakt iż model jest odwzorowaniem procesu, z góry narzucone są pewne jego elementy, jak choćby elementy wykonawcze.

1.1. Typy układów sterowania

Układy sterowania automatycznego podzielić można na układy o sygnale sterującym ciągłym (analogowym) i o sygnale sterującym dyskretnym (cyfrowym).

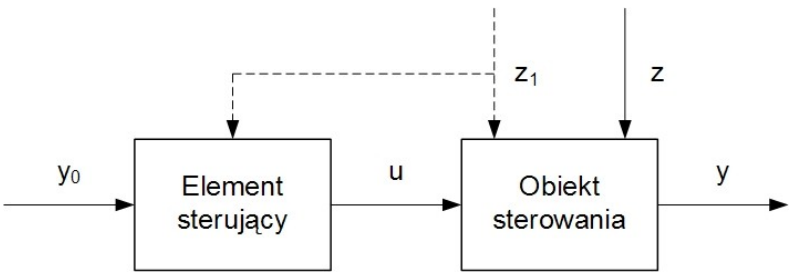
W odniesieniu do układów o sygnale sterującym ciągłym wyszczególnić można dwa sposoby sterowania – w układzie otwartym i w układzie zamkniętym [27].

W otwartym układzie sterowania (rys. 1.1) element sterujący steruje zmienną wyjściową u na podstawie wartości zadanej y_0 i pod wpływem tego samego zakłócenia z_1 oddziałującego na obiekt sterowania i element sterujący. Ze względu na nieznane zakłócenie z oddziałujące tylko na obiekt sterowania wartość zmiennej wyjściowej y może być różna w czasie, względem założeń projektowych, o czym element sterujący nie ma informacji.

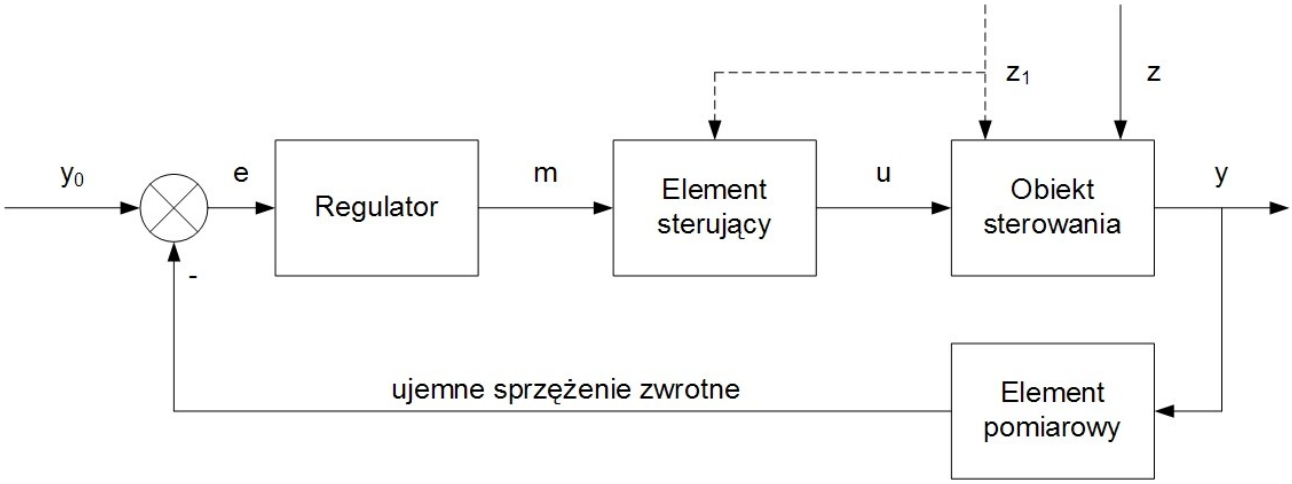
W przypadku zamkniętego układu sterowania (rys. 1.2) wartość zmiennej wyjściowej y jest mierzona przez element pomiarowy i po porównaniu z wartością zadaną y_0 otrzymujemy uchyb regulacji e. Wartość uchybu podawana jest do regulatora, który steruje zmienną m oddziałującą na element sterujący w taki sposób by uchyb regulacji e dążył do zera.

Choć wydawać by się mogło, że w XXI wieku otwarte układy sterowania nie mają racji bytu to w praktyce inżynierskiej tak nie jest. Układy tego typu znajdują zastosowanie w obiektach gdzie nie ma wysokich wymagań jakościowych odnośnie co do wartości sygnału wyjściowego, a zakłócenia są niewielkie lub nie mają znaczącego wpływu na element sterujący i obiekt. Tego typu rozwiązania charakteryzują się ponadto prostotą wykonania, a co się z tym wiąże niższym kosztem.

Podobnie jak w przypadku układów o ciągłym sygnale sterującym układy o dyskretnym sygnale sterującym mogą być realizowane również jako układy otwarte i zamknięte (z tzw. pętlą sprzężenia zwrotnego). Układy dyskretne, zwane również cyfrowe, realizowane są jako układy przełączające i znajdują zastosowanie w sterowaniu procesami przemysłowymi, przetwarzając otrzymane informacje dwustanowe za pomocą funkcji logicznych. W przeważającej mierze występują one jako zamknięte układy sterowania. Jest to konieczne, ponieważ większość z tych procesów to procedury sekwencyjne, w których kolejne etapy realizowane są po wykonaniu etapów poprzedzających. Niezbędna jest więc znajomość stanu w jakim znajduje się urządzenie, a więc dysponowanie informacją zwrotną.



Rys. 1.1. Otwarty układ automatycznego sterowania analogowego [27]



Rys. 1.2. Zamknięty układ automatycznego sterowania analogowego [27]

1.2. Cyfrowe układy automatyki

W praktyce przemysłowej w przeważającej mierze mamy do czynienia z sygnałami cyfrowymi. Układy w których występują tego typu sygnały realizowane są poprzez układy przełączające, tj. układy stykowo-przekaźnikowe, układy oparte o elementy pneumatyczne czy sterowniki PLC. Niezależnie od sposobu realizacji układu sterowania każdorazowo do opisu algorytmu sterowania zastosowanie znajduje algebra Boole'a.

Algebra Boole'a opisuje funkcjonowanie maszyn i urządzeń sterowanych przez układy przełączające które, ze względu na charakter ich pracy, podzielić można na układy kombinacyjne i sekwencyjne.

1.2.1. Układy kombinacyjne

Układy kombinacyjne to układy w których aktualny stan wyjść zależny jest jedynie od aktualnego stanu wejść. Opisane mogą być więc zależnością:

$$Y^t = f(X^t) \tag{1.1}$$

gdzie:
Y^t – stan wyjść układu w chwili t,
X^t – stan wejść układu w chwili t.

1.2.2. Układy sekwencyjne

Układy sekwencyjne [27, 35] to układy w których aktualny stan elementów wyjściowych zależy od aktualnego stanu elementów wejściowych oraz od stanu elementów wejściowych w chwilach poprzednich. Opisuje je zależność:

$$Y^t = f(X^t, X^{t-1}, X^{t-2}, ...) \tag{1.2}$$

gdzie:
Y^t – stan wyjść układu w chwili t,
X^t – stan wejść układu w chwili t,
X^{t-1} – stan wejść układu w chwili t-1,...

Układy takie charakteryzują się więc pamięcią o tym co się już zdarzyło. Pamięć ta realizowana może być w sposób sprzętowy lub programowy.

Sprzętowa realizacja pamięci polega na zastosowaniu elementów wykonawczych, które posiadają pamięć w sobie. Elementami takimi są np. pneumatyczne zawory bistabilne.

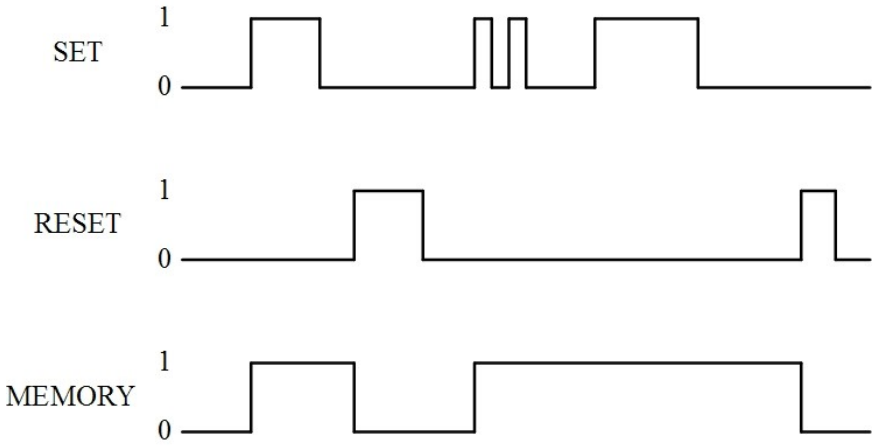
W przypadku programowej realizacji pamięci, np. za pomocą sterownika PLC, typ elementów wykonawczych nie ma znaczenia gdyż pamięć realizowana jest w obszarze układu sterowania. Programowa realizacja pamięci może przebiegać w dwojaki sposób. Pierwszym jest użycie zdefiniowanych komórek pamięci, najczęściej M (ang. *memory*) lub F (ang. *flag*), które charakteryzuje możliwość zapisywania (ang. *Set*) i kasowania (ang. *Reset*). Charakterystykę stanu logicznego komórki pamięci, w zależności od zapisu i kasowania, przedstawiono na rys.1.3. Drugim sposobem jest zastosowanie pętli podtrzymania, a więc taki dobór warunków zapisu i kasowania oraz odpowiednie umieszczenie warunku logicznego używanej komórki pamięci, by była ona w stanie logicznym '1' tylko w ściśle określonych krokach algorytmu sterowania.

W odniesieniu do sterowników PLC zastosowane mogą być dwie opisane metody, jednak ze względu na prostotę w większości przypadków używana jest pierwsza z nich.

W odniesieniu do układów sekwencyjnych wyróżnić można trzy typy struktur sterowania:

- ze sprzężeniem zwrotnym (ang. *feedback*) – rys. 1.4a,
- w układzie otwartym z bezpośrednim pomiarem zakłóceń – rys. 1.4b,
- w układzie otwartym z pośrednim pomiarem zakłóceń – rys. 1.4c.

Ze względu na prostotę oraz koszty w większości urządzeń spotykany jest pierwszy z przedstawionych typów struktury sterowania.



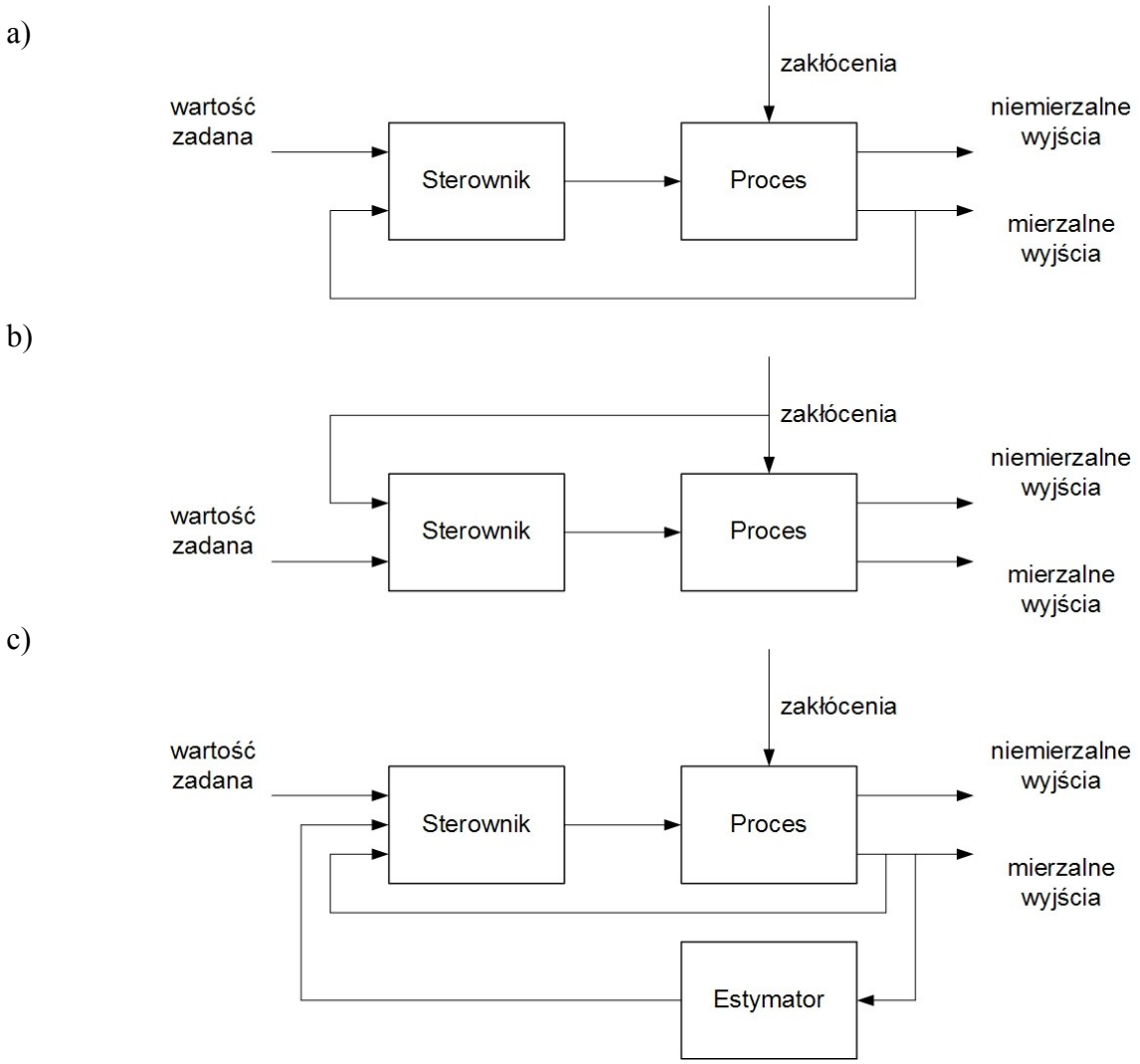
Rys. 1.3. Stan logiczny komórki pamięci w zależności od zapisu i kasowania

Z automatycznymi układami sekwencyjnymi związane są pewne pojęcia wyjaśnione poniżej. Pojęcia te przytoczono dla lepszego zrozumienia zalet opracowanych nowych zasad realizacji pamięci w metodzie Grafpol. Pierwszym pojęciem jest takt, czyli okres w którym w układzie nie zachodzi żadna zmiana. W odniesieniu do układów kombinacyjnych takt należy utożsamiać z kombinacją, zaś w przypadku układów sekwencyjnych z etapem.

Sekwencyjne układy sterowania, w zależności od sposobu realizacji, podzielić możemy na układy synchroniczne i układy asynchroniczne.

Układ synchroniczny to układ w którym sygnały z elementów wejściowych przechodzą z jednego stanu do drugiego pod wpływem przyczyny wywołującej tą zmianę tylko w chwili zmiany sygnału synchronizującego. Wartość opóźnienia zmiany sygnału wyjściowego czujnika względem zmiany sygnału wejściowego układu sterowania związana jest więc tylko z częstotliwością sygnału synchronizującego.

Układ asynchroniczny to układ w którym elementy przechodzą z jednego stanu do drugiego z opóźnieniem niezależnym od czynników zewnętrznych. Opóźnienie zmiany stanu elementu układu względem przyczyny wywołującej tą zmianę wynika jedynie z właściwości tego elementu.



Rys. 1.4. Podstawowe struktury cyfrowych układów sterowania: a) ze sprzężeniem zwrotnym, b) w układzie otwartym z bezpośrednim pomiarem zakłóceń, c) w układzie otwartym z pośrednim pomiarem zakłóceń

W układach sekwencyjnych zrealizowanych w oparciu o przekaźniki lub elementy pneumatyczne często występują niekorzystne zjawiska hazardu i wyścigu.

Hazard jest zjawiskiem polegającym na niepoprawnym funkcjonowaniu układu sterowania zbudowanego poprawnie pod względem logicznym. Spowodowane jest to niedokładnością wykonania zastosowanych do budowy układu sterowania elementów, np. przekaźników, lub opóźnieniami związanymi z przepływem sygnału jakim jest powietrze. W układach sterowania opartych na sterownikach PLC zjawisko to nie występuje.

Wyścig to zjawisko polegające na każdorazowym przechodzeniu układu z identycznego stanu początkowego do stanu końcowego poprzez odmienne stany pośrednie. Zjawisko to spowodowane jest niejednoczesną zmianą stanu zastosowanych w układzie sterowania elementów pamięci. W zagadnieniach związanych z wyścigiem wyszczególnić możemy:

- wyścig niekrytyczny, gdy układ po przejściu poprzez odmienne stany zawsze dochodzi do tego samego stanu końcowego,
- wyścig krytyczny, gdy układ po przejściu poprzez odmienne stany zawsze dochodzi do innego stanu końcowego.

1.3. Sterowanie napędami elektrycznymi, pneumatycznymi i hydraulicznymi

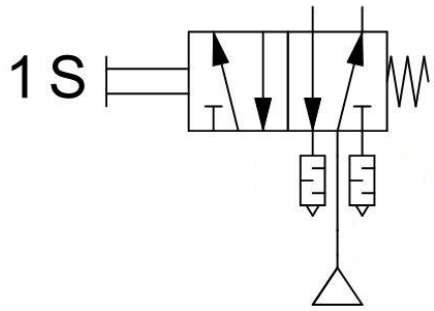
W maszynach przemysłowych obecnie najpowszechniej spotykane są napędy elektryczne, pneumatyczne i hydrauliczne. Elementy sterujące napędami pneumatycznymi i hydraulicznymi mogą być sterowane na dwa sposoby - monostabilnie lub bistabilnie, zaś napędy elektryczne monostabilnie. W odniesieniu do napędów pneumatycznych i hydraulicznych sposób sterowania należy utożsamiać z zaworami rozdzielającymi, a w przypadku napędów elektrycznych z przekaźnikami i stycznikami.

1.3.1. Sterowanie monostabilne

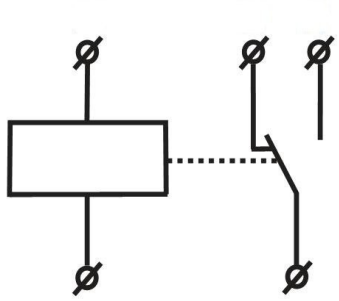
Monostabilny, jak wskazuje sama nazwa, oznacza stabilny w tylko jednym położeniu. Elementem składowym układów monostabilnych jest sprężyna lub element sprężysty powodujący powrót układu do pozycji ustalonej po zaniku wymuszenia. Wymuszenie może być m.in. typu:

- mechanicznego (przycisk, rolka),
- elektrycznego (podanie prądu),
- cieczowego (podanie sprężonego powietrza lub oleju pod ciśnieniem),
- termicznego (zmiana temperatury).

W odniesieniu do układów cieczowych (pneumatyka, hydraulika) monostabilnym elementem sterującym jest zawór (rys. 1.5), zaś w przypadku napędów elektrycznych stycznik lub przekaźnik (rys. 1.6).

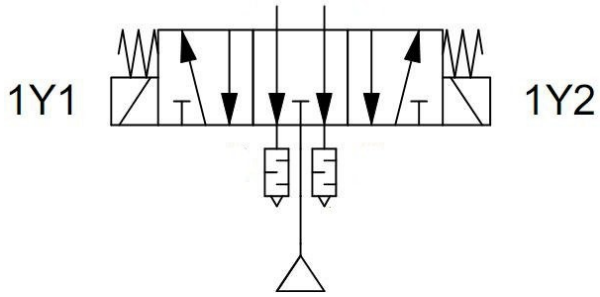


Rys. 1.5. Zawór pneumatyczny monostabilny sterowany ręcznie



Rys. 1.6. Schemat elektryczny przekaźnika monostabilnego

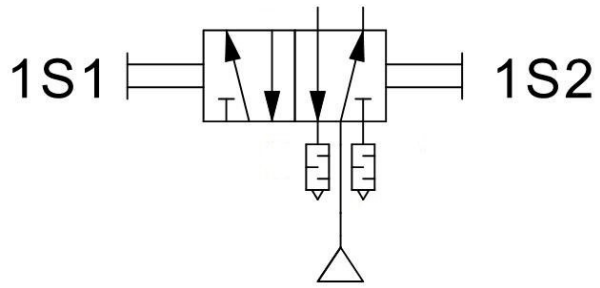
Specyficznymi zaworami monostabilnymi są zawory o trzech (rys. 1.7) położeniach. W zaworach tych występują dwa elementy wymuszające (mechaniczne, pneumatyczne lub elektryczne) oraz sprężyna lub sprężyny. W zależności od tego na którą cewkę zostanie podany prąd następuje przesterowanie zaworu. Sprężyna ma za zadanie jednoznacznie pozycjonować suwak zaworu w pozycji środkowej po zaniku wymuszenia.



Rys. 1.7. Zawór pneumatyczny trójpołożeniowy sterowany elektrycznie

1.3.2. Sterowanie bistabilne

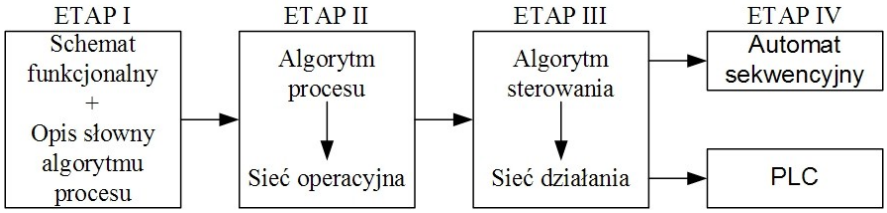
Bistabilny oznacza stabilny w dwóch położeniach. W elementach tego typu nie występują sprężyny, zaś zanik sygnału wymuszającego nie powoduje powrotu elementu do ściśle określonej pozycji wyjściowej (pozostaje on w ostatniej pozycji). Ponowne przesterowanie elementu możliwe jest jedynie w przypadku podania sygnału wymuszającego na drugie wejście elementu pod warunkiem, że nastąpił zanik pierwszego sygnału wymuszającego. Właściwość ta sprawia, że elementy tego typu często określane są mianem elementów z pamięcią. W układach cieczowych (pneumatyka, hydraulika) bistabilnym elementem sterującym jest zawór (rys. 1.8).



Rys. 1.8. Zawór pneumatyczny bistabilny sterowany ręcznie

2. METODA TRANSFORMACJI SIECI

Metoda Transformacji Sieci, w skrócie MTS, opracowana została na początku lat 90-tych XX wieku. Prezentuje ona sposób postępowania w trakcie opracowywania algorytmu sterowania, opisuje sposób odwzorowania zbioru sieci operacyjnej zbiorem sieci działania oraz pozwala na wyznaczenie postaci funkcji sygnałów wyjściowych układów sekwencyjnych. Synteza sekwencyjnego układu sterowania w metodzie MTS realizowana jest w czterech etapach przedstawionych na rys. 2.1. Szczegółowo metoda MTS opisana została w [33, 34]. W niniejszej pracy przytoczone zostaną najistotniejsze informacje.



Rys. 2.1. Etapy syntezy metodą MTS sekwencyjnych układów sterowania [34]

2.1. Schemat funkcjonalny i opis słowny algorytmu procesu

Schemat funkcjonalny procesu powinien przedstawiać ten proces w stanie początkowym. Istotne jest by zawierał wszystkie elementy lub zespoły wykonawcze oraz sygnalizacyjne poszczególnych etapów. Sam schemat funkcjonalny jest niewystarczający dlatego też uzupełnia się go opisem słownym realizacji poszczególnych etapów. Na rysunku 2.2. przedstawiono schemat funkcjonalny procesu dozowania materiału sypkiego. Opis słowny procesu jest następujący:

ETAP Z_1:: *dozowanie materiału*
Realizacja: S_1^+ (EZ_1^+)
Sygnalizacja: WP2=1

ETAP Z_2: *uzupełnianie dozownika*
Realizacja: S_1^- (EZ_2^+)
Sygnalizacja: WP1=1

W przytoczonym przykładzie użyte symbole oznaczają:

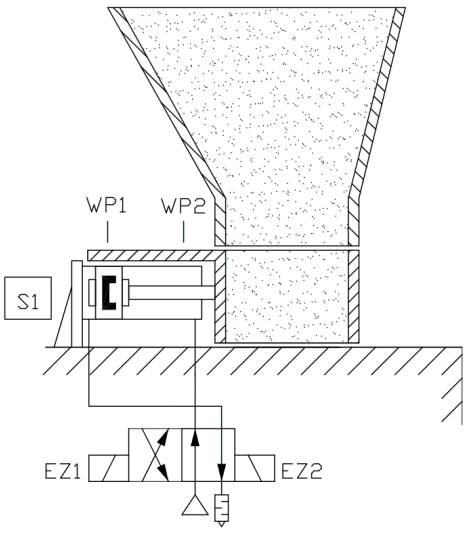
- S_1^+ - wysuw tłoczyska siłownika S1,
- S_1^- - wsuw tłoczyska siłownika S1,
- EZ_1^+ - stan prądowy cewki zaworu rozdzielającego,
- EZ_1^- - stan bezprądowy zaworu rozdzielającego.

2.2. Model matematyczny algorytmu procesu

Na podstawie schematu funkcjonalnego oraz opisu słownego budowany jest model matematyczny algorytmu dyskretnego procesu produkcyjnego reprezentowany przez sieć operacyjną. Siecią operacyjną nazywamy trójkę:

$$SO=<E, W, O> \tag{2.1}$$

gdzie:
$E=\{e_1, e_2, ..., e_e\}$ – skończony, niepusty zbiór etapów elementarnych procesu,
$W=\{w_1, w_2, ..., w_w\}$ – skończony, niepusty zbiór warunków logicznych określających realizację etapów elementarnych,
$O=\{o_1, o_2, ..., o_o\}$ – skończony zbiór węzłów operacji alternatywy i/lub koniunkcji.



Rys. 2.2. Schemat funkcjonalny procesu dozowania materiału sypkiego [34]

Do budowy sieci operacyjnej wykorzystywane są następujące symbole:

- Klatka operacyjna (rys. 2.3a), reprezentuje etap elementarny. Wewnątrz symbolu umieszcza się nazwę symboliczną etapu.
- Klatka warunkowa (rys. 2.3b), służy do reprezentacji warunków określających zakończenie realizacji poszczególnych etapów elementarnych procesu oraz warunków, których stan zależy od realizacji procesu. Klatki tego typu mogą przedstawiać warunki logiczne lub czasowe. Każda klatka warunkowa ma jedno wejście WE i dwa wyjścia TAK i NIE, które zgodnie z algebrą Boole'a przyjmują wartości odpowiednio 1 i 0.
- Klatka warunkowa niezależna (rys. 2.3c), służy do reprezentacji warunków niezwiązanych bezpośrednio z realizacją poszczególnych etapów procesu, lecz związanych z zewnętrznymi warunkami określającymi realizację określonego procesu. Tego typu klatka ma tylko wyjścia TAK lub NIE.
- Węzły alternatywny (rys. 2.3d) i koniunkcji (rys. 2.3e), służą do przedstawiania logicznych operacji alternatywy i koniunkcji, których spełnienie dla danych warunków logicznych jest konieczne i wystarczające, aby realizacja poszczególnych etapów elementarnych procesu przebiegała zgodnie z założonym algorytmem.
- Rozgałęzienie sygnału (rys. 2.3f), oznacza że sygnały za węzłem mają takie same wartości jak sygnał przed węzłem.
- Klatka START (rys. 2.3g), reprezentuje stan procesu w momencie rozpoczęcia jego realizacji.
- Klatka STOP (rys. 2.3h), przedstawia stan procesu w momencie zakończenia jego realizacji.

Algorytm procesu dozowania materiału sypkiego przedstawiono na rys. 2.4.

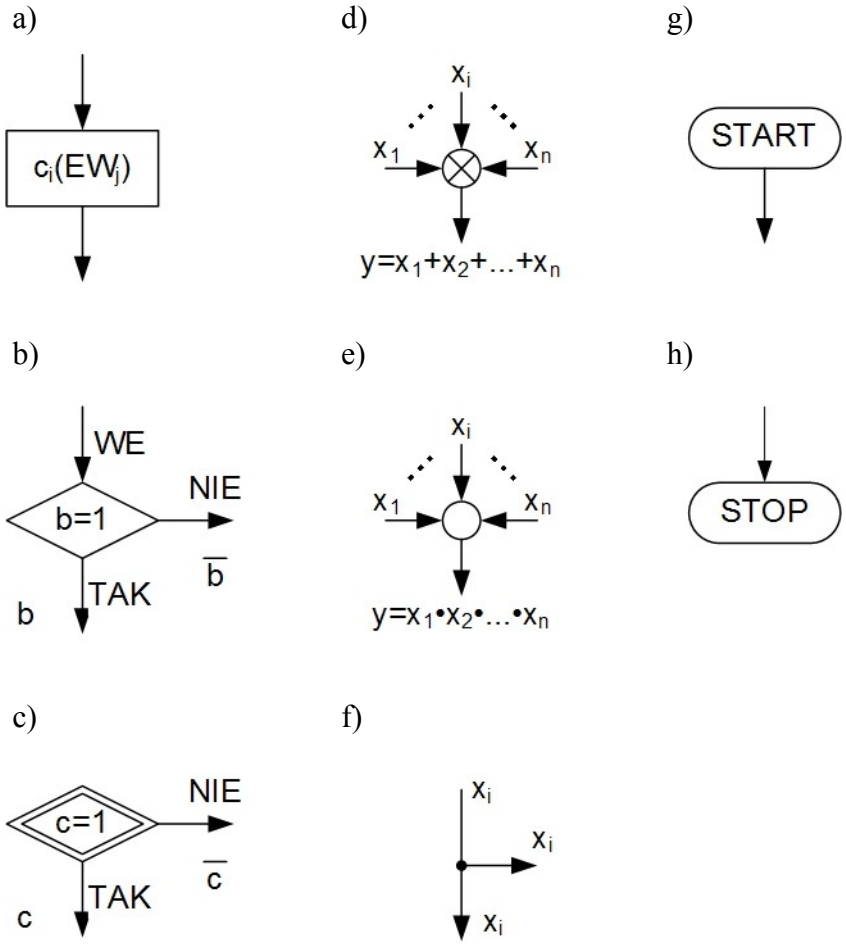
Na podstawie sieci operacyjnej możliwe jest wyznaczenie warunków realizacji etapów elementarnych, które opisuje zależność:

$$F(E_i) = f_{pi} \cdot f_{ki} \tag{2.2}$$

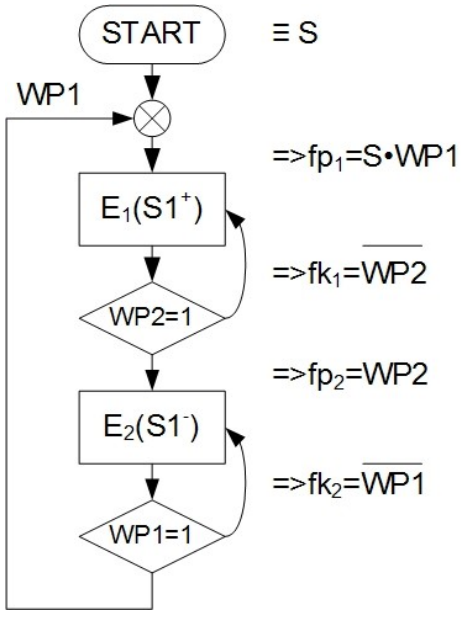
gdzie:

f_{pi} – zależność logiczna, określająca stan procesu w którym następuje rozpoczęcie realizacji i-tego etapu,

f_{ki} – zależność logiczna, określająca stan procesu w którym następuje zakończenie realizacji i-tego etapu,



Rys. 2.3. Symbole graficzne elementów sieci operacyjnej: a) klatka operacyjna, b) klatka warunkowa, c) klatka warunkowa niezależna, d) węzeł alternatywy, e) węzeł koniunkcji, f) rozgałęzienie sygnału, g) klatka START, h) klatka STOP



Rys. 2.4. Algorytm procesu dozowania materiału sypkiego

2.3. Algorytm sterowania

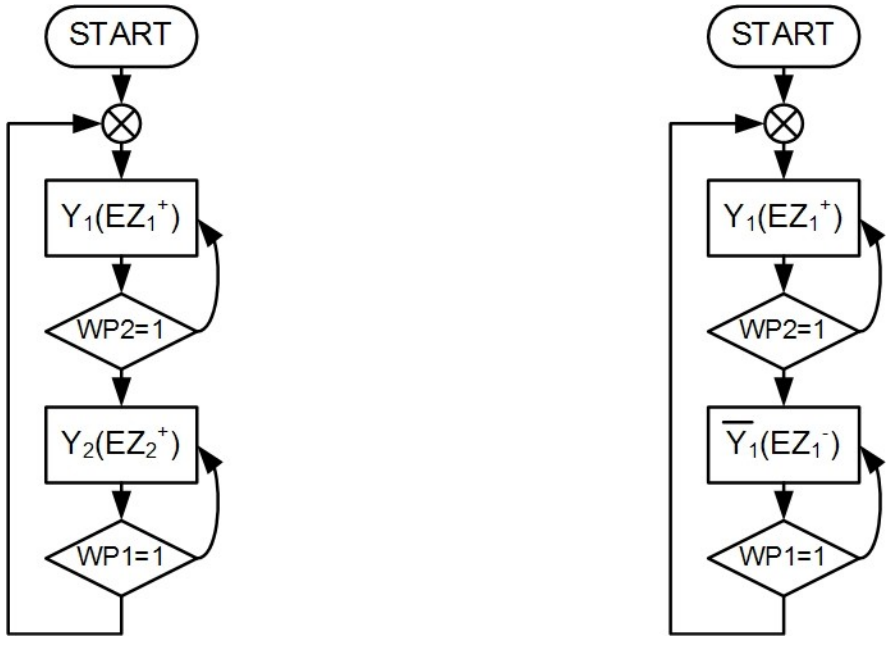
Algorytm sterowania, reprezentowany przez sieć działania, otrzymuje się w wyniku transformacji algorytmu procesu. Transformacja ta polega na odwzorowaniu zbioru etapów elementarnych zbiorem zmiennych wyjściowych układu sterowania. W zależności od zastosowanego typu elementu sterującego (w przypadku pneumatyki i hydrauliki zaworu rozdzielającego, w przypadku silnika elektrycznego stycznika) lub charakteru pracy elementu (obroty wału silnika w jednym lub obydwu kierunkach) przyporządkowywana jest liczba zmiennych wyjściowych wg zasady:

- monostabilnym elementom sterującym lub pracy silnika elektrycznego tylko w jednym kierunku jedna zmienna sterująca,
- bistabilnym elementom sterującym lub pracy silnika elektrycznego w obydwu kierunkach dwie zmienne sterujące.

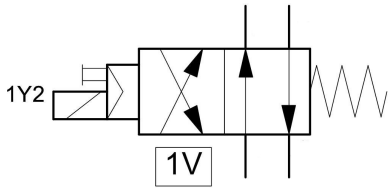
Tak otrzymana sieć działania reprezentuje zewnętrzne sygnału układu sterowania. Sieć ta stanowi podstawę do realizacji pamięci oraz syntezy równania schematowego. Równanie schematowe jest analitycznym ujęciem kompletnego algorytmu sterowania z uwzględnieniem pamięci.

Dla przykładowego procesu dozowania materiału sypkiego za pomocą siłownika sterowanego zaworem bistabilnym sieć działania przedstawiono na rys. 2.5a. Na rysunku 2.5b przedstawiono sieć działania reprezentującą proces dozowania materiału sypkiego, w której siłownik pneumatyczny jest sterowany zaworem monostabilnym (rys. 2.6).

a) b)



Rys. 2.5. Sieć działania dla zaworu: a) bistabilnego, b) monostabilnego



Rys. 2.6. Schemat zaworu monostabilnego

2.4. Realizacja pamięci i synteza równania schematowego

Metoda Transformacji Sieci pozwala na wyznaczenie postaci funkcji zmiennych sterujących w których uwzględniona jest pamięć. Pamięć ta zrealizowana może być poprzez stosowanie pętli logicznych sprzężeń zwrotnych oraz elementarnych komórek pamięci.

Metoda MTS bazuje na graficznej reprezentacji sygnałów:

- f_{pi} – warunek logiczny określający stan procesu, w którym następuje rozpoczęcie realizacji etapu E_i,
- f_{ki} – warunek logiczny określający stan procesu, w którym następuje zakończenie wykonywania etapu E_i,

oraz analizie porównawczej przebiegu sygnału $f_{pi} \cdot f_{ki}$ z Y_i. Wielkość Y_i jest zmienną sterującą, a więc sygnałem wyjściowym układu sterowania.

Realizacja procesu przebiega zgodnie z założonym algorytmem gdy spełniona jest zależność (2.3) w stanach procesu odpowiadających realizacji etapu E_i oraz (2.4) we wszystkich pozostałych stanach procesu.

$$f_{pi} \cdot f_{ki} = 1 \tag{2.3}$$

$$f_{pi} \cdot f_{ki} = 0 \tag{2.4}$$

Jeśli zależności z równań (2.3) i (2.4) nie są spełnione we właściwych stanach procesu należy zastosować poniżej przedstawione reguły realizacji pamięci metody MTS. W wyniku zastosowania tych reguł wyznaczana jest postać zależności f^*_{pi} i f^*_{ki}, w których została uwzględniona pamięć.

Reguła 1

Jeśli f_{pi}=1 w stanie układu sterowania, w którym zmienna wyjściowa Y_i przyjmuje wartość logiczną 1 lub tylko w niektórych stanach w których zmienna wyjściowa Y_i ma wartość logiczną 1, to stosuje się pętlę logicznego sprzężenia zwrotnego i wówczas zależności f^*_{pi} i f^*_{pi} mają postać:

$$\begin{aligned} f^*_{pi} &= f_{pi} + y_i \\ f^*_{ki} &= f_{ki} \end{aligned} \tag{2.5}$$

gdzie:
y_i – sygnał wyjściowy zmiennej wyjściowej Y_i.

Reguła 2

Przypadek A

Jeśli f_{ki}=0 w stanie układu sterowania, w którym zmienna wyjściowa Y_i zmienia wartość logiczną z 0 na 1 i w części stanów, w których zmienna wyjściowa Y_i ma wartość logiczną 1, to stosuje się pętlę sprzężenia zwrotnego i wówczas zależności f^*_{pi} i f^*_{ki} mają postać:

$$\begin{aligned} f^*_{pi} &= f_{pi} \\ f^*_{ki} &= f_{ki} + \overline{y_k} \end{aligned} \tag{2.6}$$

gdzie:
y_k – sygnał k-tej zmiennej wyjściowej.

Zmienna wyjściowa Y_k powinna spełniać następujące warunki:

- mieć wartość logiczną 0 w stanach układu sterowania, w których funkcja f_{ki} powinna mieć wartość logiczną 1, a nie ma.
- mieć wartość logiczną 1 w stanie układu sterowania, w którym zmienna Y_i zmienia wartość logiczną z 1 na 0.

Jeśli w algorytmie sterowania nie ma zmiennej wyjściowej spełniającej powyższe warunki, to do realizacji pamięci należy zastosować regułę określoną w przypadku B.

Przypadek B

Jeśli $f_{ki}=0$ w stanie układu sterowania, w którym zmienna wyjściowa Y_i zmienia wartość logiczną z 0 na 1 i w części stanów, w których zmienna wyjściowa Y_i ma wartość logiczną 1, to stosuje się pętlę logicznego sprzężenia zwrotnego i wówczas zależności f^*_{pi} i f^*_{ki} mają postać:

$$\begin{aligned} f^*_{pi} &= f_{pi} \\ f^*_{ki} &= f_{ki} + \overline{m} \end{aligned} \tag{2.7}$$

gdzie:

m – sygnał wyjściowy elementarnej komórki pamięci.

Warunki zapisu i kasowania elementarnej komórki pamięci są następujące:

- zapis elementarnej komórki pamięci powinien nastąpić w jednym ze stanów układu sterowania, w których f_{ki} ma wartość logiczną 1,
- kasowanie elementarnej komórki pamięci powinno nastąpić w jednym ze stanów układu sterowania występujących po stanie, w którym f_{ki} zmieniła wartość logiczną z 1 na 0.

Reguła 3

Jeśli $f_{pi} \cdot f_{ki} = 1$ w stanach układu sterowania występujących przed stanem, w którym zmienna wyjściowa Y_i zmienia wartość logiczną z 0 na 1, to stosuje się elementarną komórkę pamięci i wówczas zależności f^*_{pi} i f^*_{ki} przyjmują postać:

$$\begin{aligned} f^*_{pi} &= f_{pi} \cdot m_j \\ f^*_{ki} &\equiv f_{ki} \end{aligned} \tag{2.8}$$

przy czym, jeśli $f^*_{pi} \equiv Y_i$, to $f^*_{ki} \equiv 1$

gdzie:

mj – sygnał wyjściowy elementarnej komórki pamięci Mj.

Warunki zapisu i kasowania elementarnej komórki pamięci są następujące:

- zapis elementarnej komórki pamięci powinien nastąpić w jednym ze stanów układu sterowania, w których zależność $f_{pi} \cdot f_{ki}$ ma wartość logiczną 0. Powinien to być stan występujący przed stanem, w którym zmienna Y_i zmienia wartość logiczną z 0 na 1,
- kasowanie elementarnej komórki pamięci powinno nastąpić w stanie, w którym zmienna Y_i zmienia wartość logiczną z 1 na 0 lub w jednym z następnych stanów. Powinien to być stan, w którym zależność $f_{pi} \cdot f_{ki}$ ma wartość logiczną 0.

Reguła 4

Jeśli $f_{pi} \cdot f_{ki} = 1$ w stanach układu sterowania występujących po stanie, w którym zmienna wyjściowa Y_i zmienia wartość logiczną z 1 na 0, to stosuje się elementarną komórkę pamięci i wówczas zależności f^*_{pi} i f^*_{ki} przyjmują postać:

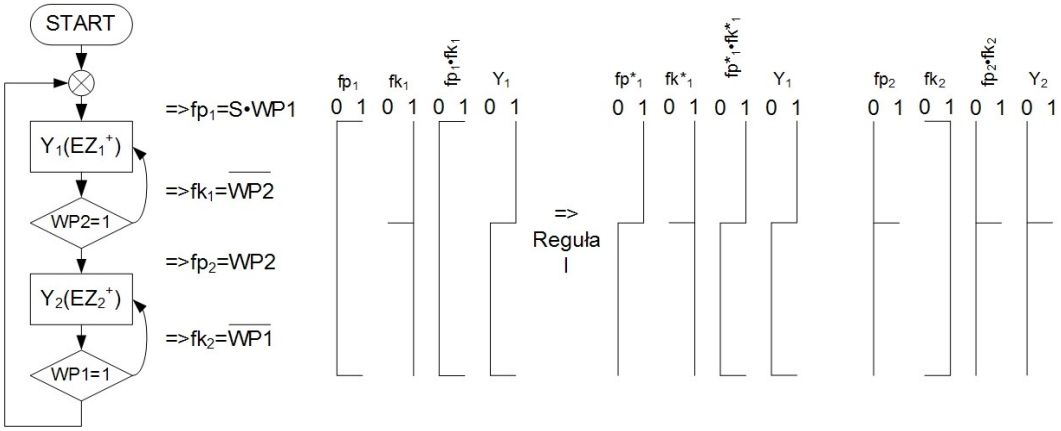
$$f^*_{pi} \equiv f_{pi} \quad \text{lub} \quad f^*_{pi} = f_{pi} + y_i \qquad f^*_{ki} = \overline{m_j} \tag{2.9}$$

gdzie:
m_j – sygnał wyjściowy elementarnej komórki pamięci.

Warunki zapisu i kasowania elementarnej komórki pamięci są następujące:

- zapis elementarnej komórki pamięci powinien nastąpić w stanie w którym zmienna wyjściowa Y_i zmienia wartość logiczną z 1 na 0,
- kasowanie elementarnej komórki pamięci powinno nastąpić w jednym z ostatnich stanów układu sterowania, w których zależność $f_{pi} \cdot f_{ki}$ ma wartość logiczną 0.

a) b)



Rys. 2.7. Algorytm sterowania przykładowym procesem dozowania materiału sypkiego: a) sieć działania, b) realizacja pamięci

Dla przykładowego przypadku dozowania materiału sypkiego sposób realizacji pamięci wg powyższych reguł przedstawiono na rys. 2.7. W rezultacie otrzymano równanie schematowe następującej postaci:

$$F(Y) = \sum \begin{cases} S \cdot (WP1 + y_1) \cdot \overline{WP2} \cdot \boldsymbol{Y_1}, \\ WP2 \cdot \overline{WP1} \cdot \boldsymbol{Y_2}, \end{cases} \tag{2.10}$$

2.5. Zalety i wady metody MTS

Do zalet Metody Transformacji Sieci należy użycie minimalnej liczby elementarnych komórek pamięci potrzebnych do wyznaczenia równania schematowego. Wynika to z faktu, iż bardzo często możliwe jest zastosowanie sprzężeń zwrotnych, pozwalających zaoszczędzić komórki pamięci. Ponadto, w przypadku niemożności użycia sygnału sprzężenia zwrotnego stosowane są tylko niezbędne komórki pamięci (nie występuje zjawisko nadmiarowości).

Niestety metoda ma również wady. Główną wadą jest pracochłonność i konieczność graficzno-analitycznej analizy wykresów przebiegu sygnałów. Liczba sygnałów, których przebiegi wymagają przeanalizowania jest co najmniej równa dwukrotności liczby etapów poszczególnego algorytmu. Dodatkowy wzrost liczby sygnałów wynika z konieczności określenia warunków zapisu i kasowania elementarnych komórek pamięci. Dlatego też analiza przebiegu tak wielu sygnałów jest czasochłonna i nierozerwalnie wiąże się z możliwością popełnienia błędu.

3. METODA GRAFCET I SFC

Metoda i język Grafcet została opracowana w połowie lat 70-tych XX wieku przez grupę naukowców francuskich. Znaczenie nazwy „GRAFCET" nie jest jednoznaczne. Przyjmuje się, że nazwa związana jest z komisją która opracowała metodę i formalnie zaprezentowała w pierwszej publikacji na ten temat [28] - „Schemat francuskiego stowarzyszenia cybernetyki, ekonomii i techniki" (fr. ***GR****aphe de l'****A****ssociation* ***F****rançaise pour la* ***C****ybernétique,* ***É****conomique et* ***T****echnique*). Można także spotkać się z innym rozwinięciem [19] - „Schemat funkcjonalny sterowania: Krok – Warunek" (fr. ***GRA****phe* ***F****onctionnel de* ***C****ommande:* ***E****tape –* ***T****ransition*). Jako język Grafcet występuje w sterownikach firmy Telemecanique, dostępnych obecnie pod marką Schneider Electric.

SFC (ang. *Sequential Function Chart*) jest zarówno siecią służącą do modelowania procedur jak i językiem programowania sterowników PLC zdefiniowanym w normie PN-EN 61131-3.

Obydwie metody bazują na sieci Petriego [3], przez co są do siebie bardzo podobne, a obydwa języki programowania należą do grupy języków graficznych, które opisują porządek wykonywania poszczególnych kroków w ramach programu. Identyczna jest również struktura obydwu metod, w których z poszczególnymi krokami programu skojarzone są wykonywane akcje. Przejście pomiędzy krokami jest kontrolowane przez tranzycje.

Ze względu na wspólne pochodzenie w trakcie modelowania stosowane są te same reguły [33, 36, 37]. Występujące różnice pomiędzy Grafcet i SFC dotyczą jedynie składni.

Algorytm procesu w sieci Grafcet i SFC analitycznie opisuje zależność [33]:

$$G=<E, T, L> \tag{3.1}$$

gdzie:
$E=\{e_1, e_2, \ldots , e_e\}$ - skończony, niepusty zbiór miejsc reprezentujących etapy elementarne,
$T=\{t_1, t_2, \ldots , t_t\}$ - skończony, niepusty zbiór tzw. tranzycji,
L - zbiór odcinków zorientowanych prowadzących od tranzycji do miejsc lub od miejsc do tranzycji.

W trakcie budowy algorytmu procesu za pomocą sieci Grafcet lub SFC należy przestrzegać poniższych reguł [3].

Reguła 1
Każdemu etapowi elementarnemu procesu musi odpowiadać tylko jedno miejsce w sieci Grafcet oznaczone innym, nie powtarzającym się, numerem.

Reguła 2
Miejsca w sieci Grafcet zawsze muszą być rozdzielone tranzycjami.

Reguła 3
Tranzycje zawsze muszą być rozdzielone miejscami.

Algorytm sterowania sieci Grafcet i SFC opisuje zależność [33]:

$$G=<S, T, L> \tag{3.2}$$

gdzie:
$S=\{s_1, s_2, \ldots , s_e\}$ - skończony, niepusty zbiór miejsc zwanych krokami, reprezentującymi sygnały wyjściowe sterownika PLC,
$T=\{t_1, t_2, \ldots , t_t\}$ - skończony, niepusty zbiór tranzycji,
L - zbiór odcinków zorientowanych prowadzących od tranzycji do miejsc lub od miejsc do tranzycji

Sposób funkcjonowania algorytmu sterowania Grafcet i SFC zdefiniowany jest następującymi regułami:

Reguła 1
Zmiana stanu układu sterowania jest reprezentowana przez aktywność tranzycji. Tranzycja jest aktywna gdy:

- krok ją poprzedzającą jest aktywny i zostało zakończone jego wykonanie,
- warunek logiczny określający tranzycję ma wartość logiczną równą 1.

Reguła 2
Aktywność tranzycji powoduje:

- przejście w stan aktywny kroku następującego po tej tranzycji,
- przejście w stan nieaktywny kroku poprzedzającego tę tranzycję.

3.1. Składnia języka Grafcet

Algorytm procesu zapisany siecią Grafcet [3] jest grafem zorientowanym składającym się z miejsc i tranzycji. Miejsce jest graficzną reprezentacją etapu elementarnego, czyli operacji nie podlegającej dalszej dekompozycji. Każde miejsce oznaczone jest niepowtarzającym się numerem. Symbol miejsca przedstawiono na poz. 1 rys. 3.1. Miejsce może być skojarzone z akcją określającą której zmiennej wyjściowej ma zostać przypisana wartość logiczna '1'. Dla przejrzystości i zrozumienia modelowanego procesu algorytm można uzupełnić o komentarze. Umieszcza się je w ramce akcji, po prawej stronie symbolu miejsca, i łączy z nim za pomocą linii ciągłej (poz. 4 rys. 3.1).

Na początku algorytmu procesu znajduje się miejsce początkowe (poz. 2 rys. 3.1). Miejsce początkowe (etap początkowy) określa stan procesu w chwili rozpoczęcia jego realizacji. Na poziomie sterowania etap ten oznacza stan po zainicjowaniu pracy sterownika.

Tranzycja reprezentuje warunek przejścia pomiędzy kolejnymi miejscami. Z każdą tranzycją skojarzone są dwa odcinki zorientowane. Pierwszy łączący etap poprzedzający z tranzycją, drugi łączący tranzycję z etapem następującym po niej. Tranzycja reprezentowana jest w sposób graficzny tak jak przedstawiono to na poz. 3 rys. 3.1.

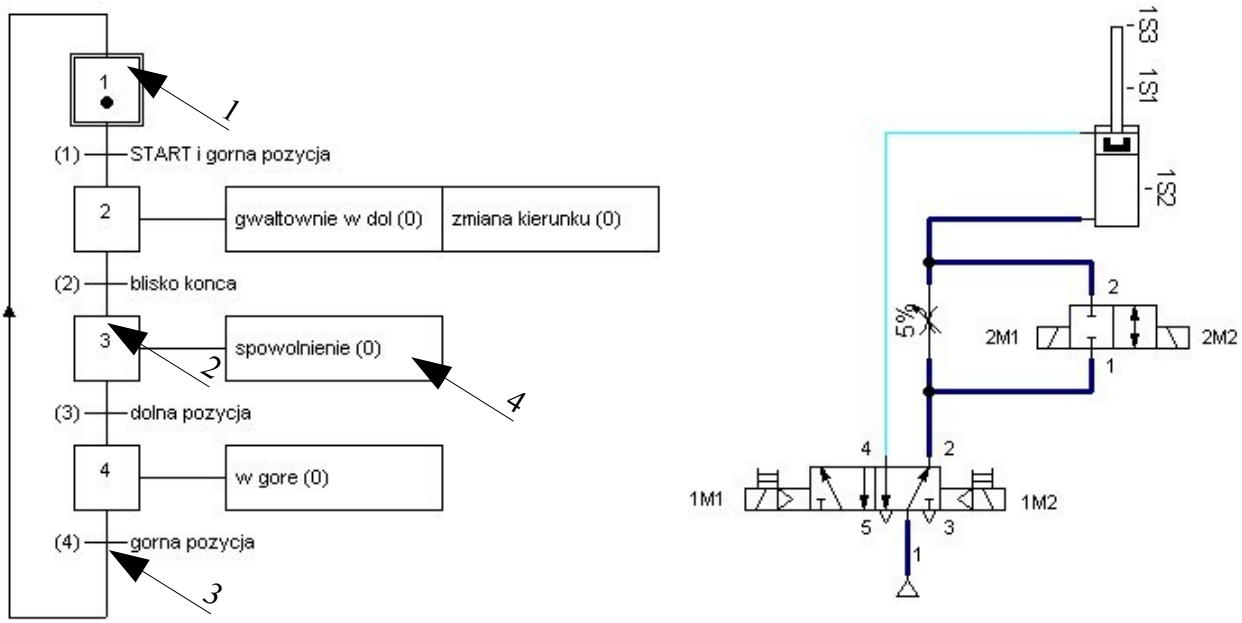
W przypadku procedur współbieżnych stosowana jest tranzycja współbieżna, która reprezentuje warunek przejścia pomiędzy:

- pojedynczym etapem a etapami rozpoczynającymi się jednocześnie (poz. 1 rys. 3.2),
- etapami realizowanymi współbieżnie, a pojedynczym etapem występującym po zakończeniu ich wykonywania (poz. 2 rys. 3.2).

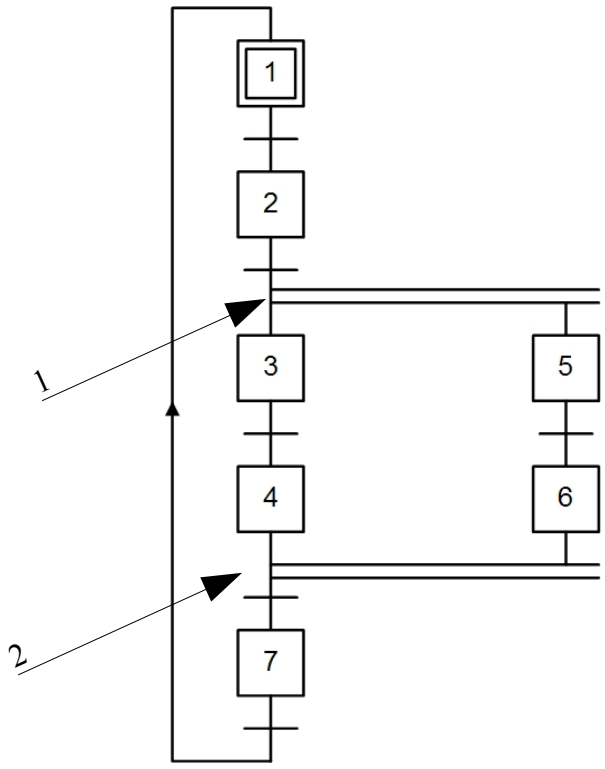
Ze względu na powszechnie przyjętą koncepcję zapisu modelowanego procesu od góry do dołu odcinki łączące miejsca i tranzycje rysuje się bez strzałek. Strzałki stosuje się jedynie w przypadku łączenia miejsc i tranzycji niezgodnie z powyższą koncepcją (rys. 3.1).

Istotnym elementem składni Grafcet są makroetapy umożliwiające dekompozycję algorytmu na mniejsze części. Dzięki temu algorytm staje się bardziej czytelny. Makroprocedura jest reprezentowana w algorytmie poprzez symbol przedstawiony na poz. 1 rys. 3.3. W algorytmie definiującym makroprocedurę stosowane są te same zasady i symbole występujące w głównym algorytmie. Wyjątkiem od tego są etapy:

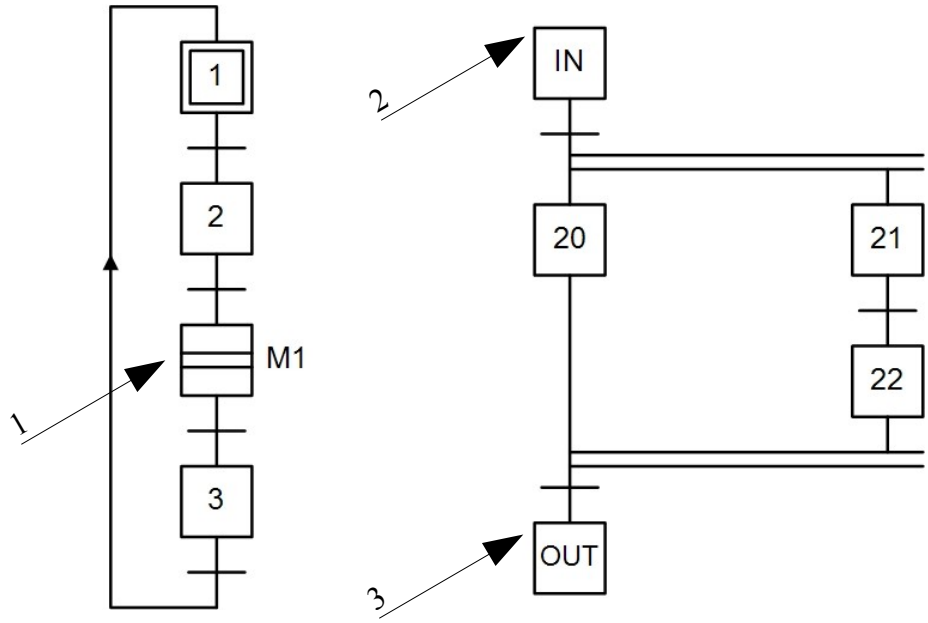
- IN (poz. 2 rys. 3.3) – reprezentujący stan procesu w momencie rozpoczęcia makroetapu,
- OUT (poz. 3 rys. 3.3) – oznaczający zakończenie makroprocedury i powrót do wykonywania algorytmu głównego.



Rys. 3.1. Przykład modelowania procedury sekwencyjnej siecą Grafcet; 1-miejsce; 2-miejsce początkowe; 3-tranzycja; 4-komentarz



Rys. 3.2. Przykład modelowania procedury współbieżnej siecią Grafcet; 1-rozpoczęcie procedury współbieżnej; 2-zakończenie procedury współbieżnej



Rys. 3.3. Przykład makroprocedury współbieżnej występującej w sieci Grafcet; 1-symbol makroetapu; 2-etap reprezentujący stan procesu w momencie rozpoczęcia makroetpu; 3-etap reprezentujący zakończenia makroetapu

3.2. Składnia języka SFC

SFC jest językiem programowania opisującym w sposób graficzny algorytm sterowania. Z tego względu w tym przypadku podstawowymi elementami są kroki. W Grafcet mieliśmy do czynienia z miejscami (uniwersalna nazwa), etapami odnoszącymi się do algorytmu procesu lub krokami w przypadku właśnie algorytmu sterowania.

W SFC wyszczególnić można dwa rodzaje kroków – inicjalizujący (poz. 1 rys. 3.4) i właściwy (poz. 2 rys. 3.4). Po uruchomieniu programu napisanego w SFC aktywny jest tylko krok inicjalizujacy. Krok właściwy staje się aktywny tylko gdy aktywny jest krok poprzedzający go
i spełniona jest tranzycja poprzedzająca (poz. 3 i 4 rys. 3.4). Do każdego z rodzaju kroków można przyporządkować akcje.

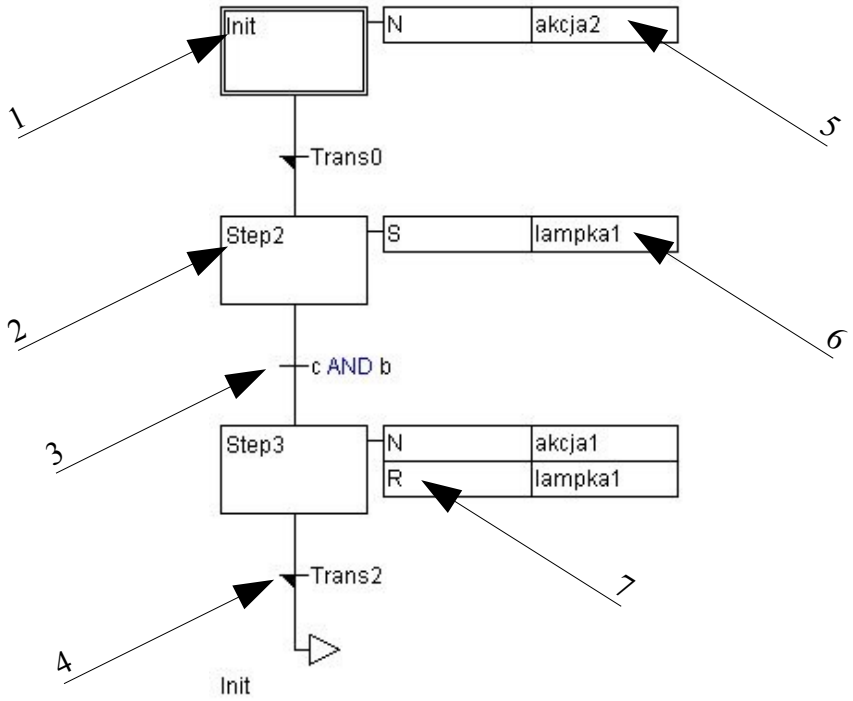
Akcje skojarzone są z krokami i to one odpowiadają za stan sygnałów wyjściowych układu sterowania. W akcjach można odwoływać się bezpośrednio do zmiennych (poz. 6 rys. 3.4) lub do programów zapisanych za pomocą języka LD, FBD, ST lub IL (poz. 5 rys. 3.4).

Każdej akcji w SFC przypisany jest jeden specyficzny kwalifikator (poz. 7 rys. 3.4). Szczegółowy opis kwalifikatorów zawarto w tabeli 3.1.

Kwalifikatory L, D, SD, DS i SL wymagają podania wartości czasu w postaci “T#5s”, umieszczanego za kwalifikatorem, lub za pomocą zmiennej typu TIME (np. „t_var”).

Poszczególne kroki rozdzielone są tranzycjami odpowiadającymi za przechodzenie algorytmu sterowania między krokami zgodnie z założonym porządkiem. Mogą być zapisywane bezpośrednio poprzez odwołanie do konkretnej zmiennej, iloczynu i/lub sum zmiennych (poz. 3 rys. 3.4) bądź poprzez warunek zapisany za pomocą jednego z języków LD, FBD, ST lub IL (poz. 4 rys. 3.4).

Na rysunku 3.5 przedstawiono przykładowy algorytm zapisany za pomocą języka SFC, w którym zaprezentowano akcje i tranzycje zapisane za pomocą języków LD, FBD, IL i ST.

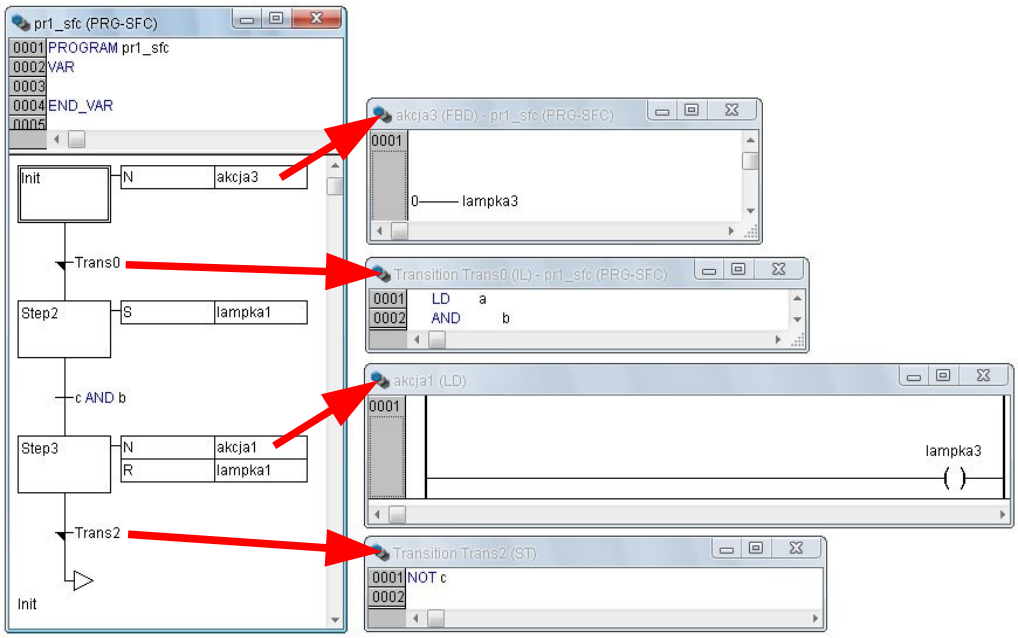


Rys. 3.4. Przykładowy algorytm zapisany za pomocą sieci SFC; 1-krok inicjalizujący; 2-krok właściwy; 3-warunek tranzycji zapisany bezpośrednio; 4-warunek tranzycji zapisany w programie Trans2; 5-akcja zdefiniowana w programie o nazwie akcja2; 6-akcja zdefiniowana poprzez bezpośrednie odwołanie do zmiennej lampka1; 7-kwalifikatory akcji

Tabela 3.1. Kwalifikatory akcji

Kwalifikator	**Znaczenie**	**Opis**
N	chwilowy (ang. *Non-stored*)	akcja jest aktywna tak długo jak aktywny jest krok
R	kasowanie/reset (ang. *overriding Reset*)	akcja jest deaktywowana (reset) gdy krok jest aktywny
S	zapis/set (ang. *set-Stored*)	akcja staje się aktywna gdy krok jest aktywny i trwa do momentu deaktywacji (reset)
L	ograniczona czasowo (ang. *time Limited*)	akcja jest aktywna przez określony czas od momentu aktywacji kroku (maksymalnie przez czas aktywności tego kroku)
D	opóźniona czasowo (ang. *time Delayed*)	akcja staje się aktywna po określonym czasie od momentu aktywacji kroku jeśli krok jest nadal aktywny i pozostaje aktywna tak długo jak krok jest aktywny
P	impuls (ang. *Pulse*)	akcja wykonywana jest tylko jeden raz gdy krok staje się aktywny
SD	zapisana i opóźniona czasowo (ang. *Stored and time Delayed*)	akcja staje się aktywna po upływie określonego czasu i trwa do momentu deaktywacji (reset)

DS	opóźniona i przechowywana (ang. *Delayed and Stored*)	akcja staje się aktywna po upływie określonego czasu pod warunkiem aktywności kroku i trwa do momentu deaktywacji (reset)
SL	zapisana i ograniczona czasowo (ang. *Stored and time Limited*)	akcja jest aktywna przez określony czas od momentu aktywacji kroku



Rys. 3.5. Algorytm zapisany za pomocą sieci SFC, z akcjami i tranzycjami zapisanymi za pomocą języków LD, FBD, IL, ST

3.3. Zalety i wady języka Grafcet i SFC

Ze względu na zbliżone pochodzenia obydwu metod i języków zarówno ich zalety jak i wady są podobne. Do zalet zaliczyć można prostotę i intuicyjność pisania programu. Po stronie wad liczba cech jest zdecydowanie większa.

Przede wszystkim maksymalna liczba kroków i tranzycji zależy od implementacji (sterownika PLC) i jest zdecydowanie mniejsza niż w przypadku LD, FBD, IL czy ST.

Po drugie, implementacja opracowanego programu możliwa jest jedynie w sterownikach PLC dysponujących tym językiem, który nadal nie jest powszechnie dostępny. Norma definiująca Grafcet (PN-EN 60848:2003), jak i norma opisująca SFC (PN-EN 61131-3) nie określają bowiem sposobu przetwarzania opracowanego modelu na równanie schematowe umożliwiające zapis za pomocą innych zdefiniowanych w normie języków. Istnieje co prawda metoda umożliwiająca przetworzenie modelu zapisanego za pomocą sieci Grafcet na języka LD [3], opierająca się na przyporządkowaniu każdemu krokowi algorytmu sterowania elementarnej komórki pamięci i odpowiednim zapisywaniu oraz kasowaniu tych komórek. Oznacza to jednak, że liczba komórek pamięci użytych w algorytmie sterowania jest nie mniejsza od liczby etapów elementarnych algorytmu procesu.

Po trzecie żadna z norm nie definiuje pojęcia etapu elementarnego oraz nie wymusza stworzenia precyzyjnego opisu słownego.

Niemniej jednak, pomimo tylu wad, metoda zdobywa coraz to większe zainteresowanie [4, 5], a w związku z rozwojem sterowników PLC ograniczenia sprzętowe stają się być coraz mniejsze.

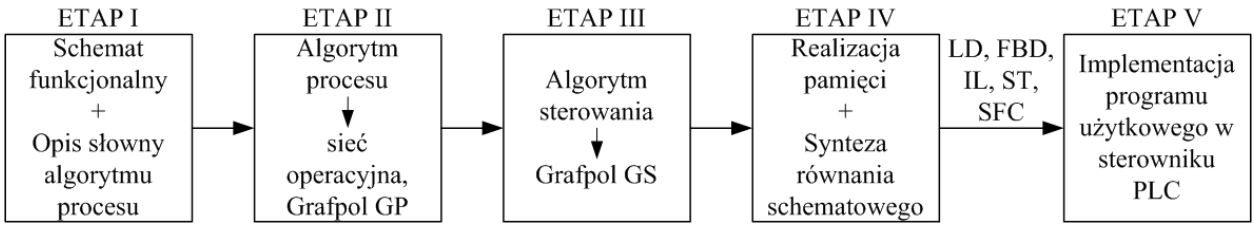
4. METODA GRAFPOL

Metoda Grafpol została opracowana w latach 90-tych XX wieku w Laboratorium Podstaw Automatyzacji Instytutu Technologii Maszyn i Automatyzacji Politechniki Wrocławskiej przez Tadeusza Mikulczyńskiego i Zdzisława Samsonowicza [32, 33].

Modelowanie i programowanie metodą Grafpol dyskretnych procesów produkcyjnych, do realizacji których są stosowane napędy pneumatyczne, hydrauliczne i elektryczne, obejmuje następujące etapy:

- Etap I - opracowanie schematu funkcjonalnego procesu, podział procesu na etapy elementarne oraz sformułowanie opisu słownego przebiegu jego realizacji (algorytm procesu). Schemat funkcjonalny powinien przedstawiać proces w stanie początkowym. Musi on zawierać wszystkie elementy lub zespoły wykonawcze poszczególnych etapów elementarnych oraz opis słowny tych etapów.
- Etap II - budowa graficzno-analitycznego modelu matematycznego algorytmu procesu. Do reprezentacji metodą Grafpol modeli matematycznych algorytmów dyskretnych procesów produkcyjnych jest stosowana sieć operacyjna i wyznaczana na jej podstawie sieć Grafpol GP.
- Etap III - synteza algorytmu sterowania, reprezentowana przez sieć Grafpol GS. Algorytm sterowania otrzymuje się w wyniku transformacji algorytmu procesu. Transformacja polega na odwzorowaniu zbioru etapów elementarnych procesu zbiorem sygnałów wyjściowych sterownika PLC, które sterują realizacją poszczególnych etapów elementarnych.
- Etap IV - realizacja pamięci algorytmu sterowania i wyznaczenie równania schematowego. Pamięć wyznaczana jest na podstawie algorytmu sterowania. Po zrealizowaniu pamięci możliwe jest wyznaczenie równania schematowego, które stanowi sumę funkcji wszystkich zmiennych wyjściowych oraz elementarnych komórek pamięci.
- Etap V - zapis programu użytkowego sterownika PLC. Podstawę do zapisu programu użytkowego sterownika PLC stanowi równanie schematowe. Program można zapisać stosując jeden z języków programowania PLC przyjętych w normie PN-EN 61131-3. Są to następujące języki: LD (ang. *Ladder Diagram*), FBD (ang. *Function Block Diagram*), IL (ang. *Instruction List*) oraz ST (ang. *Structured Text*).

Powyżej przedstawione etapy zamieszczono na rys. 4.1.



Rys. 4.1. Etapy modelowania wg metody Grafpol

4.1. Etap I - schemat funkcjonalny i opis słowny

Schemat funkcjonalny oraz opis słowny tworzone są analogicznie jak w metodzie MTS.

4.2. Etap II - algorytm procesu

W metodzie Grafpol algorytm procesu tworzony jest analogicznie jak w metodzie MTS na podstawie schematu funkcjonalnego i opisu słownego. W przypadku rozbudowanych algorytmów ich reprezentacja za pomocą sieci operacyjnej staje się nieczytelna. Dlatego też w metodzie Grafpol algorytm dyskretnego procesu produkcyjnego można przedstawić za pomocą innego, równoważnego sieci operacyjnej, modelu sieciowego, którym jest sieć Grafpol

GP. Stanowi go trójka:

$$GP=<E, T, K> \tag{4.1}$$

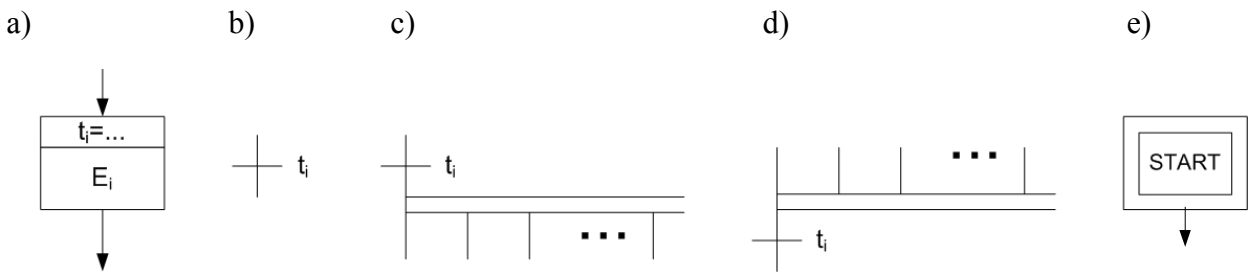
gdzie:

$E=\{e_1, e_2, ..., e_e\}$ – skończony, niepusty zbiór miejsc, które reprezentują etapy elementarne procesu,

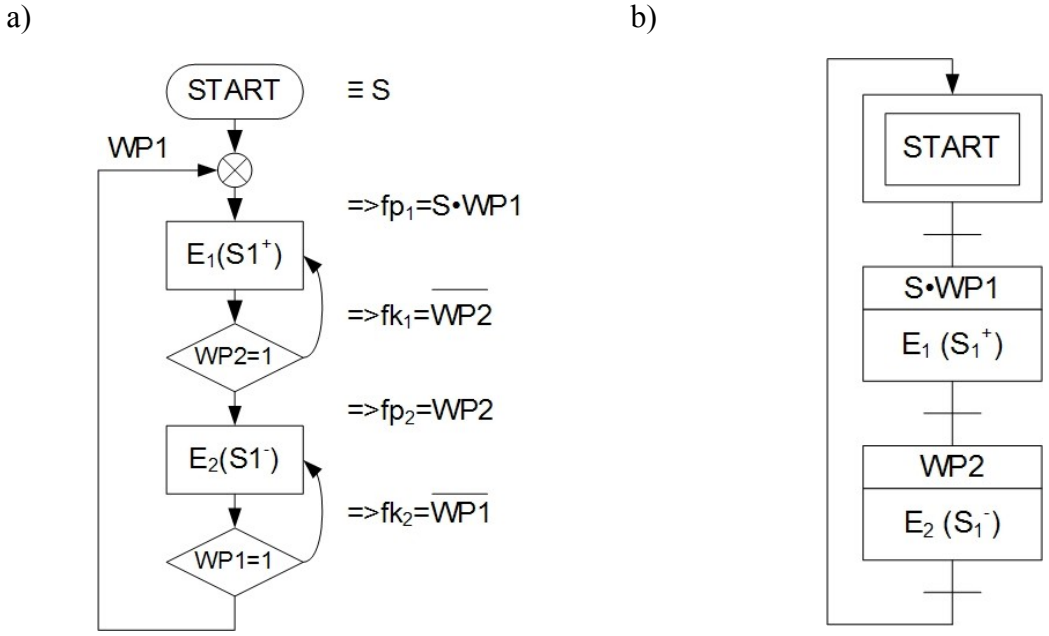
$T=\{t_1, t_2, ..., t_t\}$ – skończony, niepusty zbiór tranzycji (przejść), które reprezentują warunki logiczne realizacji etapów elementarnych procesu,

K – zbiór odcinków zorientowanych, które określają kierunki przepływu sygnałów w sieci Grafpol.

Do budowy sieci Grafpol GP stosuje się symbole graficzne pokazane na rys. 4.2. Na rysunku 4.3 przedstawiono model matematyczny przykładowego algorytmu pracy jednego napędu pneumatycznego opisanego słownie w pkt. 2.1.



Rys. 4.2. Symbole graficzne elementów sieci Grafpol: a) miejsce, b) tranzycja, c) tranzycja określająca rozpoczęcie realizacji procedury współbieżnej, d) tranzycja reprezentująca zakończenie realizacji procedury współbieżnej, e) etap START



Rys. 4.3. Algorytm przykładowego procesu dozowania materiału sypkiego opisanego w pkt. 2.1. a) sieć operacyjna, b) sieć Grafpol GP

4.3. Etap III - algorytm sterowania

W wyniku transformacji algorytmu procesu (sieci Grafpol GP) otrzymuje się algorytm sterowania, który reprezentuje sieć Grafpol GS. Jest nią trójka:

$$GS=<S, T, K> \tag{4.2}$$

gdzie:

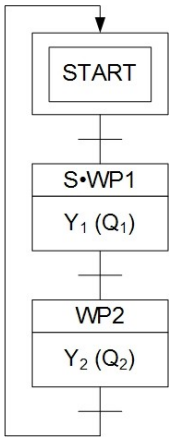
$S=\{s_1, s_2, ..., s_s\}$ – skończony, niepusty zbiór miejsc zwanych krokami. Reprezentują one sygnały wyjściowe sterownika PLC, które sterują realizacją poszczególnych etapów elementarnych procesu.

$T=\{t_1, t_2, ..., t_t\}$ – skończony, niepusty zbiór tranzycji (przejść), które reprezentują warunki logiczne realizacji etapów elementarnych procesu,

K – zbiór odcinków zorientowanych, które określają kierunki przepływu sygnałów w sieci Grafpol.

Algorytm sterowania stanowi podstawę do realizacji pamięci, i w efekcie wyznaczenia równania schematowego w oparciu o które jest zapisywany program użytkowy PLC. Program użytkowy steruje procesem zgodnie z założonym algorytmem pracy.

Na rysunku 4.4 pokazano algorytm sterowania pracą jednego napędu pneumatycznego, którego algorytm pracy przedstawiono na rys. 4.3b.



Rys. 4.4. Utworzony na podstawie algorytmu pracy (rys. 4.3) algorytm sterowania przedstawiony za pomocą sieci Grafpol GS

4.4. Etap IV - realizacja pamięci

Synteza równania schematowego jest wykonywana na bazie algorytmu sterowania, który reprezentuje sieć Grafpol GS. Podczas syntezy obowiązują reguły zdefiniowane w metodzie MTS, odnoszące się do realizacji pamięci i tworzenia równania schematowego.

4.5. Etap V - zapis programu

Zapis opracowanego równania schematowego w postaci programu zrozumiałego przez sterownik PLC. Program może być zapisany przy wykorzystaniu języków programowania ujętych w normie PN-EN 61131-3:

- LD (Ladder Diagram),
- FBD (Function Block Diagram),
- IL (Instruction List),
- ST (Structured Text).

Równanie schematowe może być również zapisane przez języki nie ujęte w normie, stworzone przez producentów sterowników i mające zastosowanie jedynie w ich wyrobach.

5. STEROWNIKI PLC

PLC (ang. *Programmable Logic Controller*) to w dosłownym tłumaczeniu Programowalny Sterownik Logiczny. W dużym uproszczeniu jest to układ elektroniczny mający możliwość wielokrotnego programowania równań boolowskich opisujących zależności między stanem wejść i wyjść, mogący uwzględniać także stan wyjść w chwilach poprzednich. Od momentu rynkowego debiutu, na przełomie lat 70-tych i 80-tych XX wieku, sterowniki PLC w zdecydowanej mierze wyparły układy sterowania realizowane za pomocą elementów pneumatycznych i stykowo-przekaźnikowych. Sterowniki PLC należą do grupy układów synchronicznych, gdzie sygnał synchronizujący związany jest z cyklem pracy sterownika. Ze względu na sposób fizycznej realizacji sterownika (układy elektroniczne oparte na tranzystorach) w sterownikach nie występuje zjawisko wyścigu i hazardu.

Sterowniki PLC pod względem budowy podzielić można na kompaktowe i modułowe.

Sterowniki kompaktowe to sterowniki w których jednostka centralna znajduje się w jednej obudowie z określoną liczbą wejść/wyjść i nie istnieje możliwość rozszerzenia ich liczby poprzez bezpośrednie dołączenie dodatkowych modułów. W niektórych przypadkach możliwe jest zwiększenie liczby wejść/wyjść poprzez dołączenie zewnętrznych (zdalnych) modułów wejść/wyjść za pomocą dostępnego w sterowniku interfejsu komunikacyjnego. Przykład takiego sterownika przedstawiono na rys.5.1a.

Sterowniki modułowe w przeciwieństwie do kompaktowych, mają możliwość znacznego wpływu przez użytkownika na konfigurację całego sterownika. Z reguły można dowolnie łączyć jednostki centralne o różnych parametrach z modułami wejść/wyjść w zależności od potrzeb danej aplikacji. Przykład takiego sterownika przedstawiono na rys. 5.1b.

Patrząc całościowo na sterowniki PLC - nie są one urządzeniami jednolitymi [26]. Z jednej strony można spojrzeć na nie od strony budowy, a więc konstrukcji mechanicznej i obwodów elektrycznych oraz elektronicznych (ang. *hardware*), z drugiej strony istotne są również programy, funkcje i procedury definiujące zakres możliwości obliczeniowych oraz sposób ich wykonywania (ang. *firmware*) oraz programy pozwalające na zaprogramowanie sterownika przez użytkownika (ang. *software*).

a) b)

Rys. 5.1. Sterowniki PLC firmy Festo: a) kompaktowy FEC 660; b) modułowy CECX

5.1. Hardware

Pod pojęciem hardware'u kryją się elementy sprzętu umożliwiające realizację przez sterownik określonych funkcji. Podstawowa funkcja, jaką jest wykonywanie obliczeń, realizowana jest przez jednostkę centralną. Do obliczeń tych niezbędne są sygnały wejściowe i wyjściowe sterowanego procesu, dlatego też do hardware'u zaliczamy moduły wejść i wyjść, zarówno analogowe jak i cyfrowe. W sposób sprzętowy realizowane są również funkcje komunikacji w sieciach rozproszonych przy wykorzystaniu różnych protokołów komunikacji, m.in. Profibus, CanOpen, Ethernet IP.

W ostatnich latach dynamiczny rozwój sterowników w kierunku zwiększenia obszaru ich zastosowań zaowocował zwiększeniem się różnego rodzaju modułów specjalizowanych, jak np. obsługujące bezpośrednio termoelementy, czy sterujące silnikami elektrycznymi i serwonapędami.

5.1.1. Jednostka centralna

Jednostka centralna (ang. *Central Processing Unit*) jest centrum obliczeniowym całego sterownika PLC. Jej zadanie polega na przetwarzaniu informacji pochodzących z modułów wejściowych, zgodnie z zaprogramowanym przez użytkownika algorytmem, na sygnały modułów wyjściowych. Ponadto, jednostka centralna odpowiada za:

- komunikację za pomocą interfejsów z innymi sterownikami, rozproszonymi modułami wejść/wyjść lub sieciami nadrzędnymi,
- kontrolę czasu cyklu,
- diagnostykę,
- zarządzanie pamięcią.

Informacje w jednostce centralnej są przechowywane i przetwarzane w różnych rodzajach pamięci. Wyróżnić można następujące rodzaje pamięci:

- RAM (ang. *Random Access Memory*) – używana w trakcie wykonywania programu, tzw. pamięć robocza. Jest to pamięć ulotna, czyli zanikająca w przypadku zaniku zasilania.
- ROM (ang. *Read Only Memory*) – zawiera dane konfiguracyjne sterownika oraz program użytkowy. Pomimo nazwy sugerującej możliwość tylko odczytu danych z tego rodzaju pamięci może być ona zapisywana (np. w trakcie wgrywania programu) – wtedy nazywana jest pamięcią PROM (ang. *Programmable Read-Only Memory*). Rozwinięciem tego rodzaju pamięci jest EPROM (ang. *Erasable Programmable Read-Only Memory*) – charakteryzująca się możliwością kasowania i odczytu. Do jej zaprogramowania wymagany jest programator. Najbardziej rozwiniętym rodzajem pamięci jest EEPROM (ang. *Electrically Erasable Programmable Memory*) którą można kasować przy użyciu prądu elektrycznego. Pamięci tego typu charakteryzują się żywotnością do 100 000 cykli zapisu oraz nieograniczoną liczbą odczytów.

Z pamięcią nierozerwalnie związany jest sposób jej adresowania. Najmniejszą logiczną jednostką pamięci jest bit. Osiem bitów tworzy bajt. Bity mogą być zapisywane (wprowadzenie wartości '1'), kasowane (wprowadzenie wartości '0') lub odczytywane. Pamięć sterownika może być odczytywana bitowo, przez podanie adresu konkretnego bitu, np. M1.0 – komórka pamięci typu memory, bajt 1, bit 0 bajtu 1 lub przez odczyt całych bajtów, np. MW1.

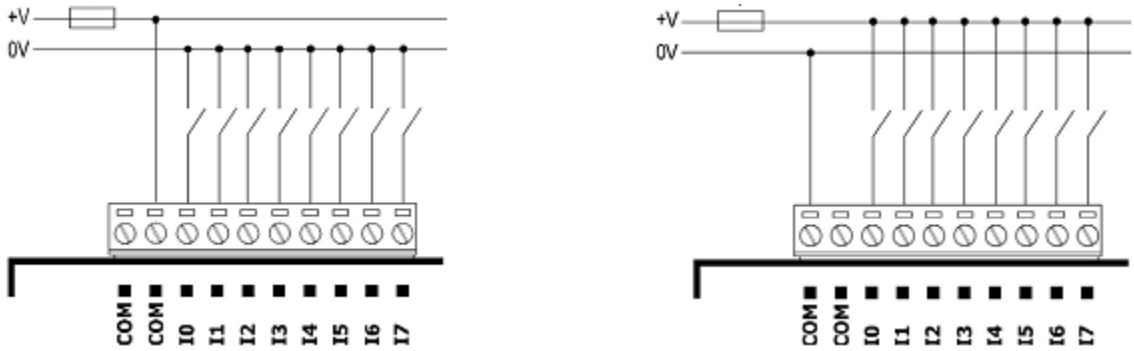
5.1.2. Typy wejść i wyjść modułów I/O

W modułach wejściowych jak i wyjściowych sterowników PLC możemy wyszczególnić dwa rodzaje przyłączy – pnp i npn. Nazwa obydwu typów przyłączy wywodzi się z kolejności ułożenia półprzewodników w tranzystorze.

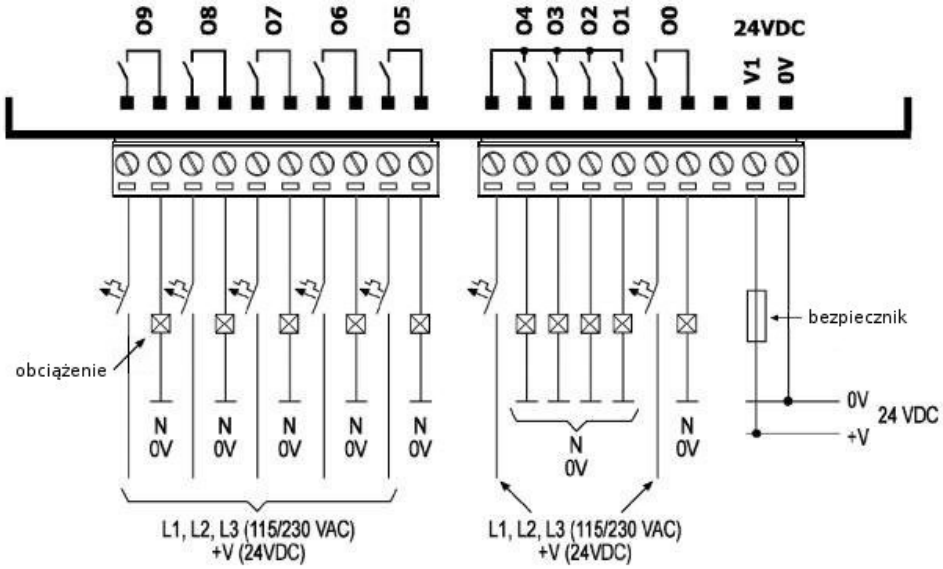
W przypadku modułów wejściowych, sygnały z czujników możemy podłączyć zgodnie z przyłączem pnp (rys. 5.2b), wpuszczając nijako potencjał dodatni do modułu, lub wg przyłącza npn (rys. 5.2a), wyciągając sygnał dodatni z modułu.

W odniesieniu do modułów wyjściowych występują dwa ich zasadnicze rodzaje. Moduły przekaźnikowe (rys. 5.3), gdzie w wyniku wysterowania wyjścia występuje jedynie przesterowanie styku lub moduły tranzystorowe (rys. 5.4). Moduły przekaźnikowe charakteryzują się większą obciążalnością, zaś moduły tranzystorowe upraszczają połączenie modułu wyjściowego z elementami wykonawczymi (zmniejszają ilość przewodów okablowania). Moduły przekaźnikowe mogą występować w dwóch wersjach npn i pnp.

a) b)

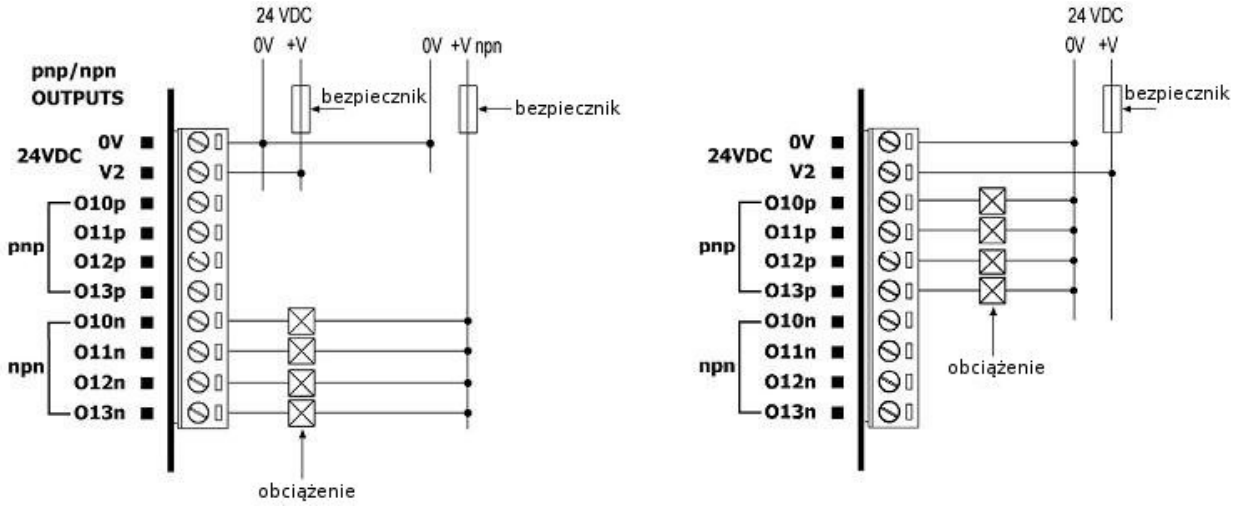


Rys. 5.2. Sposób podłączenia czujników do modułów wejściowych: a) npn (tzw. zlew), b) pnp (tzw. źródło) [40]



Rys. 5.3. Sposób podłączenia obciążenia do modułu wyjściowego przekaźnikowego [40]

a) b)



Rys. 5.4. Sposób podłączenia obciążenia do modułu wyjściowego tranzystorowego: a) npn, b) pnp [40]

5.1.3. Moduł wejść binarnych

Moduły wejść binarnych, zwane również modułami wejść cyfrowych (ang. *Digital Input*), stanowią część sterownika pozwalającą na podłączenie sygnałów zewnętrznych do sterownika PLC. Zadaniem tych modułów jest przekształcenie otrzymywanej informacji z czujnika na bit. Bit (ang. *binary digit*) to najmniejsza jednostka informacji w technice cyfrowej, która może przyjmować tylko dwie wartości – logiczne '0' lub '1'. Innymi słowy, moduł wejść cyfrowych przyporządkowuje logiczne '1' lub '0' do danego wejścia w zależności od napięcia z przyłączonego czujnika. Przykładowo moduł wejściowy V200-18-E2B sterownika Unitronics Vision V260 przy połączeniu pnp przyporządkowuje logiczne '0' dla sygnału o napięciu 0-5VDC i '1' dla sygnału o napięciu 17-28,8VDC. W stanach z zakresu 5-17VDC nie następuje zmiana przydzielonej wcześniej wartości logicznej.

Dla wygody użytkownika, biorąc pod uwagę iż czujniki bardzo często potrzebują zasilania, moduły te wyposażone są w wyprowadzone zasilanie 24VDC i 0VDC oraz wejście cyfrowe w ramach pojedynczego przyłącza czujnika. Dzięki temu proces przyłączenia czujników jest uproszczony, gdyż w jednym miejscu znajdują się zaciski zasilające i sygnałowe.

Z punktu widzenia sterownika PLC nie ma znaczenia jakie wielkości fizyczne są mierzone przez czujniki podłączone do jego wejść cyfrowych. Jednym wymaganiem jest dwustanowy charakter przekazywanych sygnałów. Dlatego też obecnie stosowane czujniki coraz częściej stanowią element integrujący w sobie czujnik oraz programowalny przetwornik. Dzięki takiemu połączeniu możliwe staje się ustawienie wartości wielkości mierzonej przy której na wyjściu czujnika ma pojawić się sygnał napięciowy 24VDC. W związku z tym do cyfrowych modułów wejściowych można podłączyć m.in. czujniki:

- indukcyjne,
- optyczne,
- stykowe (np. krańcówki, przyciski),
- ciśnieniowe,
- pojemnościowe.

5.1.4. Moduł wyjść binarnych

Moduły wyjść binarnych, zwane również modułami wyjść cyfrowych (ang. *Digital Output*), pozwalają na oddziaływanie sterownika PLC na elementy wykonawcze. Ich zadanie polega na przekształceniu dwustanowych wartości logicznych na dwustanowe sygnały prądowe (najczęściej 0VDC lub 24VDC). Moduły wyjściowe są również wyposażone w pogrupowane dla danego wyjścia przyłącze sygnałowe oraz w zależności od typu przyłącze wspólnej masy lub plusa.

Współczesne sterowniki PLC dysponują wyjściami tranzystorowymi o obciążalności 0,5A na kanał. Upraszcza to prace instalatorskie, gdyż w wielu przypadkach umożliwia bezpośrednie podłączenie, bez konieczności stosowania przekaźników, elementów takich jak m.in.:

- sygnalizatory (lampki, buzery)
- cewki elektrozaworów,
- styczniki silników.

5.1.5. Moduł wejść analogowych

Moduł wejść analogowych to specjalistyczny przetwornik analogowo-cyfrowy połączony z jednostką centralną sterownika PLC. W przeciwieństwie do modułu wejść cyfrowych, przetwarzającego sygnał wejściowy na wartość logicznego '0' lub '1', moduł wejść analogowych przetwarza sygnał na wartości z zakresu zależnego od jego rozdzielczości. Zakres ten wynosi od $0\text{-}2^x$, gdzie x-określa rozdzielczość modułu wejść analogowych. Przykładowo, dla modułu o rozdzielczości 12 bitowej zakres ten wynosi 0-4095. Otrzymana wartość cyfrowa jest

proporcjonalna do analogowego sygnału wejściowego.

Proces przetwarzania sygnału analogowego na cyfrowy składa się z dwóch głównych etapów [21, 22]. Pierwszy to próbkowanie, czyli określenie co jaki odstęp czasu sygnał analogowy będzie „odczytywany". Drugim etapem jest kwantyzacja, czyli przypisanie odczytanej wartości analogowej do najbliższego poziomu reprezentacji. Niestety, przetwarzanie sygnałów analogowych na cyfrowe wiąże się z nieuniknioną i nieodwracalną utratą informacji.

Moduły wejść analogowych, w zależności od typu, umożliwiają pomiar prądu lub napięcia którego źródłem są różnego rodzaju czujniki analogowe.

5.1.6. Moduł wyjść analogowych

Moduł wyjść analogowych jest przetwornikiem cyfrowo-analogowym, przekształcającym obliczone zgodnie z programem użytkownika sygnały sterujące z postaci cyfrowej na proporcjonalne wartości napięcia lub prądu. Podobnie jak moduł wejść analogowych moduł wyjść analogowych charakteryzuje się rozdzielczością przetwornika określającą współczynnik proporcji między wartością cyfrową a wartością analogową sygnału.

5.2. Firmware

Firmware jest oprogramowaniem „wbudowanym" urządzenia na które użytkownik sterownika nie ma wpływu. Definiuje ono podstawowe procedury obsługi danego urządzenia, sposób jego funkcjonowania, a także komunikacji z innymi urządzeniami. W odniesieniu do sterowników PLC firmware określa m.in. zakres zmiennych i spis instrukcji obsługiwanych przez daną jednostkę centralną bądź moduł wejść/wyjść. Tego typu oprogramowanie wgrywane jest przez producenta, a użytkownik, w większości przypadków, nie ma możliwości jego modyfikacji. Działanie użytkownika związane z firmwarem może dotyczyć jedynie jego aktualizacji (ang. *update*) na nowszą wersję dostarczaną przez producenta.

5.3. Software

W początkach występowania sterowników PLC na rynku do implementacji algorytmu w sterowniku (programowania sterownika) wykorzystywane były programatory. Te specjalne, dedykowane do konkretnego sterownika urządzenia pozwalały na zapis programu, jego korektę, testowanie oraz wgranie do sterownika PLC. W miarę rozwoju urządzeń elektronicznych rolę programatorów przejęły komputery osobiste (najczęściej typu notebook) wyposażone w odpowiednie oprogramowanie, tzw. software. Podstawowym celem software'u jest, podobnie jak w przypadku programatorów, przekształcenie napisanego programu z języka programisty na język maszynowy sterownika PLC. Ponadto, software umożliwia parametryzację sterownika, jego modułów wejść i wyjść oraz obsługę serwisową i diagnostyczną.

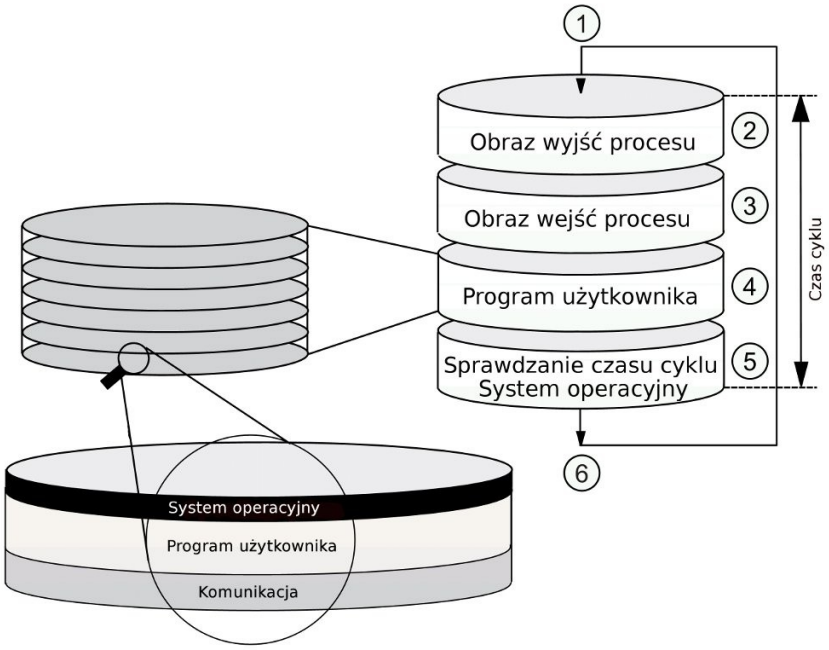
Przykładami software'u są:

- STEP7 – do programowania sterowników firmy Siemens z serii S7-300 i S7-400,
- VisiLogic – do programowania sterowników firmy Unitronics z serii Vision,
- FST – do programowania sterowników firmy Festo z serii FEC Standard i Compact oraz CPX-FEC,
- CoDeSys – uniwersalne oprogramowanie firmy 3S-Software do programowania wybranych sterowników [38] zgodnie z normą PN-EN 61131-3.

Ze względu na wygodę programisty, oraz tendencję do separacji algorytmu sterowania od konkretnego sterownika na etapie programowania, w dzisiejszych sterownikach powszechnie stosuje się nazwy symboliczne oraz adresy absolutne. Dzięki takiemu rozgraniczeniu możliwe jest napisanie programu odwołując się do zdefiniowanych przez programistę nazw symbolicznych, a dopiero przed implementacją w konkretnym sterowniku przyporządkowanie poszczególnym nazwom adresów absolutnych, a więc konkretnych wejść/wyjść sterownika PLC.

5.4. Zasada przetwarzania informacji

Opracowując zasady automatycznego generowania równania schematowego oraz pisząc programy sterujące (użytkownika) istotną rolę odgrywa znajomość zasad pracy i przetwarzania informacji przez sterowniki PLC. Uproszczony model pracy sterownika przedstawiono na rys. 5.5, zaś w tabeli 5.1 zawarto opis poszczególnych faz.



Rys. 5.5. Sekwencja wykonywania cyklicznego programu [24]

Istotną cechą sterowników PLC jest to, że są to układy synchroniczne. Dlatego też w trakcie cyklicznego wykonywania programu użytkownika, jednostka centralna wymaga spójnych i stabilnych sygnałów procesowych (wejściowych/wyjściowych). By to zapewnić, sygnały procesowe nie są odczytywane bezpośrednio z modułów wejść i wyjść, ale z obszaru pamięci jednostki centralnej w którym przechowywane są tzw. obrazy procesowe wejść i wyjść. W trakcie przetwarzania programu użytkownika jednostka centralna odczytuje te obrazy, a także zapisuje wyniki operacji na tych obrazach. Dopiero w kolejnym cyklu obrazy procesu wyjść są kopiowane do modułów wyjściowych (faza 2), a sygnały z modułów wejściowych są kopiowane do obrazu procesu wejść (faza 3).

Tabela 5.1. Wykonanie cykliczne programu [24]

Faza	Sekwencja
1	Inicjalizacja kontroli czasu cyklu przez system operacyjny
2	Kopiowanie wartości obrazu procesu wyjść do modułów wyjściowych
3	Odczyt statusu wejść z modułów wejściowych i odświeżenie obraz procesu wejść
4	Wykonanie programu użytkownika w odpowiednim przydziale czasowym oraz instrukcji programowych
5	Wykonanie kolejki zadań, np. ładowanie i kasowanie bloków, komunikacja, diagnostyka, itp.
6	Powrót na początek cyklu

6. JĘZYKI PROGRAMOWANIA STEROWNIKÓW PLC

Przed pojawieniem się swobodnie programowalnych sterowników logicznych ich funkcje realizowane były przez układy mechaniczne, elektryczne, elektroniczne lub pneumatyczne. W przypadku każdego układu zajmowali się tym różni specjaliści, dysponujący odmiennym podejściem i narzędziami. W momencie pojawienia się na rynku sterowników PLC pojawiły się także trudności dotyczące języka ich programowania. Uniwersalne w użyciu sterowniki PLC znajdowały zastosowanie w różnych branżach przemysłowych. Ich programowaniem zajmowali się więc ludzie o odmiennym podejściu do zadań, dysponujący wiedzą i doświadczeniem specyficznym dla dotychczasowego sposobu realizacji układów sterowania w ich branży. Chcąc przekonać ich wszystkich do stosowania sterowników PLC zdecydowano o zaimplementowaniu dotychczas stosowanych przez poszczególne branże metod realizacji układów sterowania w postaci języków programowania sterowników PLC. Spowodowało to powstanie wielu, mniej lub bardziej zbliżonych do siebie, języków programowania. W wyniku wielu lat rozwoju sterowników PLC oraz prac normalizacyjnych udało się stworzyć standard definiujący pięć podstawowych języków programowania sterowników PLC. Standard ten zawarto w normie PN-EN 61131-3, a definiuje on następujące języki:

- LD (ang. *Ladder Diagram*) – język wywodzący się z układów stykowo-przekaźnikowych,
- FBD (ang. *Function Block Diagram*) – język wywodzący się z logicznych bloków funkcyjnych (bramek logicznych) stosowanych do budowy pierwszych komputerów,
- IL (ang. *Instruction List*) – język wywodzący z języków programowania niskiego poziomu (assembler) związanych z programowaniem i tworzeniem kodu maszynowego dla procesorów.
- ST (ang. *Structured Text*) – język wywodzący się z języków programowania wyższego poziomu (np. Fortran, Pascal, Basic, C, C++).
- SFC (ang. *Sequential Function Chart*) – język wywodzący się z sieci Petriego, zbliżony do metody modelowania procesów i zarazem języka Grafcet stosowanego w sterownikach firmy Telemecanique.

Oprócz wymienionych języków programowania producenci sterowników stosują także swoje języki programowania oparte na językach z normy. Autorskie języki programowania przeważnie charakteryzują się odmiennym wyglądem graficznym poszczególnych elementów składowych oraz możliwością obsługi funkcji specyficznych dla danego sterownika nie zawartych w normie PN-EN 61131-3.

Choć języki programowania odgrywają istotną rolę w procesie programowania sterownika, to jednak stopień „mądrości" sterownika PLC zależy od stopnia przewidywania rzeczywistości przez programistę. Sterowniki PLC, niezależnie od ich możliwości obliczeniowych, w gruncie rzeczy działają na zasadzie „jeśli... to..." (ang. *IF...THEN...*), wykonywanej cyklicznie z określoną częstotliwością.

6.1. Elementy wspólne języków programowania

Poszczególne języki programowania różnią się składnią, gdyż wywodzą się z różnych branż. W swej istocie ich wspólnym celem jest reprezentacja i zapis algorytmu sterowania. Dlatego też norma PN-EN 61131-3 wyszczególniła elementy wspólne poszczególnych języków oraz ich odmienną składnię. Do elementów wspólnych zaliczyć możemy m.in. funkcje, typy danych oraz liczniki czasu. I choć w poszczególnych językach funkcje realizowane mogą być przez odmienną implementację (różny sposób zapisu i wygląd bloczków) to zawsze efekt działania jest ten sam.

6.1.1. Funkcje

W tabeli 6.1 przedstawiono wybrane funkcje występujące w językach LD, FBD, IL i ST.

Tabela 6.1. Wybrane funkcje języków LD, FBD, IL i ST wg PN-EN 61131-3

Funkcja	LD	FBD	IL	ST
Wartość bezwzględna	FBD[1]	ABS		
Arcus Cosinus	FBD[1]	ACOS		
Dodawanie	FBD[1]	ADD		+
Iloczyn boolowski	połączenie szeregowe	AND		AND lub &
Arcus Sinus	FBD[1]	ASIN		
Arcus Tangens	FBD[1]	ATAN		
Cosinus	FBD[1]	COS		
Dzielenie	FBD[1]	DIV		/
Porównanie	FBD[1]	EQ		=
Eksponenta	FBD[1]	EXP		
Potęgowanie	FBD[1]	EXPT		
>=	FBD[1]	GE		>=
>	FBD[1]	GT		>
<=	FBD[1]	LE		<=
<	FBD[1]	LT		<
logarytm naturalny	FBD[1]	LN		
logarytm o podstawie 10	FBD[1]	LOG		
zwraca większą wartość z dwóch podanych	FBD[1]	MAX		
zwraca mniejszą wartość z dwóch podanych	FBD[1]	MIN		
zwraca całkowitoliczbową resztę z dzielenia	FBD[1]	MOD		
przypisanie wartości jednej zmiennej innej zmiennej odpowiedniego typu	FBD[1]	MOVE		
mnożenie	FBD[1]	MUL		*
Zwraca wartość TRUE gdy argumenty są nierówne	FBD[1]	NE		<>
Bitowe zanegowanie	styk lub cewka zanegowana	NOT lub negowanie wejść/wyjść bloków	NOT	
logiczne lub	połączenie równoległe	OR		

Wybór jednej z dwóch zadeklarowanych wartości w zależności od stanu wejścia	FBD[1]	SEL	
Sinus	FBD[1]	SIN	
pierwiastek	FBD[1]	SQRT	
odejmowanie	FBD[1]	SUB	-
tangens	FBD[1]	TAN	
alternatywa wykluczająca	FBD[1]	XOR	

[1] – funkcja może być użyta poprzez wywołanie bloku FBD

Poza przedstawionymi w tabeli 6.1 funkcjami dostępny jest również szereg funkcji odnoszących się do typów danych. Funkcje te pozwalają dokonywać konwersji pomiędzy danymi różnych typów.

6.1.2. Typy danych

Każda zmienna używana w programie wykonywanym przez sterownik musi mieć ściśle określony typ. Najistotniejsze typy zmiennych, zdefiniowane w normie PN-EN 61131-3, przedstawiono w tabeli 6.2.

Tabela 6.2. Wybrane typy danych wg PN-EN 61131-3

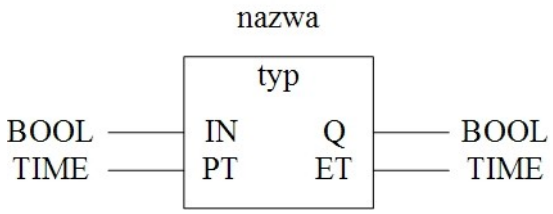
Typ	Nazwa	Dolny zakres	Górny zakres	Zajmowana pamięć w bitach
BOOL	ciąg bitów – 1 bit	0	1	1
BYTE	ciąg bitów – 8 bitów	0	255	8
WORD	ciąg bitów – 16 bitów	0	65535	16
DWORD	ciąg bitów – 32 bity	0	4294967295	32
INT	całkowita	-32768	32767	16
SINT	krótka całkowita	-128	127	8
USINT	krótka całkowita bez znaku	0	255	8
DINT	podwójna całkowita	-2147483648	2147483647	32
LINT	długa całkowita	-2^{63}	$2^{63}-1$	64
UINT	całkowita bez znaku	0	65535	16
REAL	rzeczywista / zmiennoprzecinkowa	-10^{38}	10^{38}	32
TIME	czas trwania			zależna od implementacji
STRING	łańcuch znaków	1 znak	255 znaków	8

6.1.3. Liczniki czasu

Licznik czasu (ang. *Timer*) jest elementem sygnalizującym odliczenie zadanego czasu. Norma PN-EN 61131-3 definiuje trzy rodzaje liczników czasu. Każdy z typów może być reprezentowany przez bloczek przedstawiony na rys. 6.1, gdzie:

- typ – określa rodzaj licznika czasu:
  - TP (ang. *Pulse Timer*) – załączany impulsem,

  - TON (ang. *Timer On-Delay*) – załączany z opóźnieniem,
  - TOF (ang. *Timer Off-Delay*) – wyłączany z opóźnieniem.
- nazwa – określa licznik czasu,
- IN (ang. *Input*) – wejście sygnału (zbocza narastającego) wymuszającego odliczanie czasu (zmienna typu Bool),
- PT (ang. *Preset Time*) – nastawiony do odliczenia czas (zmienna typu Time, zapis w formacie T#5s dla 5s),
- Q (ang. *Quit*) – wyjście sygnału (logiczna '1') sygnalizującego odliczenie zadanego czasu (zmienna typu Bool),
- ET (ang. *Estimate Time*) – obliczony aktualny czas od momentu aktywacji licznika czasu (zmienna typu Time).



Rys. 6.1. Schemat bloku licznika czasu wg PN-EN 61131-3

Poszczególne typy liczników różnią się między sobą wymaganym przebiegiem sygnału wymuszającego oraz zboczem od którego pojawienia się następuje odliczanie zadanego czasu. Przebiegi liczników czasu, zdefiniowanych w normie PN-EN 61131-3, przedstawiono na rys. 6.2, 6.3 i 6.4.

Przedstawiona powyżej reprezentacja graficzna znajduje zastosowanie w językach graficznych – LD i FBD. W przypadku języków tekstowych (IL, ST) implementacja licznika czasu realizowana jest w następujący sposób:
Deklaracja licznika typu TON o nazwie TONInst
TONInst : TON ;

Odwołanie się do licznika, jego wyzwolenie i użycie sygnału z wyjścia Q

Przykład w języku IL:

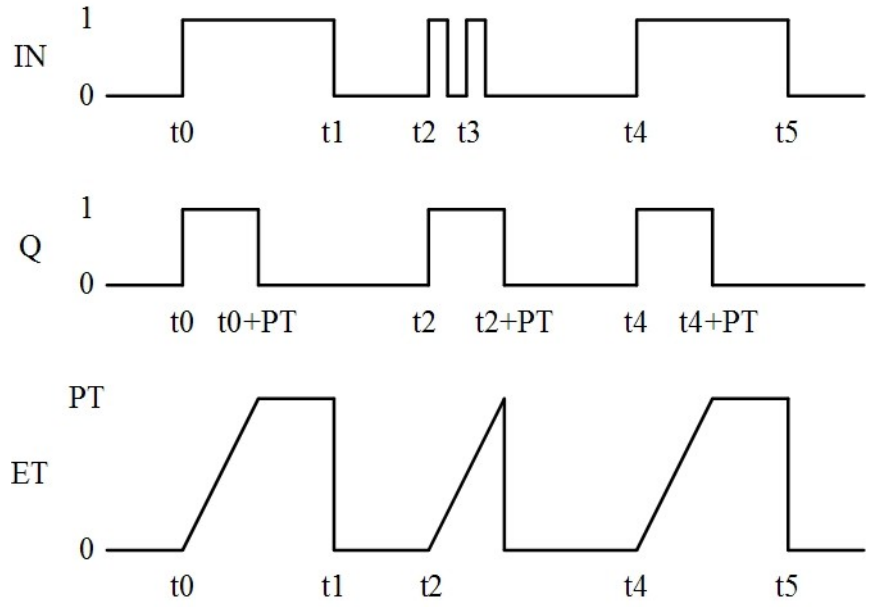
```
CAL TONInst(IN := VarBOOL1, PT := T#5s)
LD TONInst.Q
ST VarBOOL2
```

Po pojawieniu się i trwaniu sygnału *VarBOOL1* następuje uruchomienie odliczania przez licznik *TONInst* czasu 5 sekund. Po ich upływie logiczny stan 1 wyjścia *Q* licznika jest ładowany do akumulatora, którego nowa wartość jest następnie przepisywana do zmiennej *VarBOOL2*.

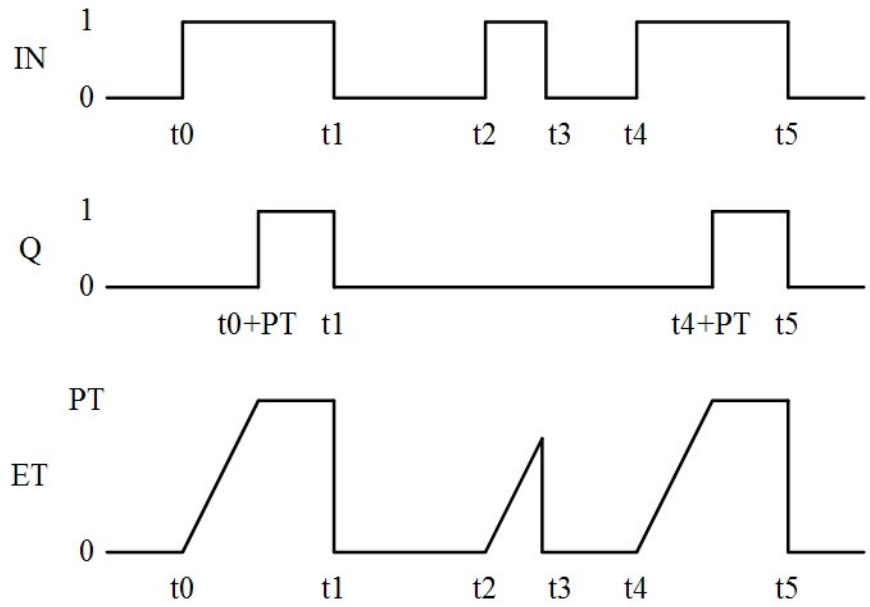
Przykład w języku ST:

```
TONInst(IN := VarBOOL1, PT:= T#5s);
IF TONInst.Q=1 THEN
VarBOOL2:=1;
ELSE
VarBOOL2:=0;
END_IF;
```

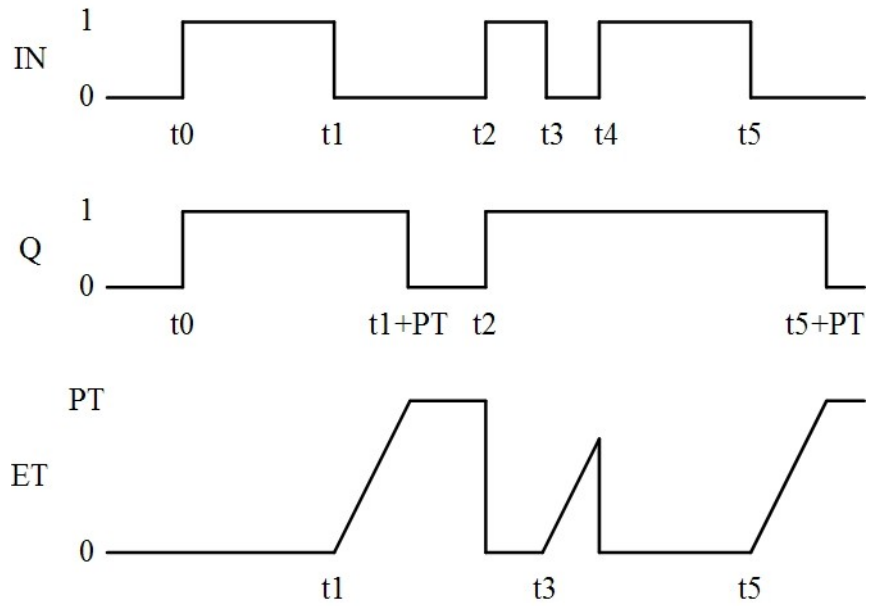
Po pojawieniu się i trwaniu sygnału *VarBOOL1* następuje uruchomienie odliczania przez licznik *TONInst* czasu 5 sekund. Cały czas jest także sprawdzana instrukcja warunkowa. Po upływie zadanego czasu instrukcja warunkowa jest prawdą, więc zmiennej *VarBOOL2* przypisywany jest stan logiczny 1.



Rys. 6.2. Diagram funkcjonalny licznika czasu TP wg PN-EN 61131-3



Rys. 6.3. Diagram funkcjonalny licznika czasu TON wg PN-EN 61131-3



Rys. 6.4. Diagram funkcjonalny licznika czasu TOF wg PN-EN 61131-3

6.2. Języki programowania wg normy PN-EN 61131-3

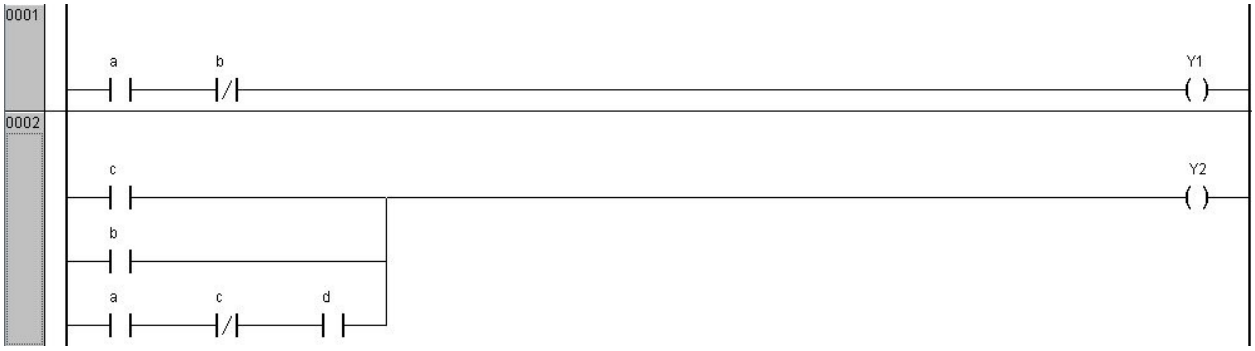
Norma PN-EN 61131-3 o tytule „Sterowniki programowalne - Część 3: Języki programowania" zawiera opis składni i semantyki języków programowania przeznaczonych do sterowników programowalnych. Zawarte w normie języki to LD, FBD, IL, ST i SFC. Ich krótki opis przedstawiono poniżej.

6.2.1. Język LD

LD (ang. *Ladder Diagram*) jest językiem graficznym wywodzącym się ze schematu stykowo-przekaźnikowego. Język LD składa się z występujących jedna nad drugą linii (ang. *networks*), które ograniczone są z lewej strony dodatnią linią zasilającą (VDC), z prawej zaś ujemną linią zasilającą (0VDC) będącą potencjałem odniesienia. Na liniach pomiędzy liniami zasilającymi umieszcza się obwód składający się ze styków, cewek i bloków FBD. Styki i bloki FBD umieszczane są od lewej do prawej strony, zaś cewki po prawej stronie. Dzięki takiemu podłączeniu cewki zawsze zasilane są potencjałem odniesienia, a potencjał dodatni doprowadzany jest po spełnieniu warunków logicznych zapisanych za pomocą styków i bloków FBD. Każdy styk może być skojarzony z komórką pamięci lub fizycznym wejściem, a cewka z komórką pamięci lub fizycznym wyjściem sterownika. Wszechstronność i elastyczność języka LD uzyskiwana jest dzięki możliwości implementowania bloków FBD, które realizować mogą funkcje określone w tabeli 6.1.

Przykładowy algorytm, reprezentowany układem równań 6.1 zapisany za pomocą języka LD, przedstawiono na rys. 6.5.

$$F(Y)=\sum\begin{cases}a\cdot\bar{b}=Y_{1,}\\ c+b+(a\cdot\bar{c}\cdot d)=Y_{2,}\end{cases} \tag{6.1}$$

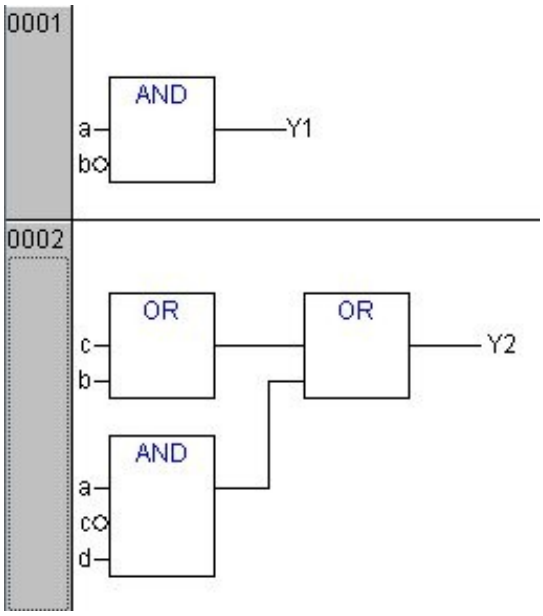


Rys. 6.5. Układ równań (6.1) zapisany za pomocą języka LD

6.2.2. Język FBD

Języka FBD (ang. *Function Block Diagram*) jest językiem graficznym. Jego struktura oparta jest o układ sieci, w których każda operacja logiczna lub arytmetyczna reprezentowana jest przez blok funkcyjny. Wynik operacji obliczonej przez jeden blok może być jednocześnie wartością wejściową kolejnego bloku. Zestawienie najczęściej wykonywanych operacji języka FBD zawarto w tabeli 6.1.

Przykładowy algorytm, reprezentowany układem równań 6.1 zapisany za pomocą języka FBD, przedstawiono na rys. 6.6.



Rys. 6.6. Układ równań (6.1) zapisany za pomocą języka FBD

6.2.3. Język IL

Język IL (ang. *Instruction List*) jest językiem testowym. Jego składnia tworzona jest przez szereg instrukcji z których każda znajduje się w nowej linii oraz zawiera:

- etykietę zakończoną dwukropkiem (opcjonalnie),
- operator,
- modyfikator (opcjonalnie),
- zmienne lub stałe oddzielone przecinkami (tzw. operandy),
- komentarz (opcjonalnie).

Specyficzne operatory języka IL przedstawiono w tabeli 6.3. W tabeli 6.1 zawarto inne

operatory możliwe do stosowania w języku IL, zaś tabela 6.4 przedstawia modyfikatory operatorów. Działania wykonywane przez poszczególne operatory odnoszą się do tzw. WYNIKU – obszaru pamięci sterownika przechowującego aktualną wartość obliczeń. Dla lepszego zrozumienia istoty języka IL przedstawiono przykładową instrukcję w postaci:

$$\begin{array}{c} LD\ 1 \\ ADD\ 2 \\ ST\ a \end{array} \tag{6.2}$$

Powyższy algorytm należy czytać w następujący sposób - w pierwszej kolejności WYNIKOWI przyporządkowana jest wartość 1, następnie do wartości tej dodawana jest wartość 2, a na koniec wartość WYNIK przypisywana jest zmiennej *a*. W rezultacie działania takiej instrukcji zmiennej *a* zostanie przypisana wartość 3.

Przykładowy algorytm, reprezentowany układem równań 6.1 zapisany za pomocą języka IL, przedstawiono na rys. 6.7.

Tabela 6.3. Specyficzne operatory języka IL i ich znacznie oraz dostępne opcjonalne modyfikatory wg PN-EN 61131-3

Operator	Modyfikator	Znaczenie
LD	N	Uczyń aktualny wynik równy operandowi
ST	N	Przypisz aktualny wynik zmiennej
S		Operand przyjmuje wartość 1
R		Operand przyjmuje wartość 0
JMP	C, CN	Skocz do etykiety
CAL	C, CN	Wywołanie bloku funkcyjnego
RET	C, CN	Porzuć POU (ang. *Program Organization Unit*) – jednostkę organizacyjną programu i powróć do punktu wywołania

Tabela 6.4. Modyfikatory operatorów języka IL i ich znaczenie wg PN-EN 61131-3

Modyfikator	Modyfikowany operator	Znaczenie
C	JMP, CAL, RET	Instrukcja wykonywana jest tylko wtedy, gdy wynik poprzedniego wyrażenia jest prawdą (*ang. TRUE*).
CN	JMP, CAL, RET	Instrukcja wykonywana jest tylko wtedy, gdy wynik poprzedniego wyrażenia jest nieprawdą (*ang. FALSE*).
N		Negacja zmiennej lub wyniku

```
0001    LD    a
0002 ANDN    b
0003 ST    Y1
0004
0005    LD    c
0006 OR    b
0007 OR(    a
0008 ANDN    c
0009 AND    d
0010 )
0011 ST    y2
```

Rys. 6.7. Układ równań (6.1) zapisany za pomocą języka IL

6.2.4. Język ST

Język ST (ang. *Structured Text*) jest językiem testowym składającym się z serii instrukcji (tabela 6.1) oraz pętli (tabela 6.5) pochodzących z języków wysokiego poziomu.

Przykładowy algorytm, reprezentowany układem równań (6.1) zapisany za pomocą języka ST, przedstawiono na rys. 6.8.

Tabela 6.5. Instrukcje i pętle języka ST wg PN-EN 61131-3

Instrukcja	Znaczenie	Przykład
Przyporząd-kowanie	Zastępuje obecną wartość zmiennej A wartością zmiennej B	*A:=B*
RETURN	*POWRÓĆ* Służy do wcześniejszego opuszczenia funkcji, bloku funkcyjnego lub programu	*IF a=6 THEN* *RETURN;* *END_IF;*
IF	*JEŚLI* Służy do sprawdzania warunku i w zależności od tego wykonania polecenia.	*IF temp<17* *THEN heating_on := TRUE;* *ELSE heating_on := FALSE;* *END_IF;*
CASE	*W PRZYPADKU GDY* Służy do wyboru polecenia w zależności od wartości sprawdzanej zmiennej	*CASE INT1 OF* *1,5: BOOL1 := TRUE;* *BOOL3 := FALSE;* *2: BOOL2 := FALSE;* *BOOL3 := TRUE;* *10..20: BOOL1 := TRUE;* *BOOL3:= TRUE;* *ELSE* *BOOL1 := NOT BOOL1;* *BOOL2 := BOOL1 OR BOOL2;* *END_CASE;*
FOR	*DLA* Pętla wykonywana jest ściśle określoną liczbę razy. Do obsługi pętli użyta jest zmienna Counter, której przyporządkowana jest wartość 1. Wartość ta jest zwiększana o 1 przy każdym wykonaniu pętli.	*FOR Counter:=1 TO 5 DO* *Var1:=Var1*2;* *END_FOR;* Pętla wykonywana jest pięciokrotnie. Jeśli początkowo zmienna *Var1* ma wartość 1, to po wykonaniu powyższej pętli zmienna *Var1* ma wartość 32

WHILE	*DOPÓKI* Pętla wykonywana jest podobnie jak pętla FOR z tą różnicą, że warunkiem kończącym może być dowolne wyrażenie logiczne. Warunek sprawdzany jest na początku pętli.	*WHILE Counter<>0 DO* *Var1 := Var1*2;* *Counter := Counter-1;* *END_WHILE* Dopóki zmienna *Counter* jest różna od zera mnóż wartość zmiennej *Var1* razy dwa i zmniejszaj wartość zmiennej *Counter* o jeden.
REPEAT	*POWTARZAJ dopóki* Pętla jest podobna do WHILE z tą różnicą, że warunek jest sprawdzany dopiero po wykonaniu pojedynczego przebiegu pętli. Oznacza to, że pętla będzie działać co najmniej raz, niezależnie od warunku.	*REPEAT* *Var1 := Var1*2;* *Counter := Counter-1;* *UNTIL* *Counter=0* *END_REPEAT;*
EXIT	*WYJŚCIE* Instrukcja EXIT jest używana zakończenia powtórzeń pętli FOR, WHILE lub REPEAT zanim warunek zakończenia wykonywania pętli zostanie spełniony.	*FOR J:=1 TO 3 DO* *IF FLAG THEN EXIT;* *END_IF;* *SUM:=SUM + J;* *END_FOR;*

```
0001 IF a AND NOT b THEN Y1:=1;
0002 ELSE Y1:=0;
0003 END_IF;
0004
0005 IF c OR b OR (a AND NOT c AND d) THEN Y2:=1;
0006 ELSE Y2:=0;
0007 END_IF;
```

Rys. 6.8. Układ równań (6.1) zapisany za pomocą języka ST

7. ZAŁOŻENIA PROGRAMOWE PRACY

7.1. Cel pracy

Celem pracy jest opracowanie nowych zasad syntezy równania schematowego metody Grafpol. Dotychczas synteza równania schematowego oparta była na regułach przyjętych w metodzie MTS (Metoda Transformacji Sieci). Synteza równania schematowego metodą MTS wymagała graficznej analizy sygnałów wejściowych i wyjściowych projektowanych układów sterowania, dlatego jej praktyczne zastosowanie było pracochłonne i uciążliwe.

W celu opracowania nowych zasad syntezy równania schematowego sekwencyjnych algorytmów sterowania opracowano teoretyczne podstawy określające zasady zapisu i kasowania elementarnych komórek pamięci, które muszą być stosowane w procedurach sekwencyjnych oraz zastosowania informacji przechowywanych w tych komórkach do syntezy równania schematowego algorytmów sekwencyjnych.

Realizacja założonego celu pracy wymagała dokonania:

- analizy dotychczas stosowanych metod syntezy równania schematowego,
- określenia zalet i wad dotychczasowych metod,
- opracowania i uogólnienia warunków zapisu, kasowania i użycia elementarnych komórek pamięci.

7.2. Tezy pracy

Równanie schematowe jest uniwersalnym, analitycznym modelem matematycznym sekwencyjnego algorytmu sterowania, w którym jest uwzględniana pamięć.

W metodzie Grafcet oraz języku programowania SFC warunki zapisu i kasowania elementarnych komórek pamięci, niezbędnych do realizacji sekwencyjnych algorytmów sterowania, wyznaczane są bez analizy przebiegu sygnałów wejściowych układu sterowania. Dlatego też istnieje duże prawdopodobieństwo, że możliwość taka istnieje także w metodzie Grafpol.

W związku z tym tezy niniejszej pracy zostały sformułowane w następujący sposób:

TEZA 1

Równanie schematowe procedur sekwencyjnych można wyznaczyć w oparciu o algorytm sterowania, który reprezentuje sieć Grafpol GS przedstawiająca zewnętrzne sygnały układu sterowania (wejściowe i wyjściowe), bez konieczności analizy tych sygnałów przedstawianych w postaci graficznej.

TEZA 2

Równanie schematowe wyznaczone za pomocą nowych zasad syntezy równania schematowego metody Grafpol może stanowić podstawę do realizacji dowolnych sekwencyjnych układów sterowania realizowanych za pomocą elementów stykowo-przekaźnikowych, pneumatycznych oraz jako zapisu programu sterownika PLC.

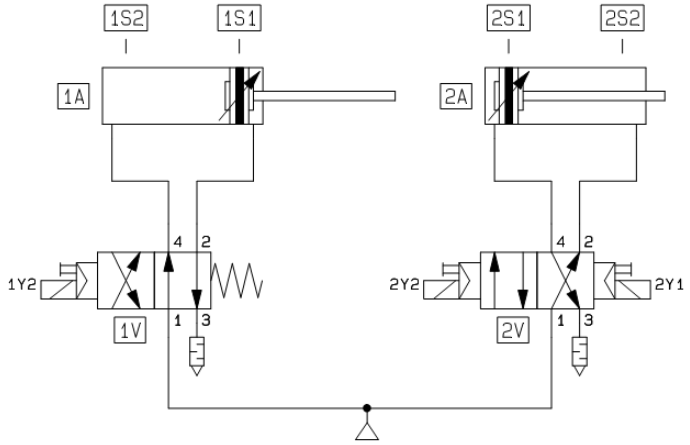
Proces udowodnienia postawionych tez możliwy jest poprzez weryfikację opracowanych za pomocą stworzonych zasad i reguł równań schematowych wybranych procedur na modelu komputerowym układu sterowania oraz procesu. Procedura udowadniania tez będzie przebiegała według następujących etapów:

- opracowanie równania schematowego wybranej procedury według nowych reguł,
- implementacja opracowanego równania schematowego w rzeczywistym lub zamodelowanym w komputerze układzie sterowania,
- weryfikacja zgodności wygenerowanego równania schematowego (algorytmu sterowania ze zrealizowaną pamięcią) z algorytmem procesu.

8. NOWE ZASADY SYNTEZY RÓWNANIA SCHEMATOWEGO METODĄ GRAFPOL

W dotychczasowym sposobie syntezy równania schematowego w metodzie Grafpol wykorzystywana była Metoda Transformacji Sieci (MTS). Używając tej metody otrzymuje się minimalną liczbę komórek pamięci, które należy użyć. Metoda ta jest jednak bardzo pracochłonna. Dlatego też prowadzone były prace w kierunku zastąpienia MTS w metodzie Grafpol rozwiązaniem szybszym i prostszym w praktycznej realizacji. W wyniku przeprowadzonych prac opracowano rozwiązanie zaprezentowane w niniejszym rozdziale. W celu lepszego wytłumaczenia opracowanych zasad zilustrowano je przykładowymi algorytmami, odwołującymi się do schematu funkcjonalnego przedstawionego na rys. 8.1.

W wyniku prowadzonych prac ujednolicono także stosowane w metodzie Grafpol oznaczenia. Oznaczenia te występują na schemacie funkcjonalnym, dokumentacji technicznej oraz sieciach reprezentujących algorytm procesu i algorytm sterowania.



Rys. 8.1 Schemat funkcjonalny dwóch napędów pneumatycznych

8.1. Stosowane oznaczenia

W opracowanym rozwinięciu metody Grafpol ujednolicono oznaczenia na schemacie funkcjonalnym zgodnie z normą PN-ISO 1219-2 „Napędy i sterowania hydrauliczne i pneumatyczne - Schematy układów". Uzasadniając wybór tej właśnie normy warto zwrócić uwagę na fakt, iż metoda Grafpol znajduje zastosowanie w sterowaniu napędami pneumatycznymi, hydraulicznymi oraz elektrycznymi, a ponadto nie istnieje jedna norma dotycząca oznaczania elementów z tych trzech grup. Dlatego też nie jest możliwe znalezienie uniwersalnych oznaczeń dla wszystkich rodzajów napędów. Ponadto poszczególne normy wykluczają się. Przykładowo norma PN-ISO 1219-2 zaleca oznaczanie czujników położenia tłoka symbolem S, np. 1S2, gdy jednocześnie norma PN-EN 61082-1:2006 "Przygotowanie dokumentów używanych w elektrotechnice - Podstawowe zasady" sugeruje oznaczanie elementów sygnalizacyjnych symbolem B.

Zgodnie z normą PN-ISO 1219-2 przyjęto następujące oznaczenia:

- iS1 – oznaczenie czujnika (także sygnału logicznej '1' z tego czujnika) określającego pierwsze położenie skrajne *i*-tego napędu A (pozycja początkowa),
- iS2 – oznaczenie czujnika (także sygnału logicznej '1' z tego czujnika) określającego drugie położenie skrajne *i*-tego napędu A.

W zbliżonej konwencji przyjęto oznaczać, nie zdefiniowane w normie, elementy sterujące napędami (cewki elektrozaworów pneumatycznych lub hydraulicznych, przekaźniki lub styczniki silników elektrycznych). W odniesieniu do cewek zaworów pneumatycznych wygląda to następująco:

- iY1 – oznaczenie cewki (także sygnału logicznej '1' tej cewki) i-tego zaworu V odpowiadającej za takie przesterowanie zaworu, z uwzględnieniem połączenia między zaworem a napędem, iż następuje powrót tłoka i-tego siłownika A do pozycji początkowej,
- iY2 – oznaczenie cewki (także sygnału logicznej '1' tej cewki) i-tego zaworu V odpowiadającej za takie przesterowanie zaworu, z uwzględnieniem połączenia między zaworem a napędem, iż następuje ruch tłoka i-tego siłownika A z pozycji początkowej.

Ponadto, w przypadku zaworów lub przekaźników monostabilnych, stosowane są następujące oznaczenia:

- iY2 – podanie napięcia na cewkę i-tego zaworu V lub zaciski przekaźnika monostabilnego trwające do momentu zdjęcia prądu z tej cewki,
- $\overline{iY2}$ – zdjęcie napięcia z cewki i-tego zaworu V lub zaciski przekaźnika monostabilnego trwające do momentu ponownego podania prądu na tą cewkę.

W odniesieniu do zaworów bistabilnych podawane jest jedynie oznaczenie xY2 równoważne podaniu napięcia trwającego nie dłużej niż do chwili podania prądu na cewkę przeciwną.

W algorytmie procesu stosowane są oznaczenia:

- $iA^{(1)}$ – ruch i-tego napędu z pozycji początkowej,
- $iA^{(2)}$ - ruch i-tego napędu do pozycji początkowej.

W algorytmie sterowania stosowane są oznaczenia:

- $Y_i^{(1)-}$ zmienna sterująca odpowiadająca za ruch i-tego napędu z pozycji wyjściowej,
- $Y_i^{(2)-}$ zmienna sterująca odpowiadająca za ruch i-tego napędu w kierunku pozycji wyjściowej.

W przypadku wielokrotnych ruchów tego samego napędu z pozycji początkowej powyższe oznaczenia wyglądają następująco:

- $Y_{i\_k}^{(1)-}$ zmienna sterująca odpowiadająca za k-ty ruch i-tego napędu z pozycji wyjściowej,
- $Y_{i\_k}^{(2)-}$ zmienna sterująca odpowiadająca za k-ty ruch i-tego napędu w kierunku pozycji wyjściowej.

8.2. Sieć Grafpol GS a równanie schematowe

W dotychczasowej metodzie Grafpol, w której do realizacji pamięci stosowana była Metoda Transformacji Sieci, algorytm sterowania reprezentowany siecią Grafpol GS służył do wyznaczenia warunków opisujących zmienne sterujące Y_i. Na jego podstawie realizowana była także pamięć. Wyznaczone postaci zmiennych wyjściowych Y_i oraz warunki zapisu i kasowania elementarnych komórek pamięci przedstawione były jedynie w równaniu schematowym.

W związku z opracowaniem nowych zasad realizacji pamięci metodą Grafpol, zmianie uległ nieznacznie sposób prezentacji algorytmu sterowania za pomocą sieci Grafpol GS. Teraz, podobnie jak w MTS, sieć ta służy do wyznaczenia równania schematowego, ale ponadto, uzupełniona o zrealizowaną pamięć, jest jego wiernym odzwierciedleniem. W blokach reprezentujących kroki algorytmu sterowania umieszcza się warunki opisujące tranzycje z uwzględnionymi sygnałami pamięci oraz zmienne sterujące. Zapis i kasowanie elementarnych komórek pamięci oraz operacje na licznikach czasu są zawarte w blokach skojarzonych z blokami reprezentującymi kroki. Bloki te umieszcza się po prawej stronie (rys. 8.2) kroków. Taki zapis, zaczerpnięty częściowo z języka SFC, pozwala na czytelną prezentację algorytmu sterowania, który składa się z bloków zawierających zmienne sterujące oraz bloków skojarzonych w których zawarte są komórki pamięci oraz liczniki czasu.

Spełnienie warunku tranzycji opisanej w górnej części bloku reprezentującego krok

powoduje, że zmienna sterująca zdefiniowana w dolnej części bloku przyjmuje wartość logiczną "1". Innymi słowy na przyporządkowane jej wyjście układu sterowania podawany jest sygnał. Przykładowo z sieci Grafpol GS przedstawionej na rys. 8.2 odczytać można, że:

$$\begin{aligned} Y_3^{(1)} &= St \cdot 2S1 \cdot \overline{m_1} \cdot 1S1 \cdot \overline{m_3} \\ Y_3^{(2)} &= 3S2 \cdot \overline{m_2} \end{aligned} \tag{8.1}$$

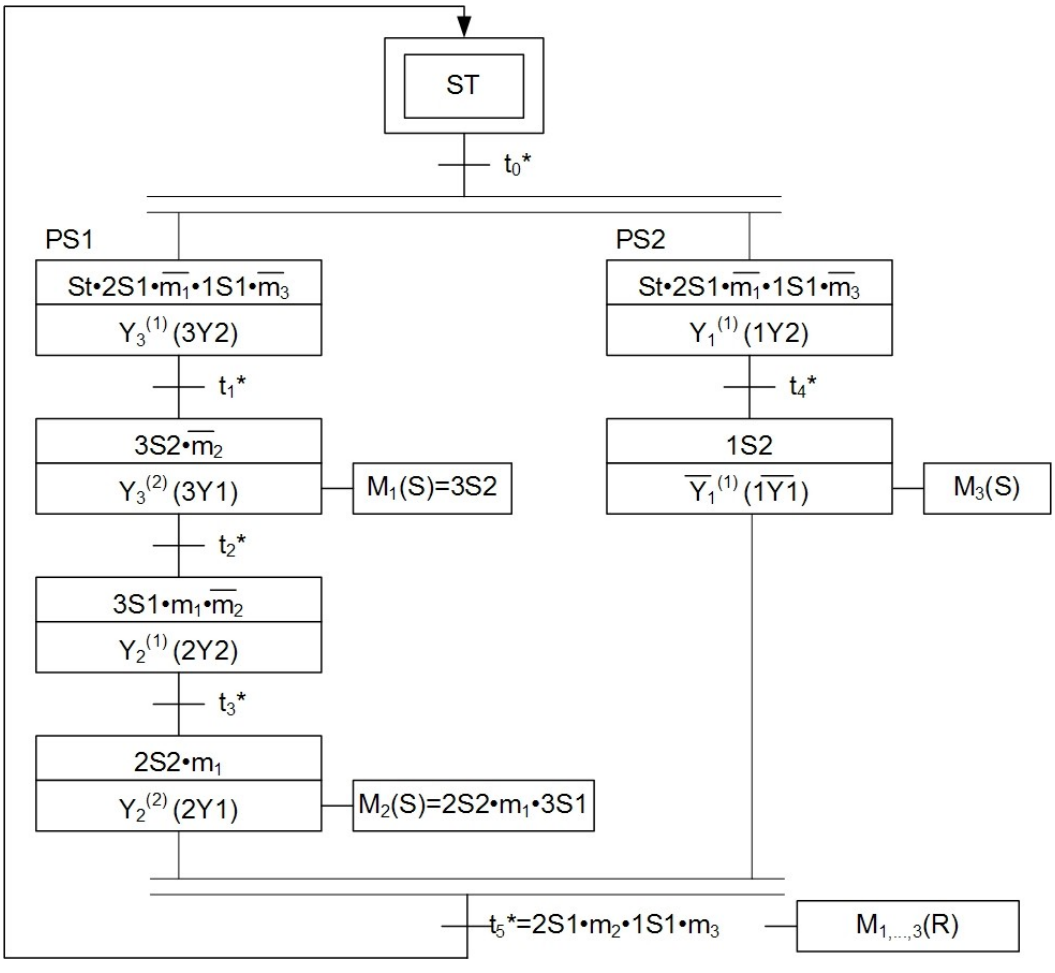
W przypadku zmiennej Y_1 zapis jest nieco inny. Ponieważ zmienna ta występuje w sieci jako zanegowana i nie zanegowana, więc jej zapis w równaniu schematowym jest następujący:

$$\begin{aligned} Y_1^{(1)} &\equiv Y_1^{(1)}(S) \equiv 1Y1(S) = St \cdot 2S1 \cdot \overline{m_1} \cdot 1S1 \cdot \overline{m_3} \\ \overline{Y_1^{(1)}} &\equiv Y_1^{(1)}(R) \equiv 1Y1(R) = 1S2 \end{aligned} \tag{8.2}$$

W blokach skojarzonych zawarte są zależności logiczne związane z elementarnymi komórkami pamięci lub licznikami czasu.

W przypadku gdy w bloku skojarzonym zdefiniowany jest zapis lub kasowanie elementarnej komórki pamięci i warunki logiczne nie zostały określone oznacza to, że obowiązują warunki określone przez tranzycję. Przypadek taki zaprezentowano na rys. 8.2, gdzie:

$$\begin{aligned} M_3(S) &= 1S1 \\ M_{1,\ldots,3}(R) &= 2S1 \cdot m_1 \cdot 1S1 \cdot m_3 \end{aligned} \tag{8.3}$$



Rys. 8.2. Algorytm sterowania zapisany za pomocą sieci Grafpol GS z wyraźnie widoczną strukturą zmiennych sterujących oraz elementarnych komórek pamięci

8.3. Procedury sekwencyjne

Procedura sekwencyjna jest podstawową procedurą spotykaną w zautomatyzowanych urządzeniach i procesach. Procedura ta składa się z poszczególnych etapów realizowanych jeden po drugim. Nowe zasady syntezy równania schematowego dla procedur sekwencyjnych przedstawiono poniżej.

Zasada 1

Liczba komórek pamięci może być określona dla procedur sekwencyjnych na podstawie zależności:

$$L=\frac{S_n}{2}+X \tag{8.4}$$

gdzie:

S_n – n-ty krok algorytmu sterowania (ostatni),

X – suma liczby przypadków $iA^{(1)},...,iA^{(2)}$, $iA^{(1)},....$, tj. gdy dany i-ty napęd wykonuje kilkukrotnie ruch z pozycji początkowej i w trakcie sekwencji powraca do pozycji początkowej, by w następnym kroku ponownie wykonać ruch z tej pozycji oraz dodatkowych komórek pamięci MT stosowanych w przypadku używania elementów odliczających czas.

Zasada 2

Zapis komórek pamięci, nazywanych komórkami sterującymi, następuje po zakończeniu wykonania kroków sterujących realizacją pierwszych etapów realizowanych przez poszczególne napędy. W algorytmie procesu etapy te oznaczane są przez $iA^{(1)}$,

gdzie:

i – numer napędu.

Stany w których następuje zapis opisują tranzycje t_i, które sygnalizują zakończenie wykonania kroku. Zależności opisujące zapis elementarnych komórek pamięci mają następujące postaci:

$$\begin{aligned}
&\boldsymbol{M_1}(S)=t_1,\\
&\boldsymbol{M_2}(S)=t_i\cdot m_1\cdot t_{i-1},\\
&\vdots\\
&\boldsymbol{M_i}(S)=t_j\cdot m_{i-1}\cdot t_{j-1},\\
&\vdots\\
&\boldsymbol{M_L}(S)=t_k\cdot m_{L-1}\cdot t_{k-1}
\end{aligned} \tag{8.5}$$

gdzie:

$\boldsymbol{M_i}$ – i-ta elementarna komórka pamięci,

m_i – sygnał wyjściowy i-tej elementarnej komórki pamięci.

Poszczególna komórka pamięci (za wyjątkiem M_1) zapisywana jest wg zależności:

$$\boldsymbol{M_i}(S)=t_j\cdot m_{i-1}\cdot t_{j-1}, \tag{8.6}$$

gdzie:

t_i – wynika z sieci Grafpol GP,

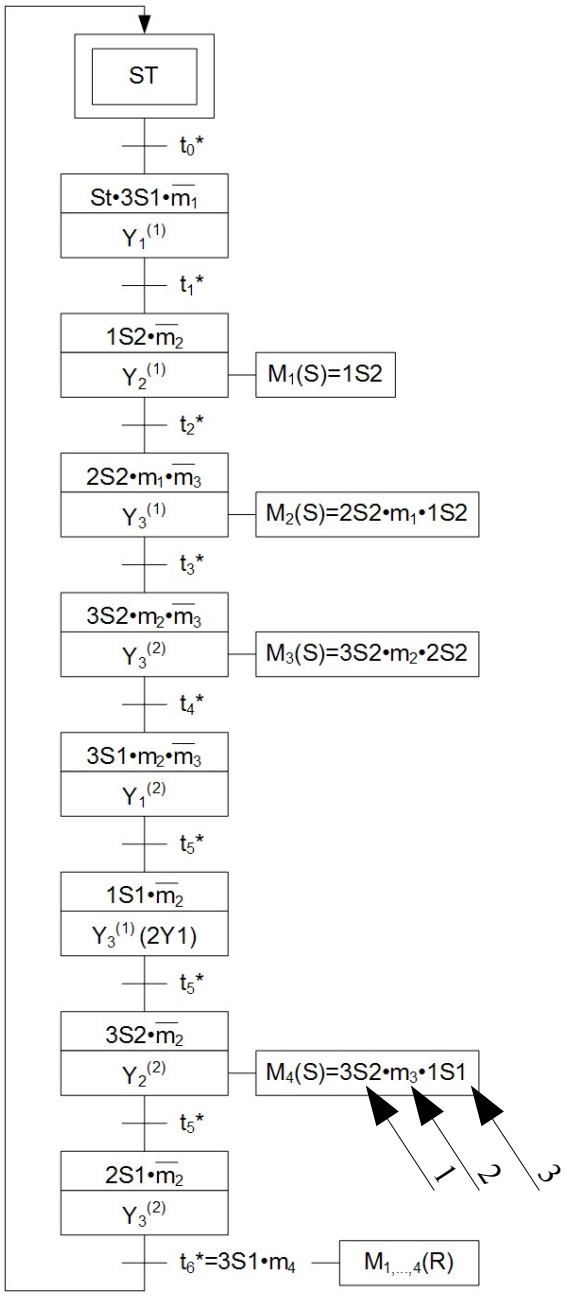
m_{i-1} – gwarantuje iż zapis kolejnej komórki pamięci (M_j) nastąpi tylko gdy zapisano wcześniejszą komórkę pamięci,

t_{i-1} – gwarantuje zapis komórki pamięci tylko gdy zakończona została realizacja poprzedniego

etapu, co ma istotne znaczenie w przypadku wielokrotnych ruchów napędów z pozycji wyjściowej.

W przypadku braku spełnienia warunku t_{i-1} (poz.3 rys. 8.3) zapis komórki pamięci M_4 dla przykładu z rys. 8.3 nastąpiłby natychmiast po zapisie komórki pamięci M_3. Warunek ten jest wymagany gdy następuje wielokrotny ruch tego samego napędu z pozycji wyjściowej, a pomiędzy tymi ruchami realizowane są jedynie ruchy powrotne napędów do pozycji wyjściowej.

Graficzną ilustrację, uzupełniającą powyższe uzasadnienie warunków zapisujących komórkę pamięci, przedstawiono na rys. 8.3.



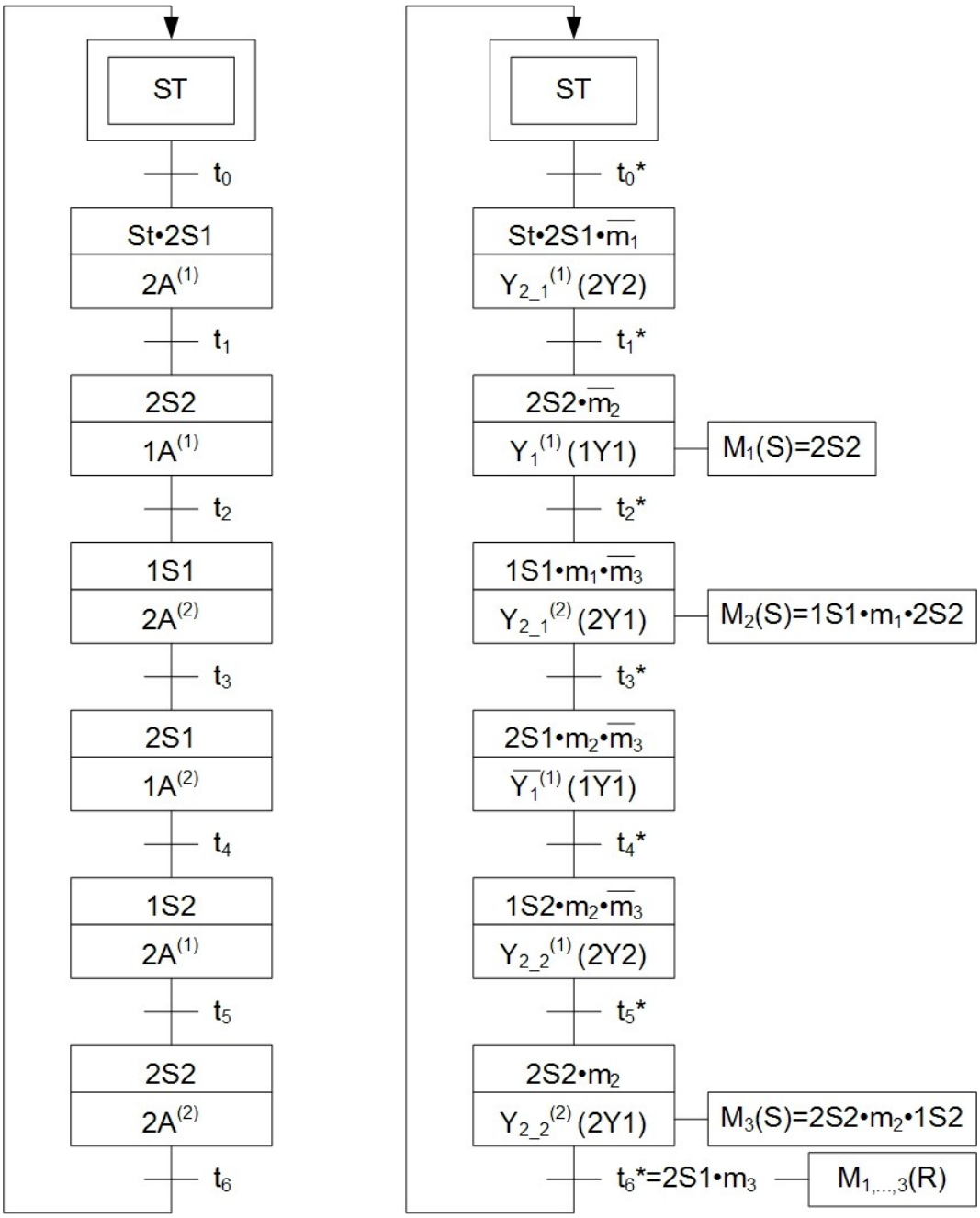
Rys. 8.3. Algorytm uzasadniający użycie poszczególnych warunków do zapisu pamięci; 1-warunek wynikający z sieci, 2-warunek zezwalający na zapis kolejnej komórki pamięci tylko gdy zapisano poprzednią komórkę pamięci, 3-warunek gwarantujący zapis komórki pamięci tylko gdy zakończony został poprzedni krok

UWAGA 1:

W przypadku wielokrotnej pracy tego samego napędu z pozycji wyjściowej każdorazowo jest zapisywana nowa elementarna komórka pamięci. W przykładowym algorytmie z rys. 8.4 są to M_1 i M_3. Ponadto w takim przypadku funkcja zmiennej wyjściowej Y_i *i*-tego napędu, wykonującego wielokrotne ruchy z pozycji wyjściowej, ma postać:

$$\begin{aligned} Y_i^{(1)} &= Y_{i\_1}^{(1)} + Y_{i\_2}^{(1)} + \ldots + Y_{i\_n}^{(1)} \\ Y_i^{(2)} &= Y_{i\_1}^{(2)} + Y_{i\_2}^{(2)} + \ldots + Y_{i\_n}^{(2)} \end{aligned} \tag{8.7}$$

gdzie:
$Y_{i\_n}^{(1)}$ – zmienna sterująca *n*-tym powtórzeniem ruchu *i*-tego napędu z pozycji wyjściowej,
$Y_{i\_n}^{(2)}$ – zmienna sterująca *n*-tym powtórzeniem ruchu *i*-tego napędu do pozycji wyjściowej.



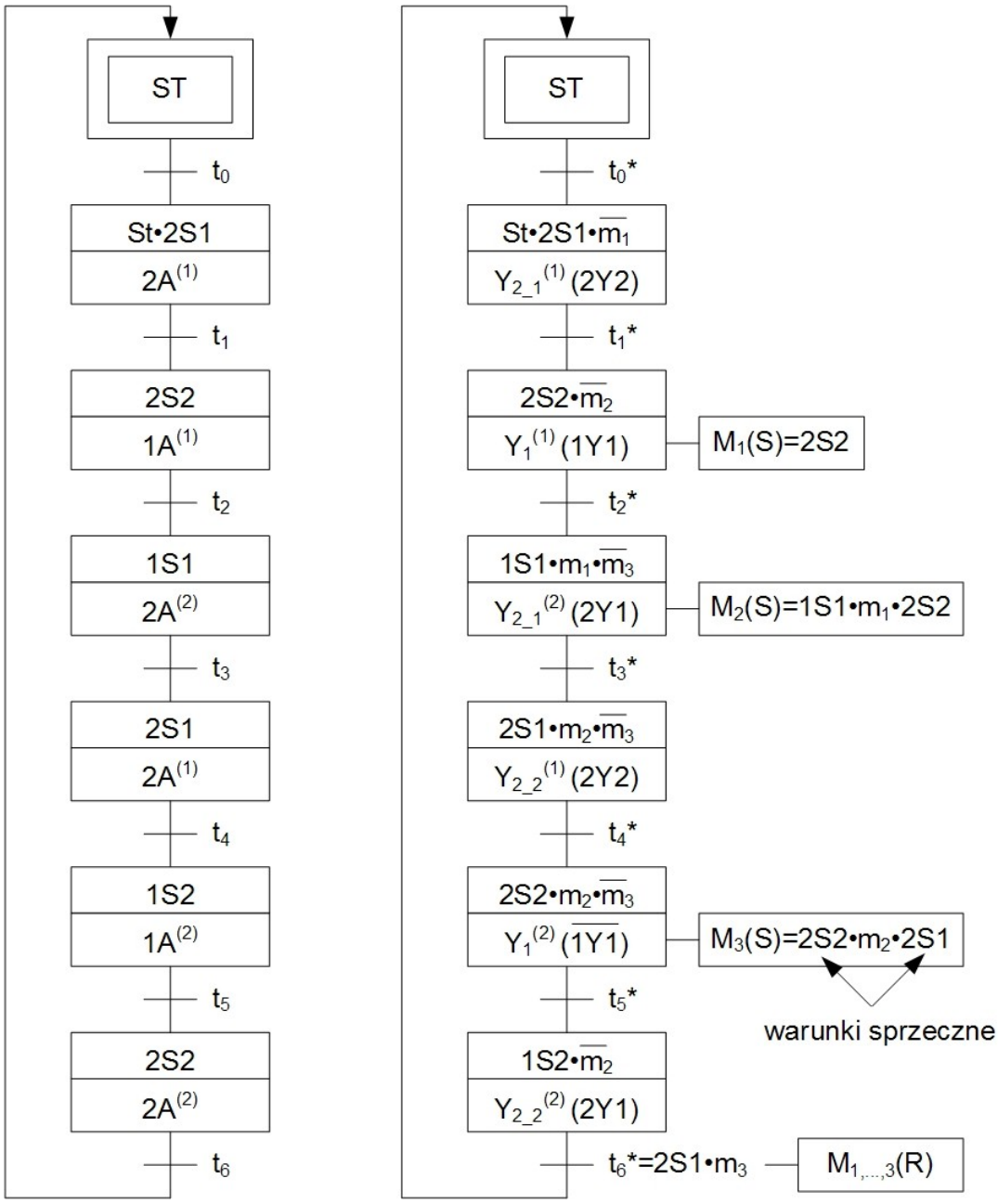
Rys. 8.4. Wielokrotne ruchy tego samego napędu z pozycji wyjściowej

UWAGA 2:

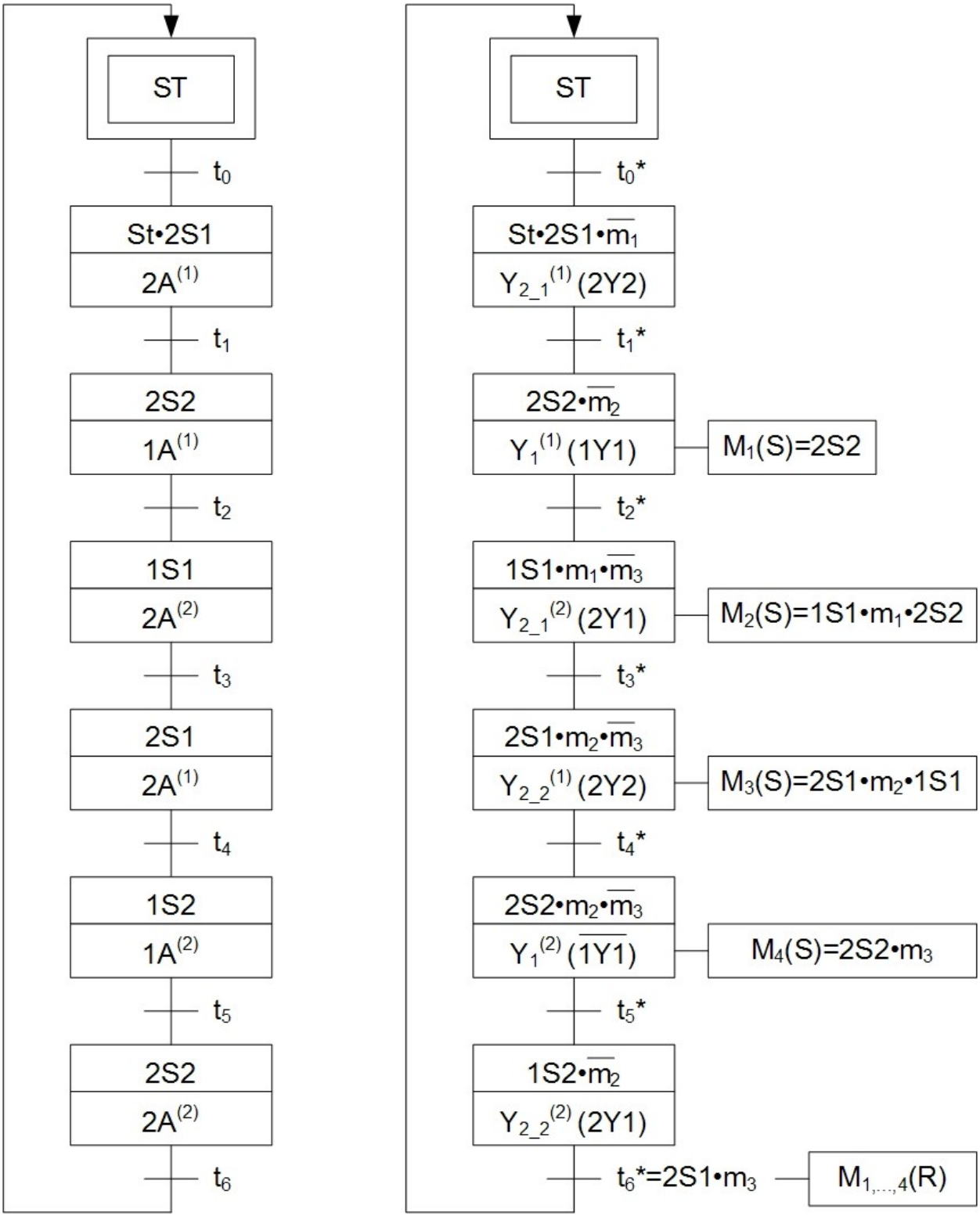
W przypadku gdy w sekwencji etapów występuje specyficzny przypadek …,$iA^{(1)}$,...,$iA^{(2)}$,$iA^{(1)}$,.... konieczne są zmiany w zapisie komórek pamięci. Dodatkowa komórka pamięci jest zapisywana po etapie $iA^{(2)}$ zgodnie z równaniem (8.6). Komórka zapisywana po drugim ruchu napędu 1A z pozycji wyjściowej zapisywana jest wg równania:

$$\boldsymbol{M}_i(S)=t_j\cdot m_{i-1}, \tag{8.8}$$

Ten specyficzny przypadek spowodowany jest sprzecznością warunków określających zapis komórki pamięci dla drugiego ruchu napędu 1A z pozycji początkowej. Ilustrację tej sytuacji przedstawiono na rys. 8.5. Na rysunku 8.6 przedstawiono poprawne rozwiązanie sprzeczności warunków z uwzględnieniem powyższej uwagi.



Rys. 8.5. Specyficzny przypadek procedury sekwencyjnej – rozwiązanie błędne



Rys. 8.6. Specyficzny przypadek procedury sekwencyjnej – rozwiązanie poprawne

Kasowanie wszystkich elementarnych komórek pamięci następuje w ostatnim stanie algorytmu sterowania, gdy spełniona jest tranzycja t_n^* o postaci:

$$t_n^* = t_n \cdot m_L \tag{8.9}$$

Tak więc zależność opisująca kasowanie wszystkich elementarnych komórek pamięci jest następująca:

$$M_{1,2,\ldots,L}(R) \equiv M_1(R) + M_2(R) + \ldots + M_L(R) = t_n \cdot m_L \tag{8.10}$$

gdzie:
t_n – tranzycja opisująca ostatni stan algorytmu sterowania

Zasada 3

Do określenia zmiennych wyjściowych Y algorytmu sterowania konieczna jest znajomość tranzycji t_i^*, w których zostały uwzględnione informacje przechowywane w komórkach pamięci. Tranzycja t_i^* jest równoważna iloczynowi tranzycji t_i i sygnału lub zanegowanego sygnału komórki lub komórek pamięci.

$$t_i^* = t_i \cdot \boldsymbol{M} \tag{8.11}$$

gdzie:

$\boldsymbol{M} = < m_i, \overline{m_i}, m_i \cdot \overline{m_{i+1}} >$

Zasada 4

Zapis komórki pamięci M_j powoduje, że:

- wszystkie tranzycje t_i^* poprzedzające tranzycję w której nastąpił zapis komórki pamięci, aż do tranzycji w której została zapisana poprzednia komórka pamięci włącznie lub tranzycji zerowej, mają postać:

$$t_i^* = t_i \cdot \overline{m_j} \tag{8.12}$$

- wszystkie tranzycje t_i^* następujące po tranzycji w której nastąpił zapis, aż do tranzycji w której została zapisana następna komórka pamięci włącznie lub ostatniej tranzycji, mają postać:

$$t_i^* = t_i \cdot m_j \tag{8.13}$$

gdzie:
$\underline{m_j}$ – sygnał j-tej komórki pamięci,
$\overline{m_j}$ – zanegowany sygnał j-tej komórki pamięci.

W przypadku napędów sterowanych zaworami monostabilnymi spełnienie tranzycji t_i^*, określającej 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).

Przykładowy algorytm ilustrujący powyższe reguły użycia sygnałów komórek pamięci przedstawiono na rys. 8.4.

8.4. Procedury sekwencyjne z etapami czasowymi

W odniesieniu do procedur sekwencyjnych, w których realizację etapów elementarnych określają warunki czasowe, podczas syntezy równania schematowego obowiązują następujące zasady:

Zasada 1

Zgodnie z zasadami syntezy algorytmu sterowania metodą Grafpol dla procedur sekwencyjnych wyznacza się warunki zapisu, użycia i kasowania poszczególnych elementarnych komórek pamięci.

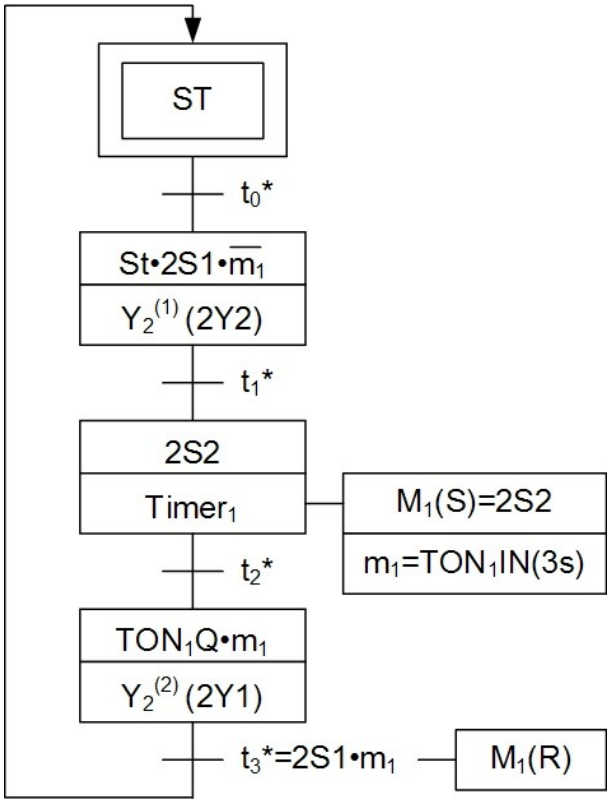
W krokach czasowych stosowany jest licznik czasu typu TON wg normy PN-EN 61131-3, którego blok oraz diagram funkcjonalny przedstawiono na rys. 6.1 i 6.3.

Zasada 2

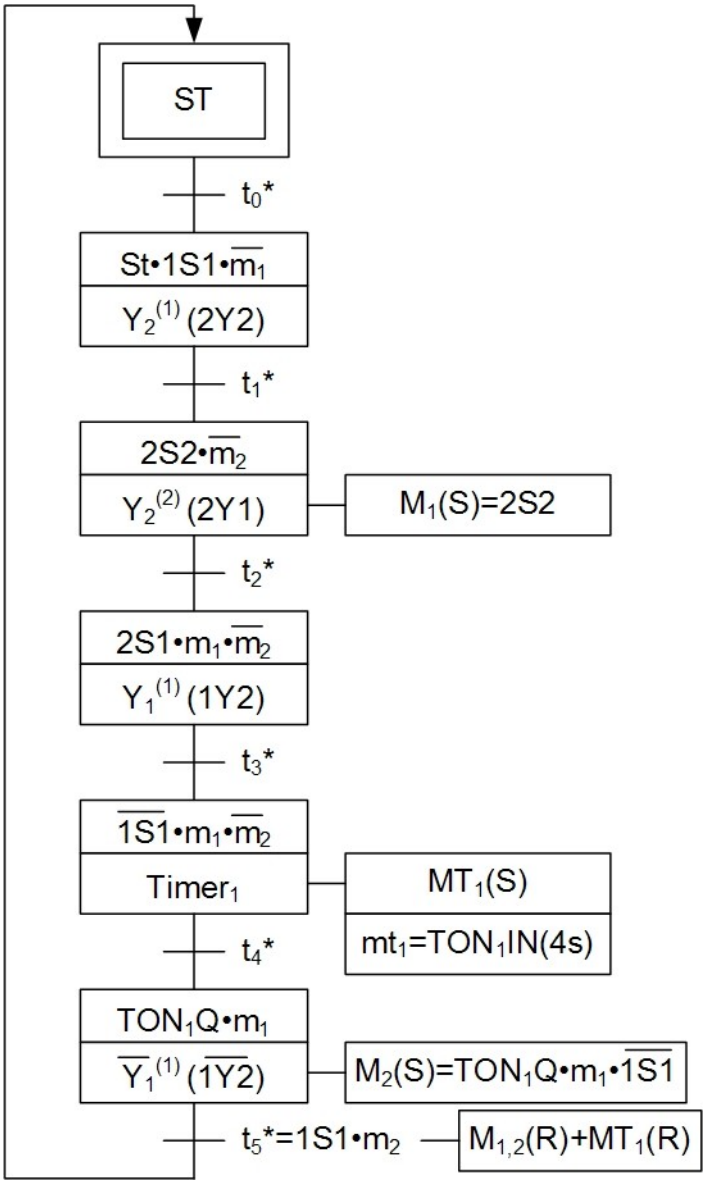
Warunki, które reprezentuje tranzycja t_j poprzedzająca krok, w którym następuje inicjacja k-tego licznika czasu, zapisują tzw. skojarzoną z licznikiem czasu komórkę pamięci inicjującą licznik czasu.

Jeśli z zasad realizacji pamięci dla procedur sekwencyjnych wynika, że w tranzycji t_j poprzedzającej krok, w którym następuje inicjacja licznika czasu:

- następuje zapis komórki pamięci, związany z pierwszym ruchem napędu z pozycji wyjściowej, wówczas sygnał tej komórki pamięci inicjuje licznik. Komórka ta jest jednocześnie komórką sterującą oraz komórką skojarzoną z licznikiem czasu (rys. 8.7).
- nie następuje zapis komórki pamięci, związany z pierwszym ruchem napędu z pozycji wyjściowej, wówczas należy ją zapisać przez warunki tranzycji t_j^*, a jej sygnał zastosować do inicjacji licznika czasu. Taką komórkę pamięci nazywamy skojarzoną z licznikiem czasu i oznaczamy MT_k, a jej sygnał mt_k (rys. 8.8).



Rys. 8.7. Ilustracja zasady 2, przypadek gdy sterująca komórka pamięci jest jednocześnie komórką pamięci skojarzoną z licznikiem czasu



Rys. 8.8. Ilustracja zasady 2, przypadek gdy licznik czasu jest inicjowany przez skojarzoną z nim komórkę pamięci MT_1

Zasada 3

W tranzycji występującej po kroku w którym nastąpiła inicjacja *k*-tego licznika czasu (rys. 8.7) stosujemy sygnał wyjściowy tego licznika (8.14).

$$t_{j+1} = TON_k Q \cdot \ldots \tag{8.14}$$

gdzie: TON_kQ – sygnał wyjściowy *k*-tego licznika czasu.
Tą zasadę zilustrowano przykładami na rys. 8.7 i 8.8.

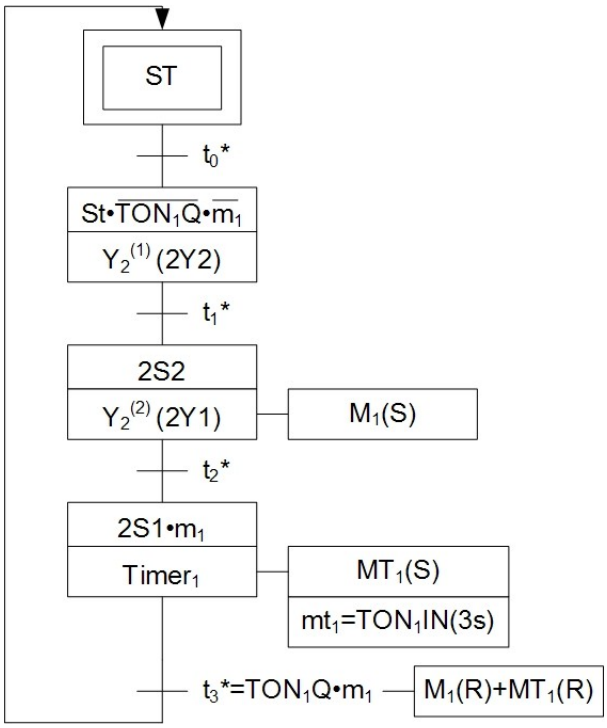
Zasada 4

Jeśli w ostatniej tranzycji t_n^* występuje sygnał *k*-tego licznika czasu, wówczas w tranzycji t^*_0 należy zastosować zanegowany sygnał wyjściowy *k*-tego licznika czasu (8.15).

$$\begin{aligned} t_n &= TON_k Q \cdot \ldots \\ t_0 &= ST \cdot \overline{TON_k Q} \cdot \ldots \end{aligned} \tag{8.15}$$

gdzie:
ST – sygnał wyjściowy elementu niezależnego od procesu (np. przycisk).

Tą zasadę zilustrowano przykładem na rys. 8.9.



Rys. 8.9. Algorytm ilustrujący zasadę 4 programowanie procedur sekwencyjnych z etapami czasowymi

Zasada 5

Kasowanie wszystkich zastosowanych w algorytmie sterowania komórek pamięci (sterujących i skojarzonych z licznikami czasu) następuje, gdy jest spełniony warunek ostatniej tranzycji t^*_n – rys. 8.7 i 8.8.

8.5. Procedury współbieżne

W przypadku kilku procedur sekwencyjnych rozpoczynających się w tym samym momencie i realizowanych jednocześnie mamy do czynienia z procedurami współbieżnymi. Syntezę równania schematowego współbieżnych algorytmów sterowania określają następujące zasady:

Zasada 1

Zgodnie z zasadami syntezy algorytmu sterowania metodą Grafpol dla poszczególnych procedur sekwencyjnych wyznacza się warunki zapisu i kasowania poszczególnych elementarnych komórek pamięci oraz określa się funkcje zmiennych wyjściowych.

Numerację etapów, tranzycji i komórek pamięci rozpoczyna się od pierwszej procedury sekwencyjnej i kontynuuje w sposób ciągły, aż do ostatniej procedury sekwencyjnej.

Zasada 2

Postać tranzycji t_0^*, określającej współbieżne rozpoczęcie procedur sekwencyjnych, opisuje zależność:

$$t_0^* = t_i^* \cdot t_j^* \cdot \ldots \cdot t_n^* \tag{8.16}$$

gdzie:
t^*_0 – zerowa tranzycja procedury współbieżnej,
t^*_i – zerowa tranzycja *i*-tej (pierwszej) procedury sekwencyjnej,
t^*_j – zerowa tranzycja *j*-tej (drugiej) procedury sekwencyjnej,
t^*_n – zerowa tranzycja *n*-tej (ostatniej) procedury sekwencyjnej.

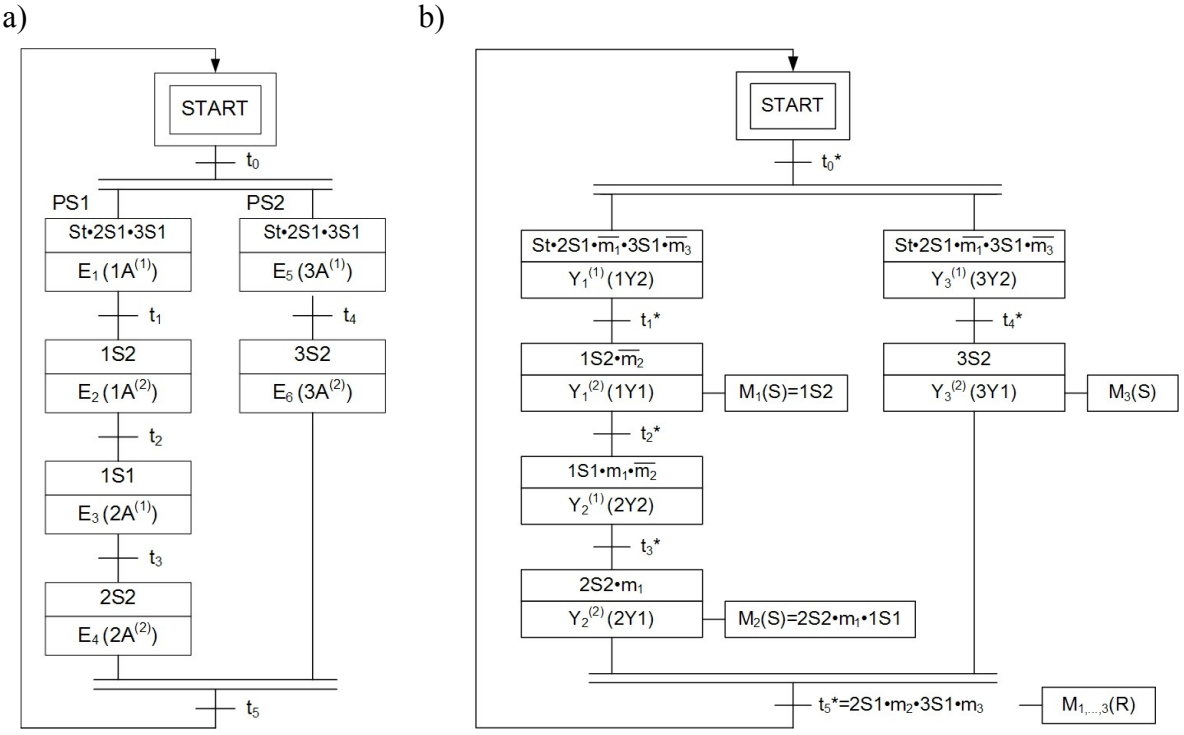
Zasada 3

Kasowanie wszystkich elementarnych komórek pamięci użytych w poszczególnych procedurach sekwencyjnych następuje zawsze w ostatnim stanie algorytmu sterowania i w procedurze współbieżnej określane jest przez następującą zależność:

$$M_{i,\ldots,n}(R)=t_i\cdot m_i\cdot t_j\cdot m_j\cdot\ldots\cdot t_n\cdot m_n=t^*_i\cdot t^*_j\cdot t^*_n \tag{8.17}$$

gdzie:
$M_{i,\ldots,n}(R)$ - zależność określająca kasowanie wszystkich użytych w procedurze współbieżnej pamięci,
t_i, t_j, t_n – ostatnia tranzycja *i*-tej (pierwszej), *j*-tej (drugiej), *n*-tej (ostatniej) procedury sekwencyjnej,
m_i – sygnał ostatniej komórki pamięci użytej w *i*-tej (pierwszej) procedurze sekwencyjnej,
m_j – sygnał ostatniej komórki pamięci użytej w *j*-tej (drugiej) procedurze sekwencyjnej,
m_n – *sygnał ostatniej komórki pamięci użytej w n-tej (ostatniej) procedurze sekwencyjnej,*
t^*_i, t^*_j, t^*_n – ostatnia tranzycja i-tej (pierwszej), j-tej (drugiej), n-tej (ostatniej) procedury sekwencyjnej z uwzględnionym sygnałem pamięci.

Powyższe zasady zilustrowano na rys. 8.10.



Rys. 8.10. Algorytm ilustrujący zasady obowiązujące przy syntezie równania schematowego procedur współbieżnych: a) algorytm procesu procedury współbieżnej; b) rozwiązanie procedury współbieżnej zapisane za pomocą sieci Grafpol GS

8.6. Procedury złożone

Ze względu na opracowane zasady w niniejszej pracy pod pojęciem procedury złożonej przyjęto procedurę składającą się z procedur sekwencyjnych, współbieżnych oraz zawierających kroki czasowe. W związku z tym do syntezy równania schematowego procedur złożonych stosowane są zasady opracowane dla wspomnianych typów procedur. Jedyną różnicą w przypadku procedur współbieżnych, będących procedurami składowymi złożonego algorytmu, jest miejsce kasowanie elementarnych komórek pamięci. Kasowanie wszystkich użytych w procedurze złożonej elementarnych komórek pamięci następuje zawsze w ostatnim stanie algorytmu sterowania oznaczanym tranzycją t_n*.

8.7. Schemat funkcjonalny a równanie schematowe

Postać syntezowanego równania schematowego zależy od schematu funkcjonalnego, a w szczególności od rodzaju zaworów sterujących. Co jednak w przypadku, gdy w wyniku awarii konieczna jest wymiana zaworu bistabilnego na, jedyny dostępny na magazynie utrzymania ruchu, zawór monostabilny? Co w przypadku gdy budowany jest kolejny egzemplarz urządzenia, który został jednak zmodyfikowany względem pierwowzoru, poprzez zastosowanie innych zaworów? Stosując Metodę Transformacji Sieci konieczne było syntezowanie równania schematowgo całkowicie od nowa. W przypadku nowych zasad syntezy równania schematowego metodą Grafpol wymagane są jedynie drobne poprawki. Przykładowe równanie schematowe (8.18) odnosi się do schematu funkcjonalnego przedstawionego na rys. 9.2.

$$F(Y,M)=\sum\begin{cases} ST\cdot 1S1\cdot\overline{m_1}=Y_2^{(1)}, \\ 2S2\cdot\overline{m_2}=Y_2^{(2)}, \\ 2S1\cdot m_1\cdot\overline{m_2}=Y_1^{(1)}(S), \\ 1S2\cdot m_1=Y_1^{(1)}(R), \\ \\ 2S2=M_1(S), \\ 1S2\cdot m_1\cdot 2S1=M_2(S), \\ 1S1\cdot m_2=M_1(R)+M_2(R) \end{cases} \tag{8.18}$$

Przyjmijmy, że nastąpiła zmiana polegająca na tym, iż zawór 1V jest zaworem bistabilnym, a zawór 2V zaworem monostabilnym. W takiej sytuacji modyfikujemy zmienne Y_2 poprzez usunięcie zmiennej $Y_2^{(2)}$ i zastąpienie jej warunkami zapisu i kasowania zmiennej $Y_2^{(1)}$. Zmienne Y_1 modyfikujemy poprzez usunięcie zapisu i kasowania, które zastępowane jest wprowadzeniem zmiennej $Y_1^{(2)}$ (8.19).

$$F(Y,M)=\sum\begin{cases} ST\cdot 1S1\cdot\overline{m_1}=Y_2^{(1)}(S), \\ 2S2\cdot\overline{m_2}=Y_2^{(1)}(R), \\ 2S1\cdot m_1\cdot\overline{m_2}=Y_1^{(1)}, \\ 1S2\cdot m_1=Y_1^{(2)}, \\ \\ 2S2=M_1(S), \\ 1S2\cdot m_1\cdot 2S1=M_2(S), \\ 1S1\cdot m_2=M_1(R)+M_2(R) \end{cases} \tag{8.19}$$

8.8. Implementacja równania schematowego w różnych realizacjach układu sterowania

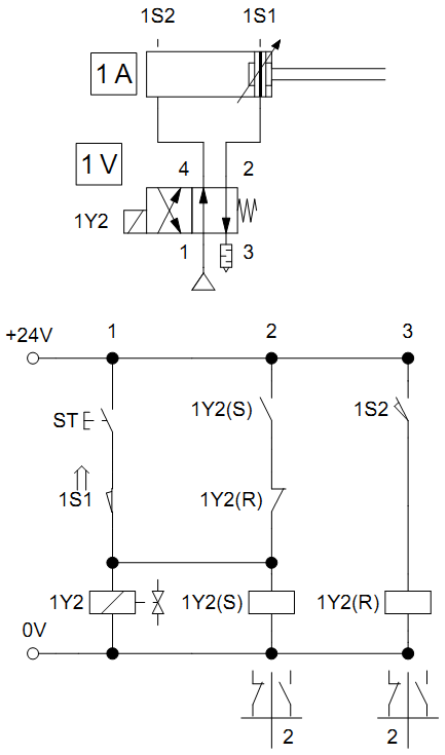
Równanie schematowe stanowi uniwersalny zapis opracowanego algorytmu sterowania. Równanie to może być zaimplementowane w układach sterowania realizowanych za pomocą:

- elementów pneumatycznych,
- układów stykowo-przekaźnikowych,
- mikroprocesorów (sterowniki PLC).

W odniesieniu do sterowników PLC równanie schematowe może być bezpośrednio zaimplementowane w sterowniku. Wynika to z faktu, iż sterowniki PLC umożliwiają zapis (*ang.* SET) i kasowanie (*ang.* RESET) wyjść. Jest to istotne w odniesieniu do monostabilnych elementów sterujących, tj. zaworów pneumatycznych czy przekaźników.

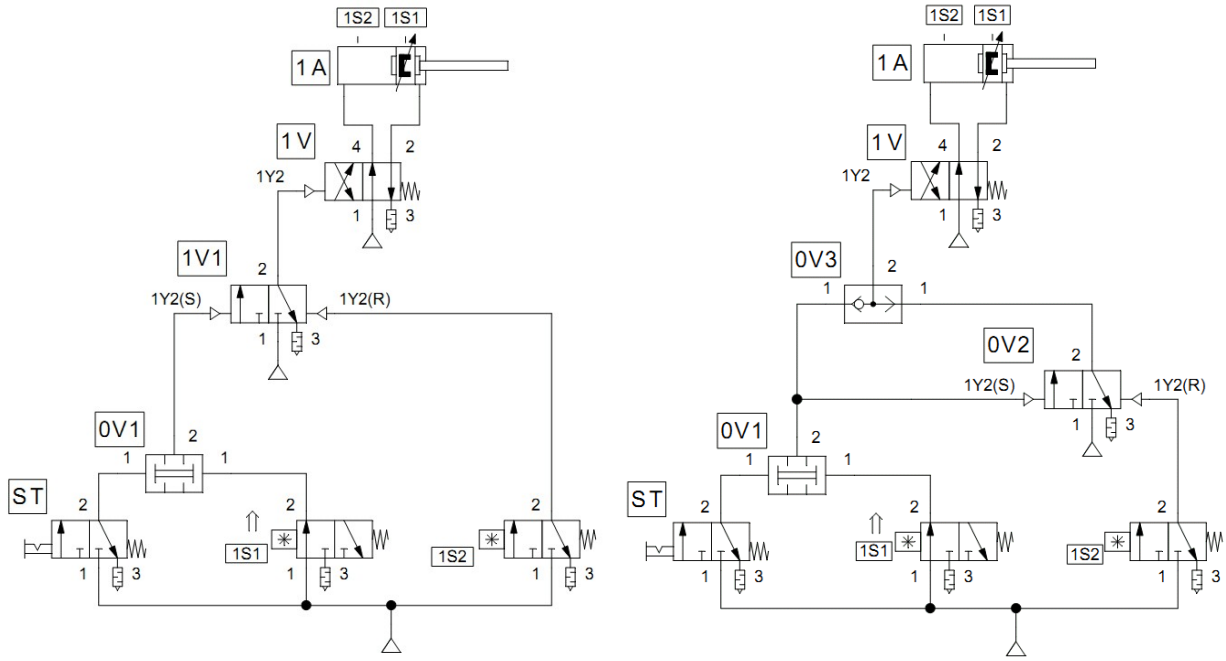
W przypadku realizacji układu sterowania za pomocą elementów stykowo-przekaźnikowych oraz elementów pneumatycznych konieczne jest stosowanie pętli podtrzymań. Sposoby implementacji przykładowego równania (8.20) w odniesieniu do realizacji układu sterowania za pomocą układów stykowo-przekaźnikowych i elementów pneumatycznch przedstawiono na rys. 8.11 i 8.12.

$$\begin{aligned} ST \cdot 1S1 &= 1Y2(S) \\ 1S2 &= 1Y2(R) \end{aligned} \tag{8.20}$$



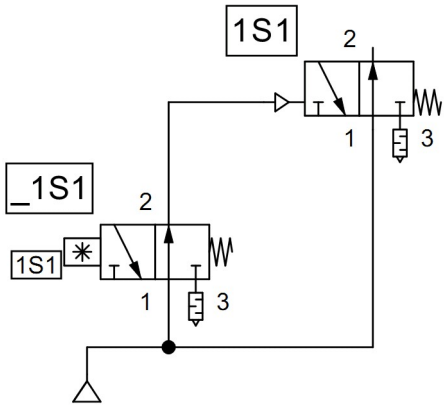
Rys. 8.11. Implementacja równania schematowego (8.20) w układzie stykowo-przekaźnikowym z zastosowaniem pętli podtrzymania

a) b)



Rys. 8.12. Implementacja równania schematowego (8.20) w układzie sterowania realizowanym za pomocą elementów pneumatycznych: a) z zastosowaniem bistabilnego zaworu 1V1 realizującego funkcję zapisu i kasowania; b) z zastosowaniem zaworu logicznego sumy 0V3 i bistabilnego zaworu 0V2 realizującego funkcję podtrzymania poprzez zapis i kasowanie

W przypadku stosowania sygnałów zanegowanych, np. $\overline{1S1}$, układ należy zrealizować w taki sposób by zawór z sygnałem zanegowanym sterował zaworem z sygnałem niezanegowanym (rys. 8.13). Na załączonych schematach pneumatycznych sygnały zanegowane oznaczane są symbolem "_" stosowanym przed oznaczeniem, np. _1S1.



Rys. 8.13. Kolejność instalacji zaworów w układzie sterowania realizowanym przez elementy pneumatyczne w przypadku użycia sygnału zanegowanego

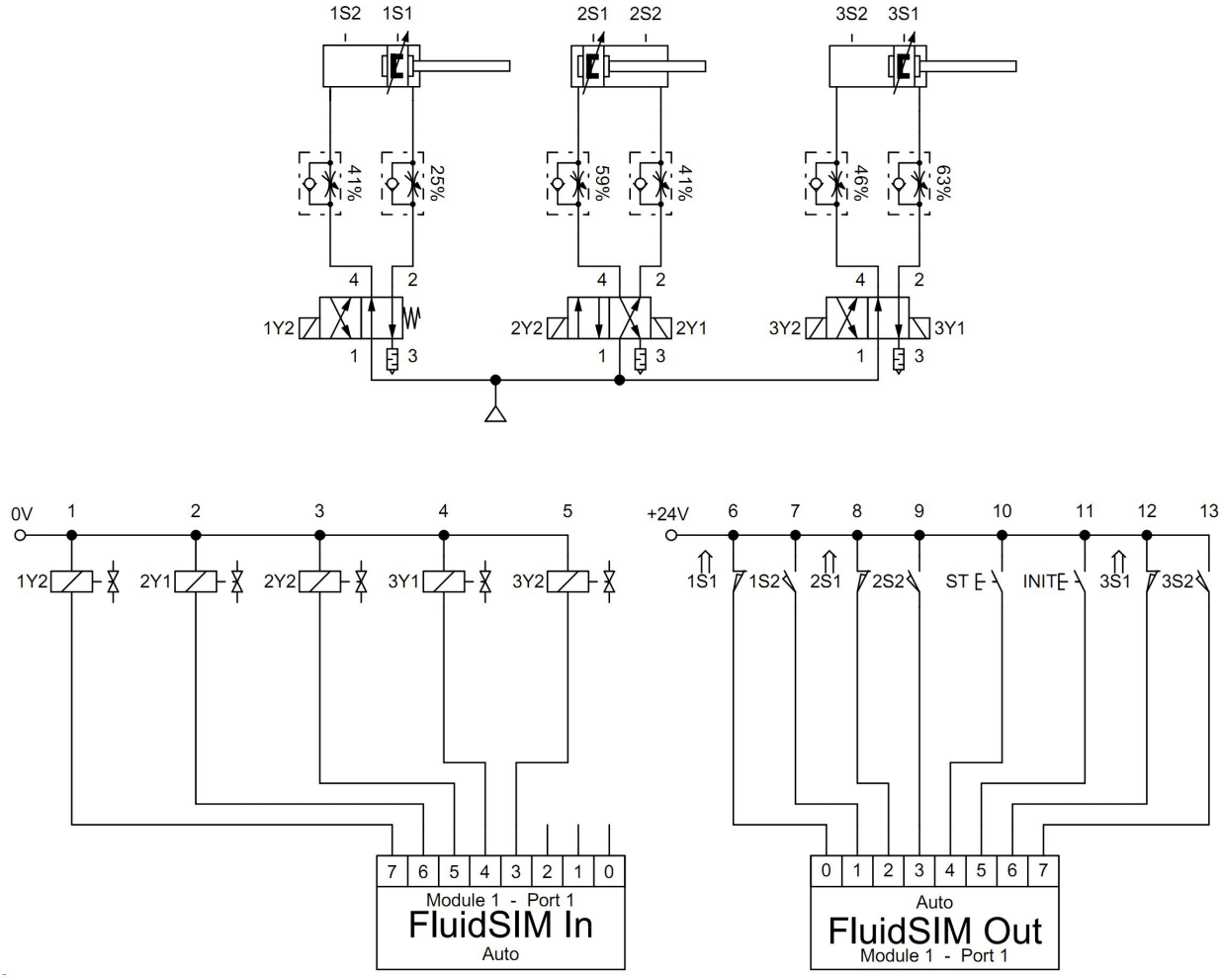
9. BADANIA SYMULACYJNE

Program badań ma na celu weryfikację postawionych tez. Przeprowadzone badania będą badaniami porównawczymi. W pierwszej kolejności syntezowane będą, w oparciu o opracowane zasady, równania schematowe przykładowych algorytmów pracy. Następnie otrzymane równania schematowe zostaną zaimplementowane w odmiennie zrealizowanych układach sterowania. Celem takiego działania będzie weryfikacja pracy napędów sterowanych przez opracowany algorytm sterowania w odniesieniu do oczekiwanego algorytmu pracy.

Implementacja równań schematowych nastąpi w układach sterowania zrealizowanych:

- jako układ stykowo-przekaźnikowy,
- poprzez elementy pneumatyczne,
- za pomocą sterownika PLC.

W dwóch pierwszych powyższych przypadkach zarówno proces, jak i układ sterowania zrealizowane zostaną jako model komputerowy w oprogramowaniu FluidSim. W przypadku użycia sterownika PLC oprogramowanie FluidSim zastosowane zostanie do stworzenia modelu procesu, który połączony zostanie z rzeczywistym sterownikiem za pomocą protokołu OPC. Wygląd tego modelu, wraz z modułami komunikacyjnymi, przedstawiono na rys. 9.1.

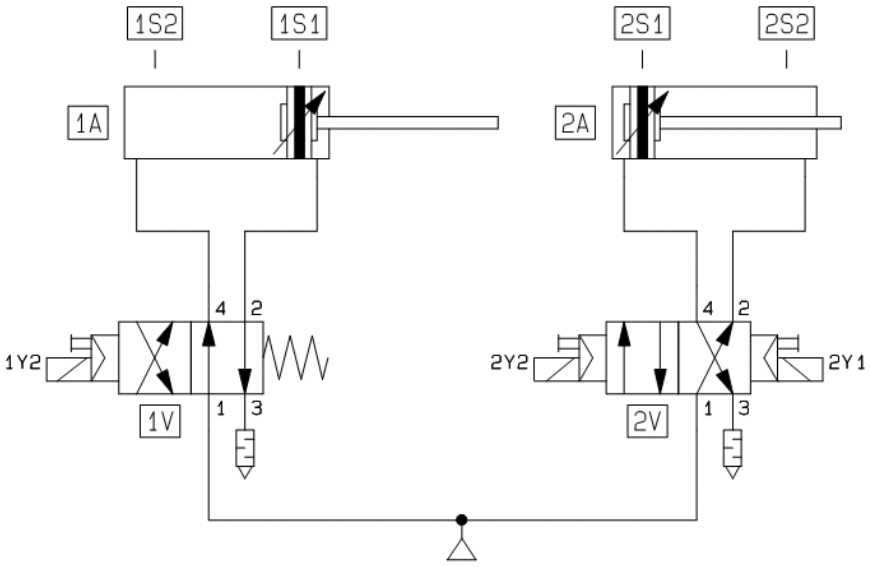


Rys. 9.1. Schemat funkcjonalny trzech napędów pneumatycznych wraz z przyporządkowaniem sygnałów wejściowo/wyjściowych do modułu komunikacyjnego protokołu OPC

9.1. Procedury sekwencyjne

Na rysunku 9.2 zaprezentowano schemat funkcjonalny dwóch napędów pneumatycznych 1A i 2A. Napędy te sterowane są odpowiednio zaworem 1V – monostabilnym i 2V – bistabilnym. Każdy z napędów pneumatycznych wyposażony jest w czujniki położeń krańcowych tłoka. Na poniżej zaprezentowanych przykładach, odnoszących się do procedur sekwencyjnych, schemat funkcjonalny uzupełniono o opis słowny opisujący jednoznacznie algorytm pracy poszczególnych napędów. Opis słowny składa się z nazwy symbolicznej etapu (np. E2) i komentarza do tego etapu (*wsuw tłoczyska siłownika 1A*). Ponadto zawiera informacje odnośnie do działania wykonywanego w danym etapie (np. $1A^{(1)}$ - pierwszy ruch napędu 1A), sposobu jego realizacji (np. podanie prądu na cewkę 1Y1) i oznaczenia czujników (np. 1S1) których sygnały informują o zakończeniu wykonania poszczególnych etapów.

Ze względu na brak możliwości zapisu i kasowania (setowania i resetowania) pneumatycznych zaworów monostabilnych oraz przekaźników konieczne było wprowadzenie zmian. W układzie sterowania realizowanym przez elementy pneumatyczne dodatkowo wprowadzono zawór 1V1 (rys. 9.4). W układzie pneumatycznym zawory których nazwy zaczynają się od znaku "_" reprezentują sygnały zanegowane (np. _M1 jest zanegowanym sygnałem M1). W układzie sterowania realizowanym za pomocą styków i przekaźników wprowadzono dodatkowo linie emulujące zapis (ang. *SET*), poprzez podtrzymanie przekaźnika w stanie prądowym, i jego kasowanie (ang. *RESET*) - np. linia 4 i 5 na rys. 9.5.



Rys. 9.2. Schemat funkcjonalny dwóch napędów pneumatycznych 1A i 2A

9.1.1. Przykład 1

Schemat funkcjonalny napędów zaprezentowano na rys. 9.2. Opis słowny algorytmu procesu prezentuje się następująco:

ETAP E1: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

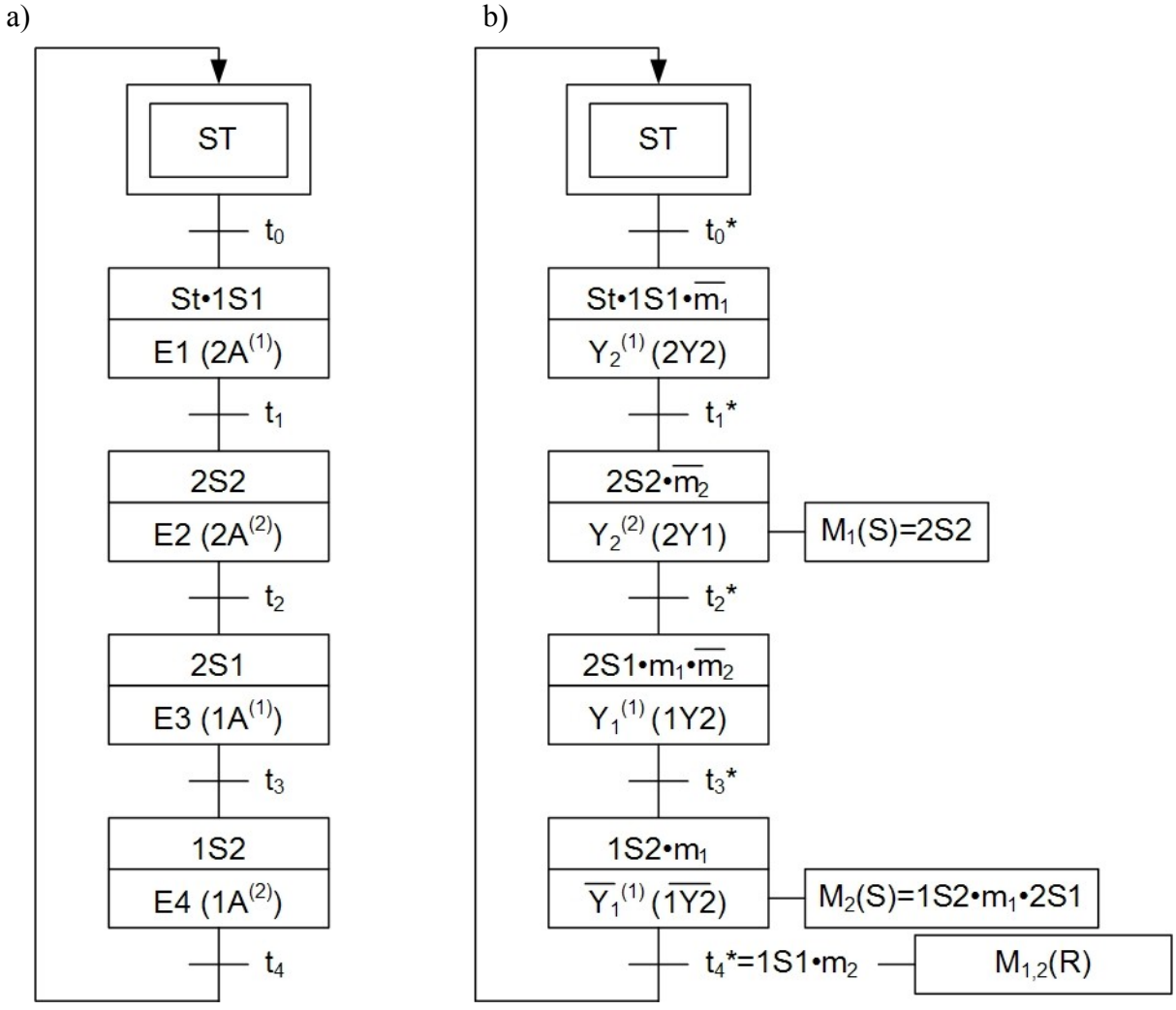
ETAP E2: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E3: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E4: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

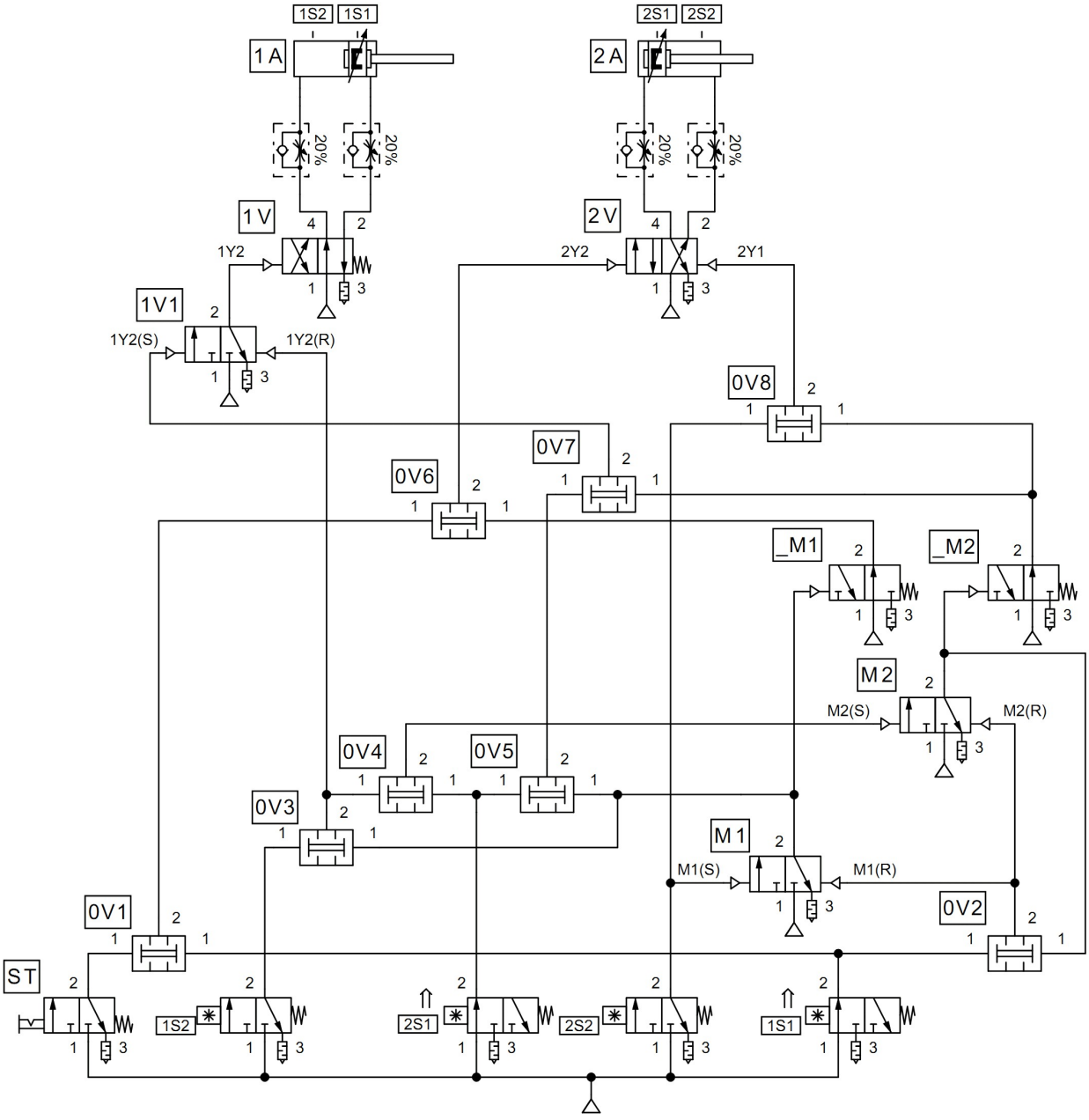
Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.3a). Na podstawie sieci Grafpol GP, stosując opracowane zasady, zaprojektowano algorytm sterowania wraz ze zrealizowaną pamięcią reprezentowany przez sieć Grafpol GS (rys. 9.3b). Na podstawie sieci Grafpol GS wyznaczono równanie schematowe projektowanego algorytmu sterowania (9.1). Implementację równania schematowego dla pneumatycznego, stykowo-przekaźnikowego i elektronicznego (PLC) układu sterowania przestawiono na rys. 9.4, 9.5 i 9.6.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

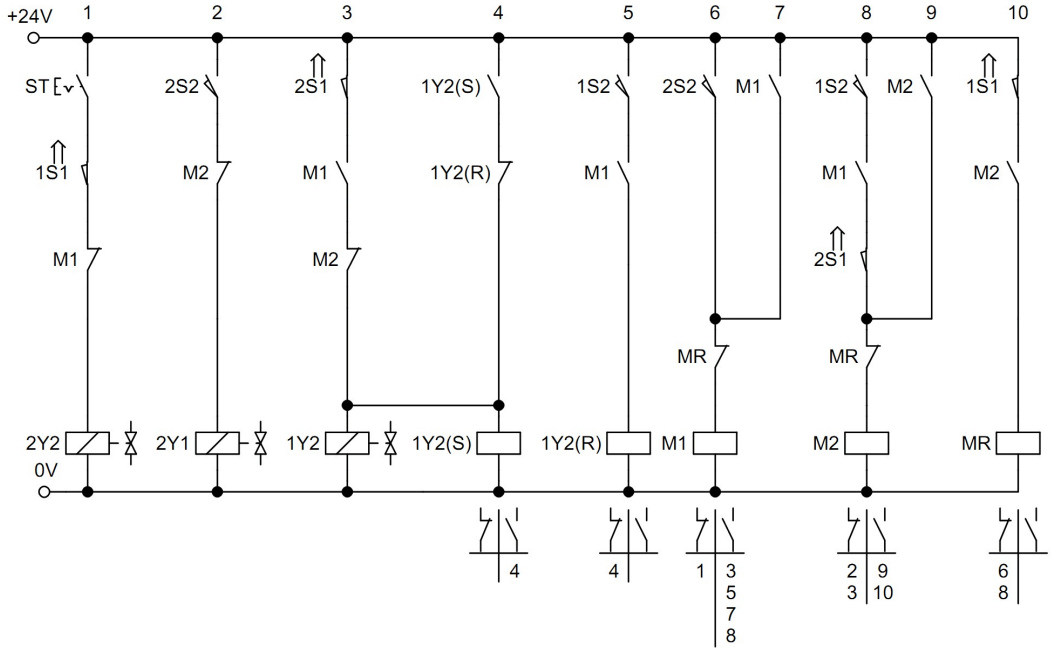


Rys. 9.3 Algorytmy przykładu 1 procedury sekwencyjnej : a) procesu; b) sterowania wraz ze zrealizowaną pamięcią

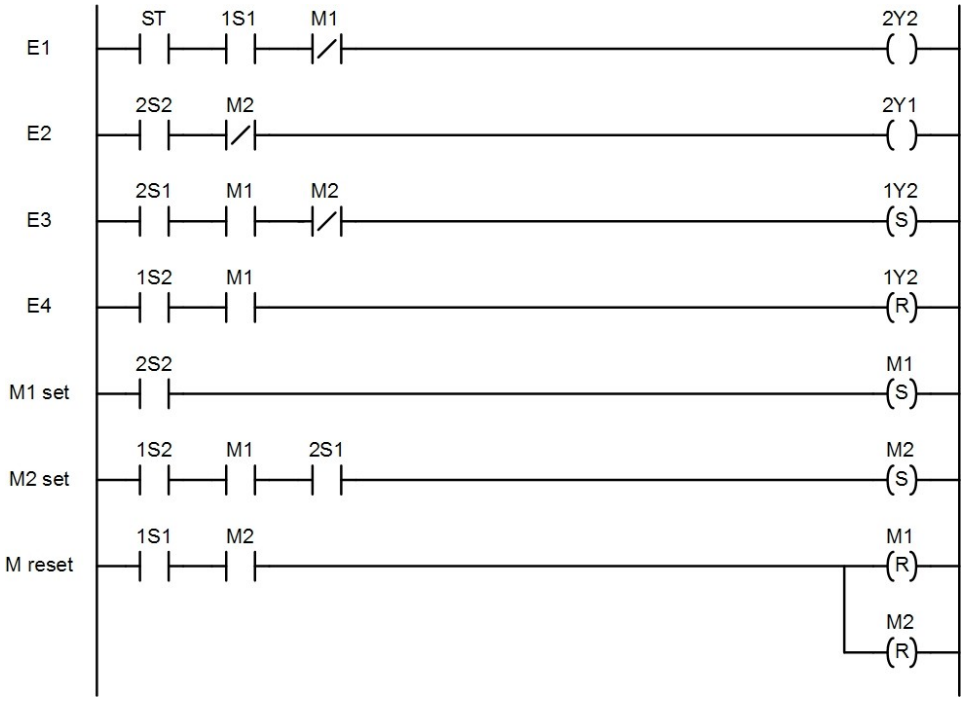
$$F(Y,M)=\sum\begin{cases} ST\cdot 1S1\cdot\overline{m_1}=Y_2^{(1)}, \\ 2S2\cdot\overline{m_2}=Y_2^{(2)}, \\ 2S1\cdot m_1\cdot\overline{m_2}=Y_1^{(1)}(S), \\ 1S2\cdot m_1=Y_1^{(1)}(R), \\ \\ 2S2=M_1(S), \\ 1S2\cdot m_1\cdot 2S1=M_2(S), \\ 1S1\cdot m_2=M_1(R)+M_2(R) \end{cases} \tag{9.1}$$



Rys. 9.4. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.1)



Rys. 9.5. Schemat stykowo-przekaźnikowego układu sterowania wyznaczonego na podstawie równania schematowego (9.1)



Rys. 9.6. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.1)

9.1.2. Przykład 2

Schemat funkcjonalny napędów zaprezentowano na rys. 9.2. Opis słowny algorytmu procesu prezentuje się następująco:

ETAP E1: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E2: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E3: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E4: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

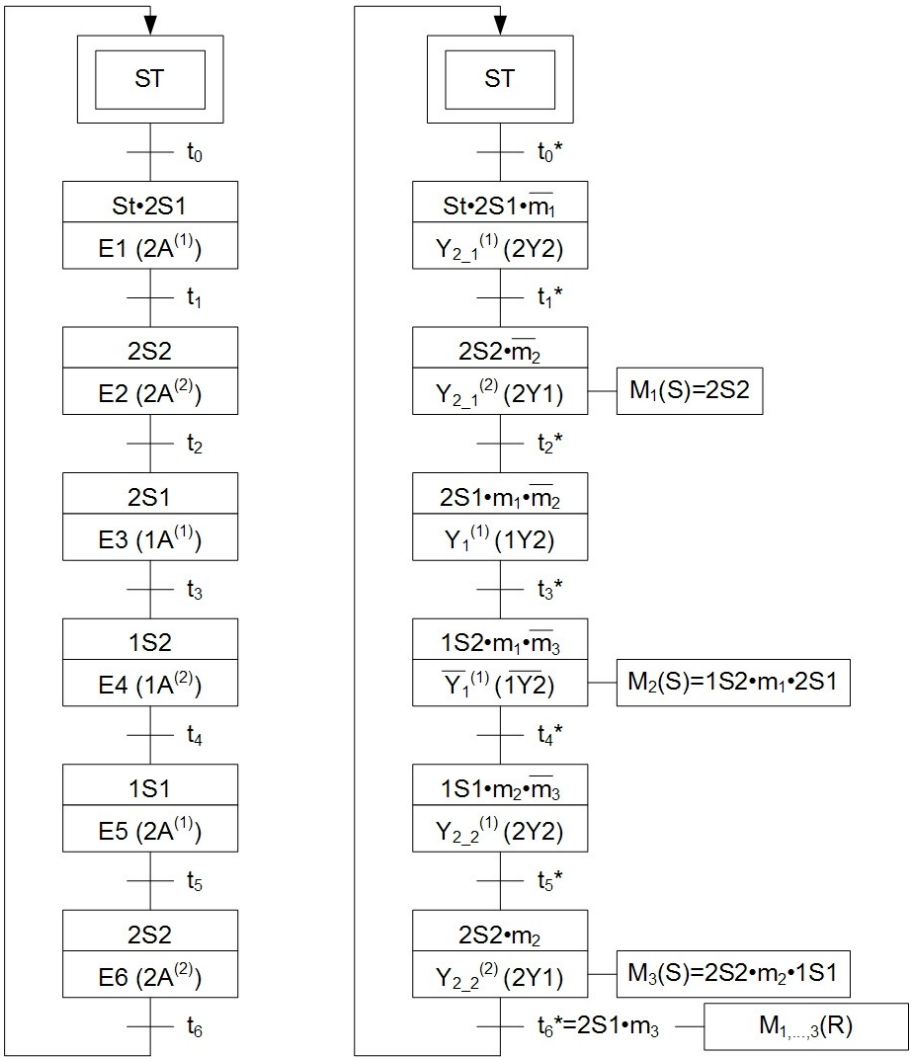
ETAP E5: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E6: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.7a). Na podstawie sieci Grafpol GP, stosując opracowane zasady, zaprojektowano algorytm sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.7b). W oparciu o sieć Grafpol GS dokonano syntezy równania schematowego (9.2). Implementację równania schematowego dla pneumatycznego, stykowo-przekaźnikowego i elektronicznego (PLC) układu sterowania przestawiono na rys. 9.8, 9.9 i 9.10.

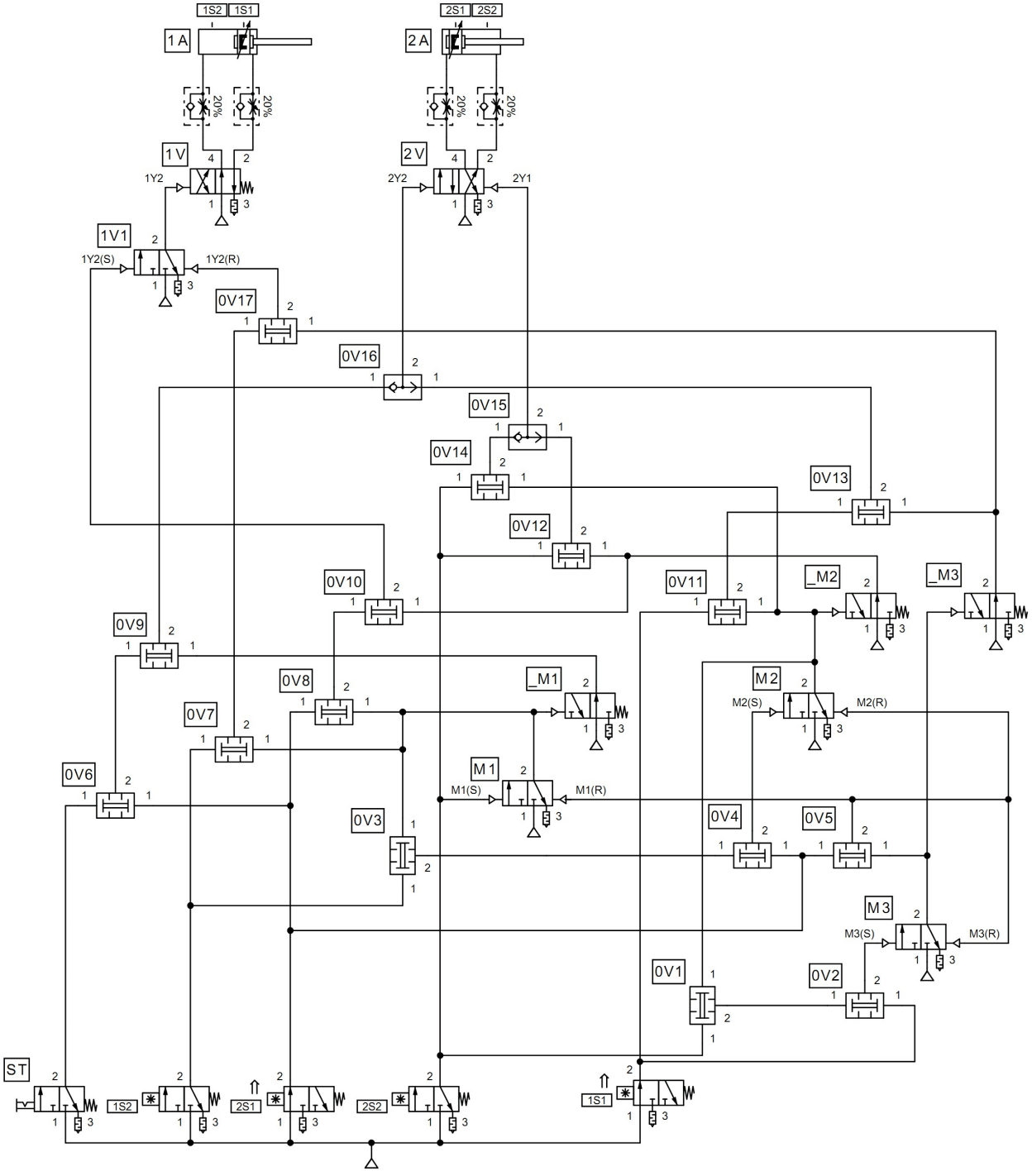
W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

a) b)

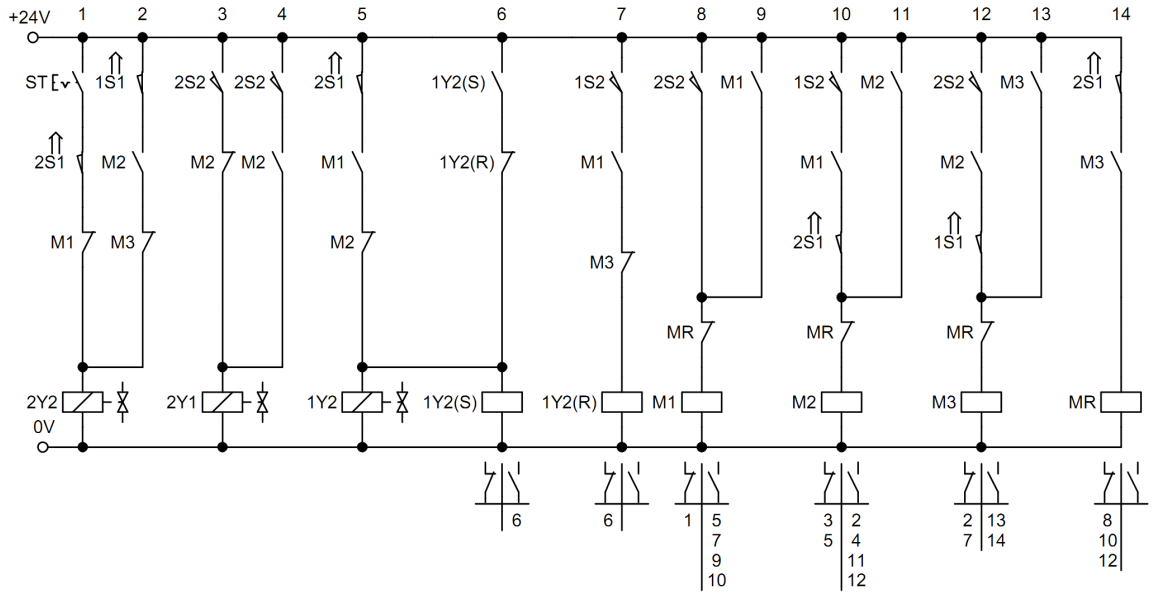


Rys. 9.7. Algorytmy przykładu 2 procedury sekwencyjnej: a) procesu; b) sterowania wraz ze zrealizowaną pamięcią

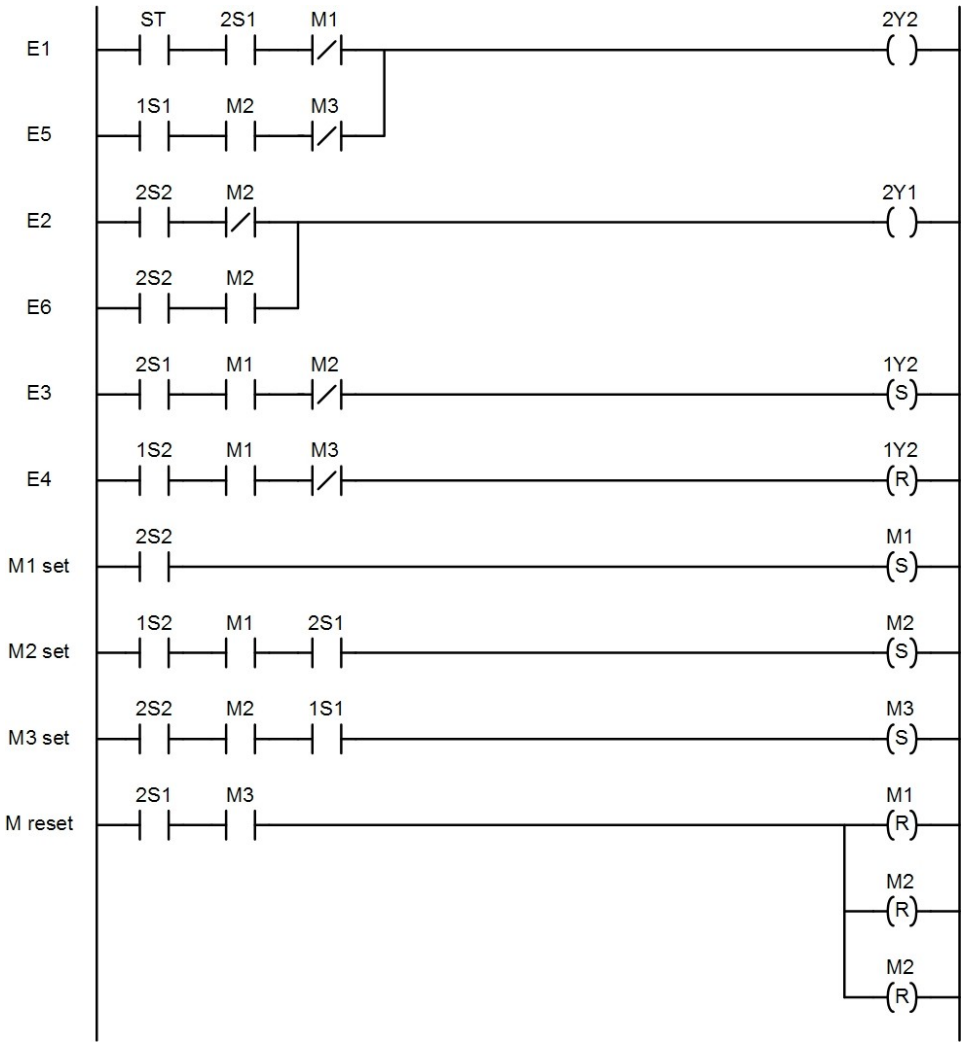
$$F(Y,M)=\sum\begin{cases}\left.\begin{aligned}&ST\cdot 2S1\cdot\overline{m_1}=Y^{(1)}_{2\_1}\\&1S1\cdot m_2\cdot\overline{m_3}=Y^{(1)}_{2\_2}\end{aligned}\right\}ST\cdot 2S1\cdot\overline{m_1}+1S1\cdot m_2\cdot\overline{m_3}=Y^{(1)}_{2},\\\left.\begin{aligned}&2S2\cdot\overline{m_2}=Y^{(2)}_{2\_1}\\&2S2\cdot m_2=Y^{(2)}_{2\_2}\end{aligned}\right\}2S2\cdot\overline{m_2}+2S2\cdot m_2=Y^{(2)}_{2},\\2S1\cdot m_1\cdot\overline{m_2}=Y^{(1)}_{1}(S),\\1S2\cdot m_1\cdot\overline{m_3}=Y^{(1)}_{1}(R),\\\\2S2=M_1(S),\\1S2\cdot m_1\cdot 2S1=M_2(S),\\2S2\cdot m_2\cdot 1S1=M_3(S),\\2S1\cdot m_3=M_1(R)+M_2(R)+M_3(R)\end{cases}\tag{9.2}$$



Rys. 9.8. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.2)



Rys. 9.9. Schemat stykowo-przekaźnikowego układu sterowania wyznaczonego na podstawie równania schematowego (9.2)



Rys. 9.10. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.2)

9.1.3. Przykład 3

Schemat funkcjonalny napędów zaprezentowano na rys. 9.2. Opis słowny algorytmu procesu prezentuje się następująco:

ETAP E1: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E2: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

ETAP E3: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

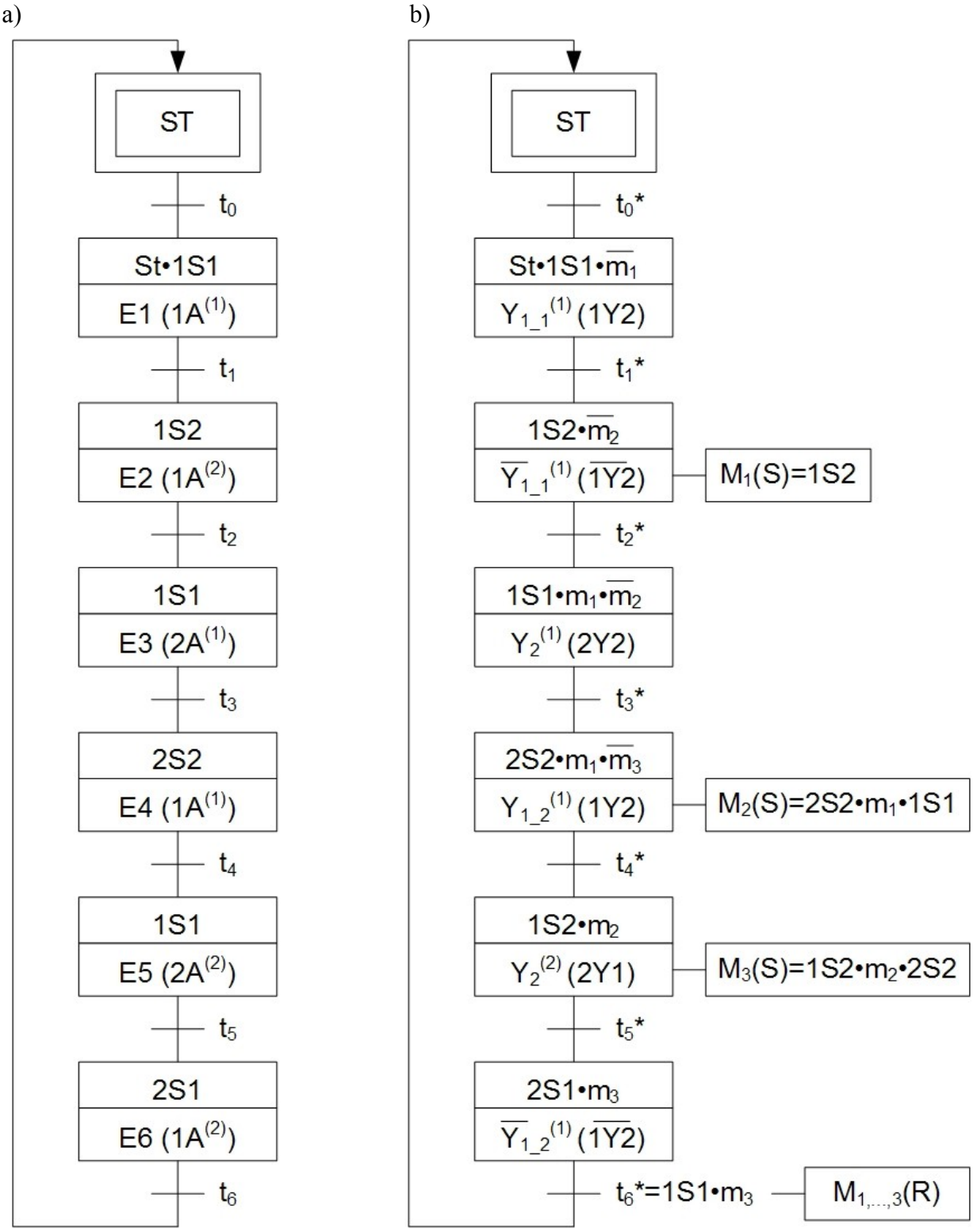
ETAP E4: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E5: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E6: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

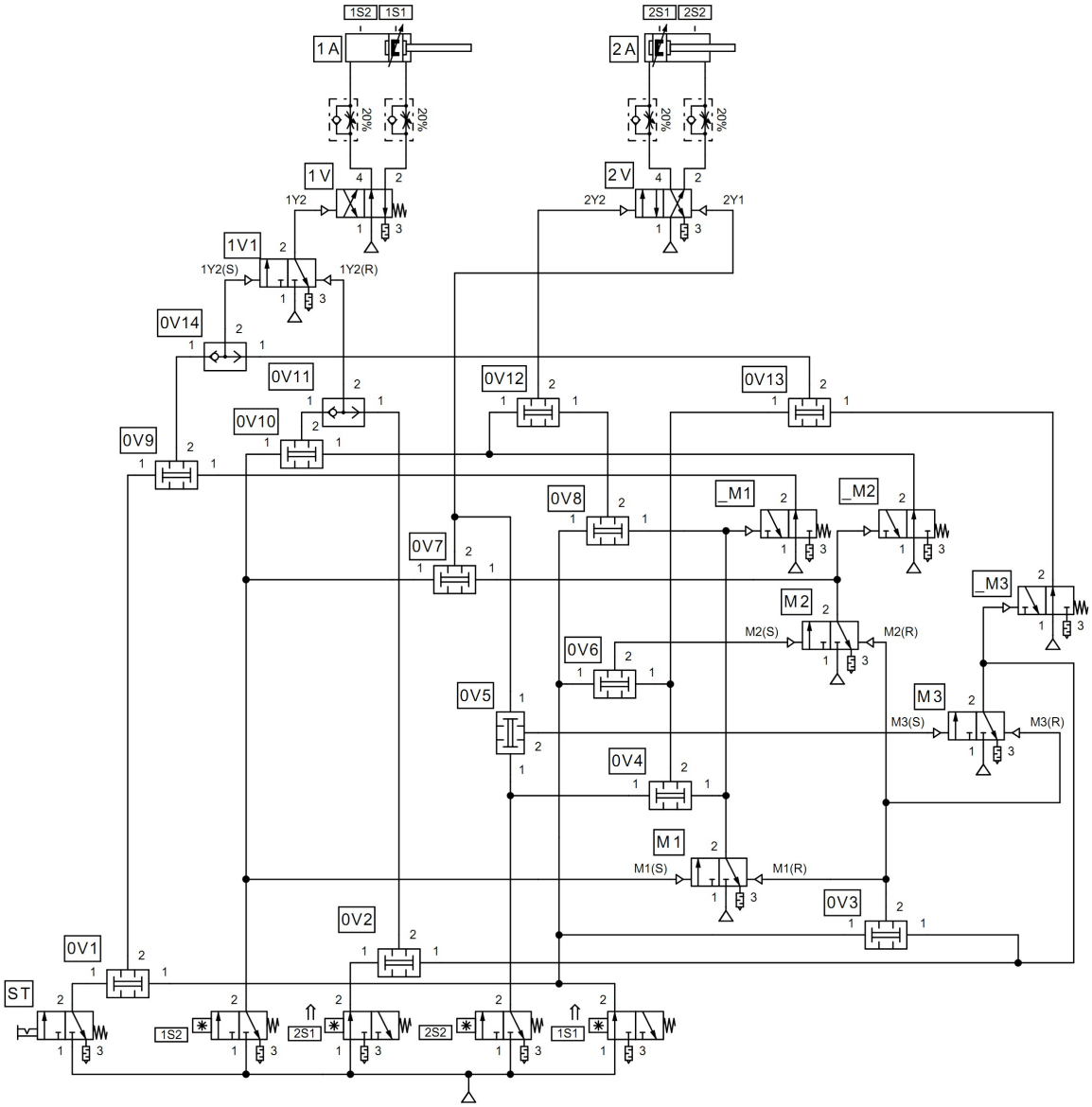
Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.11a). Na podstawie sieci Grafpol GP, stosując opracowane zasady, zaprojektowano algorytm sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.11b). W oparciu o sieć Grafpol GS dokonano syntezy równania schematowego (9.3). Implementację równania schematowego dla pneumatycznego, stykowo-przekaźnikowego i elektronicznego (PLC) układu sterowania przestawiono na rys.9.12, 9.13 i 9.14.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

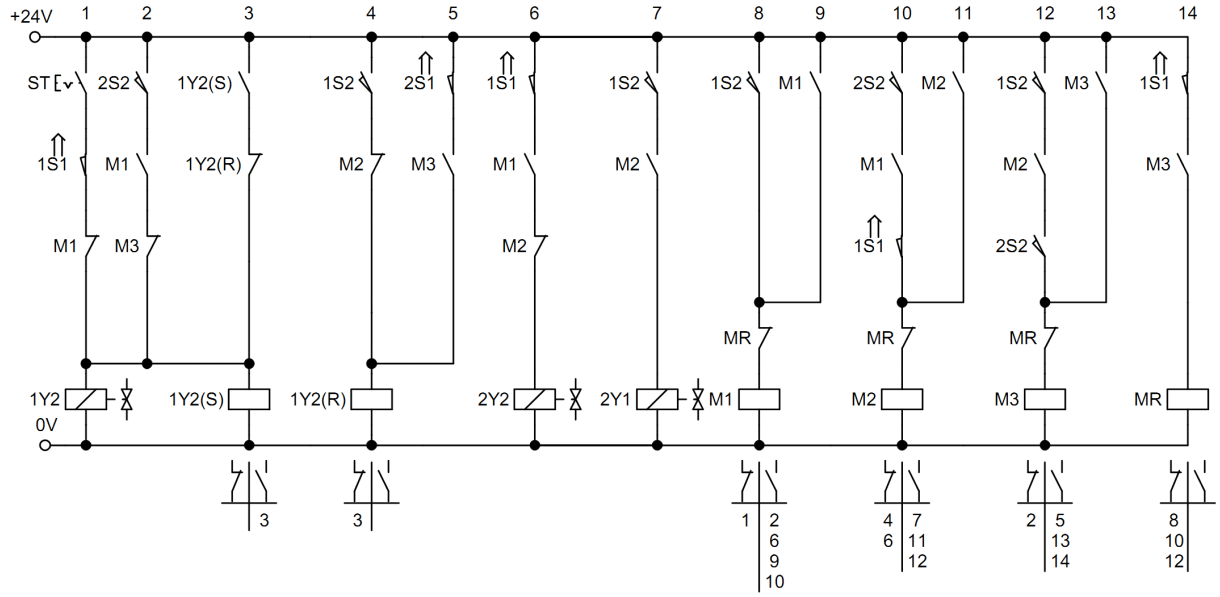


Rys. 9.11. Algorytmy przykładu 3 procedury sekwencyjnej: a) procesu; b) sterowania wraz ze zrealizowaną pamięcią

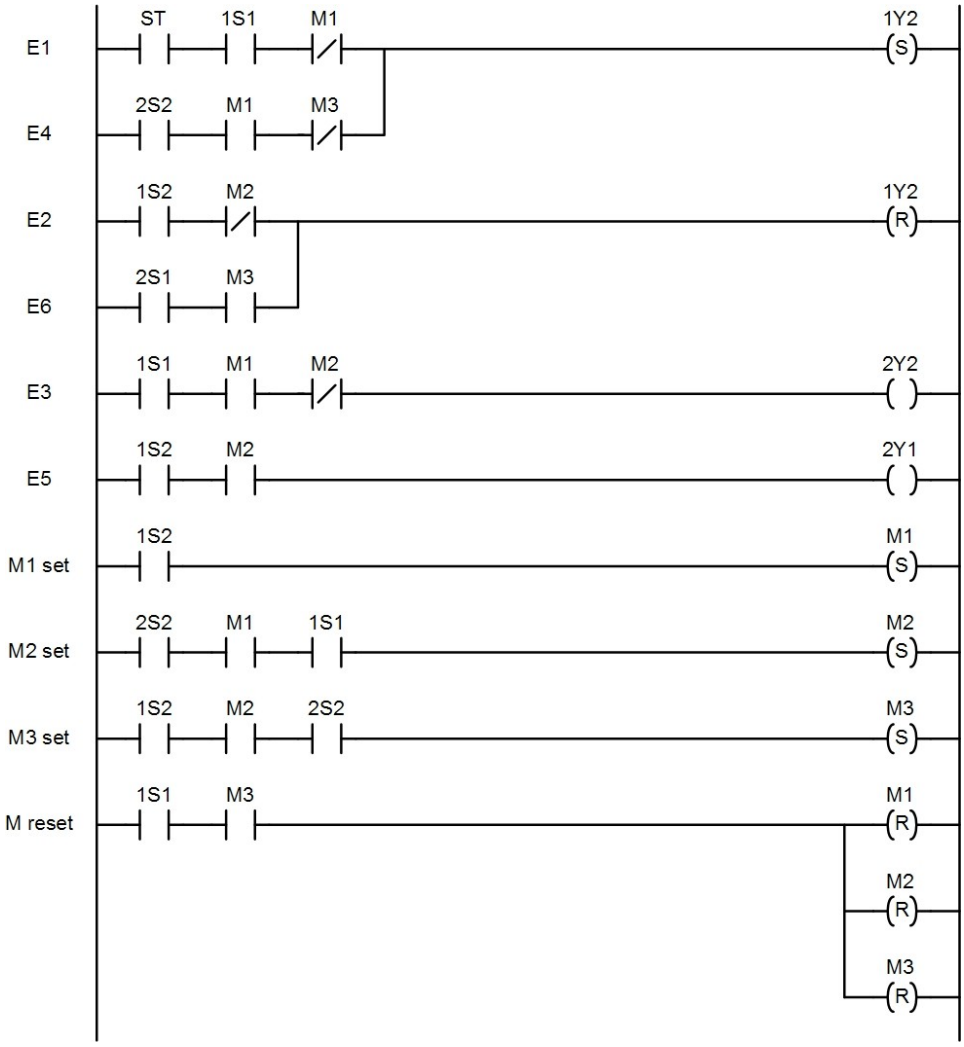
$$F(Y,M)=\sum\begin{cases}\left.\begin{aligned}ST\cdot 1S1\cdot\overline{m_1}&=Y^{(1)}_{1\_1}(S)\\2S2\cdot m_1\cdot\overline{m_3}&=Y^{(1)}_{1\_2}(S)\end{aligned}\right\}ST\cdot 1S1\cdot\overline{m_1}+2S2\cdot m_1\cdot\overline{m_3}=Y^{(1)}_{1}(S),\\\left.\begin{aligned}1S2\cdot\overline{m_2}&=Y^{(1)}_{1\_1}(R)\\2S1\cdot m_3&=Y^{(1)}_{1\_2}(R)\end{aligned}\right\}1S2\cdot\overline{m_2}+2S1\cdot m_3=Y^{(1)}_{1}(R),\\1S1\cdot m_1\cdot\overline{m_2}=Y^{(1)}_{2},\\1S2\cdot m_2=Y^{(2)}_{2},\\\\1S2=M_1(S),\\2S2\cdot m_1\cdot 1S1=M_2(S),\\1S2\cdot m_2\cdot 2S2=M_3(S),\\1S1\cdot m_3=M_1(R)+M_2(R)+M_3(R)\end{cases}\tag{9.3}$$



Rys. 9.12. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.3)



Rys. 9.13. Schemat stykowo-przekaźnikowego układu sterowania wyznaczonego na podstawie równania schematowego (9.3)



Rys. 9.14. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.3)

9.2. Procedury sekwencyjne z etapami czasowymi

W zaprezentowanych poniżej przykładach procedur sekwencyjnych z etapami czasowymi rozpatrywano dwa "typy" etapów czasowych. Pierwszy gdy po osiągnięciu pozycji skrajnej przez napęd następuje oczekiwanie przez zadany czas, drugi zaś gdy praca danego napędu trwa przez zadany czas.

9.2.1. Przykład 1

Schemat funkcjonalny napędów zaprezentowano na rys. 9.2. Opis słowny algorytmu procesu prezentuje się następująco:

ETAP E1: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E2: *odmierzanie czasu*
Realizacja: $Timer_1$(2s)
Sygnalizacja: TON_1Q=1

ETAP E3: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

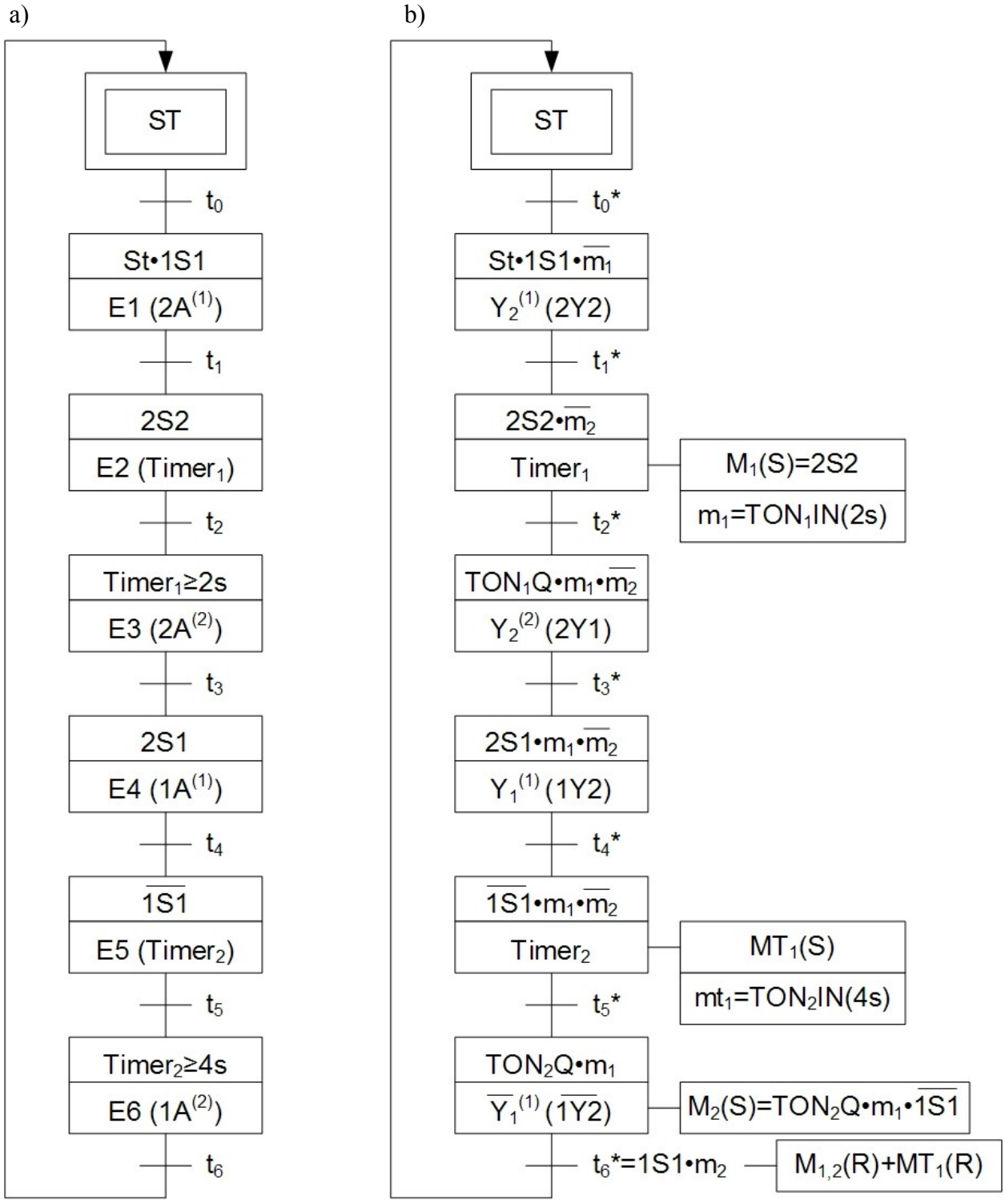
ETAP E4: *wsuw tłoczyska siłownika 1A przez zadany czas*
Realizacja: $1A^{(1)}$ (1Y2),
Sygnalizacja: $\overline{1S1}$=1

ETAP E5: *odmierzanie czasu*
Realizacja: $Timer_2$(4s)
Sygnalizacja: TON_2Q=1

ETAP E6: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

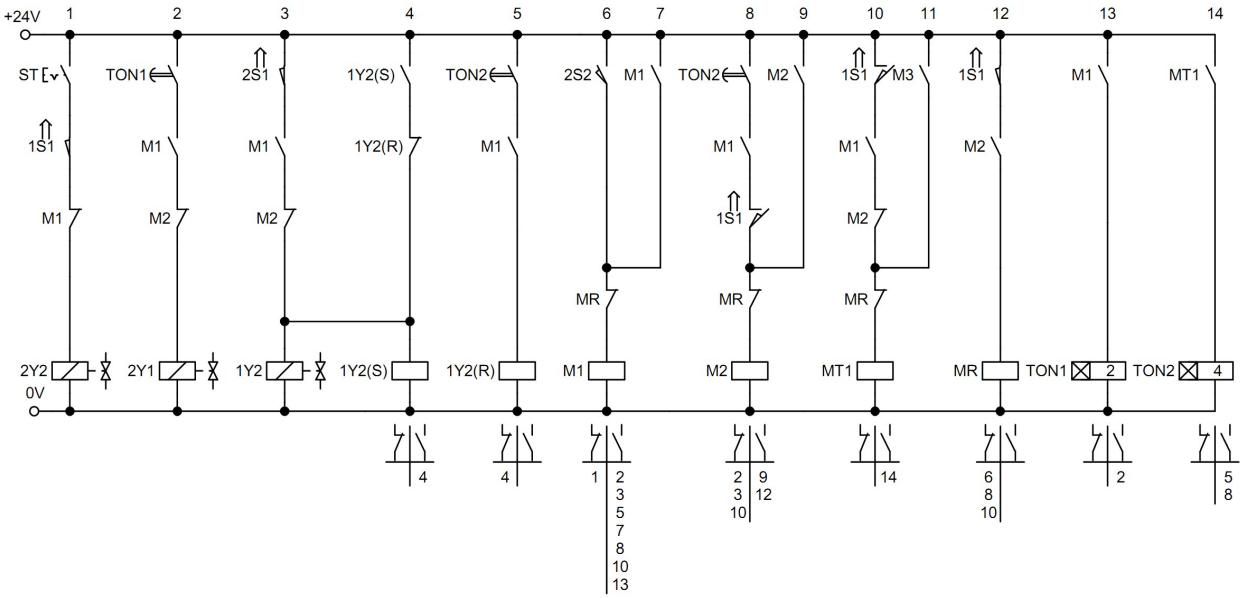
Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.15a). Na podstawie sieci Grafpol GP, stosując opracowane zasady, zaprojektowano algorytm sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.15b). W oparciu o sieć Grafpol GS dokonano syntezy równania schematowego (9.4). Implementację równania schematowego dla stykowo-przekaźnikowego, pneumatycznego i elektronicznego (PLC) układu sterowania przestawiono na rys. 9.16, 9.17 i 9.18.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

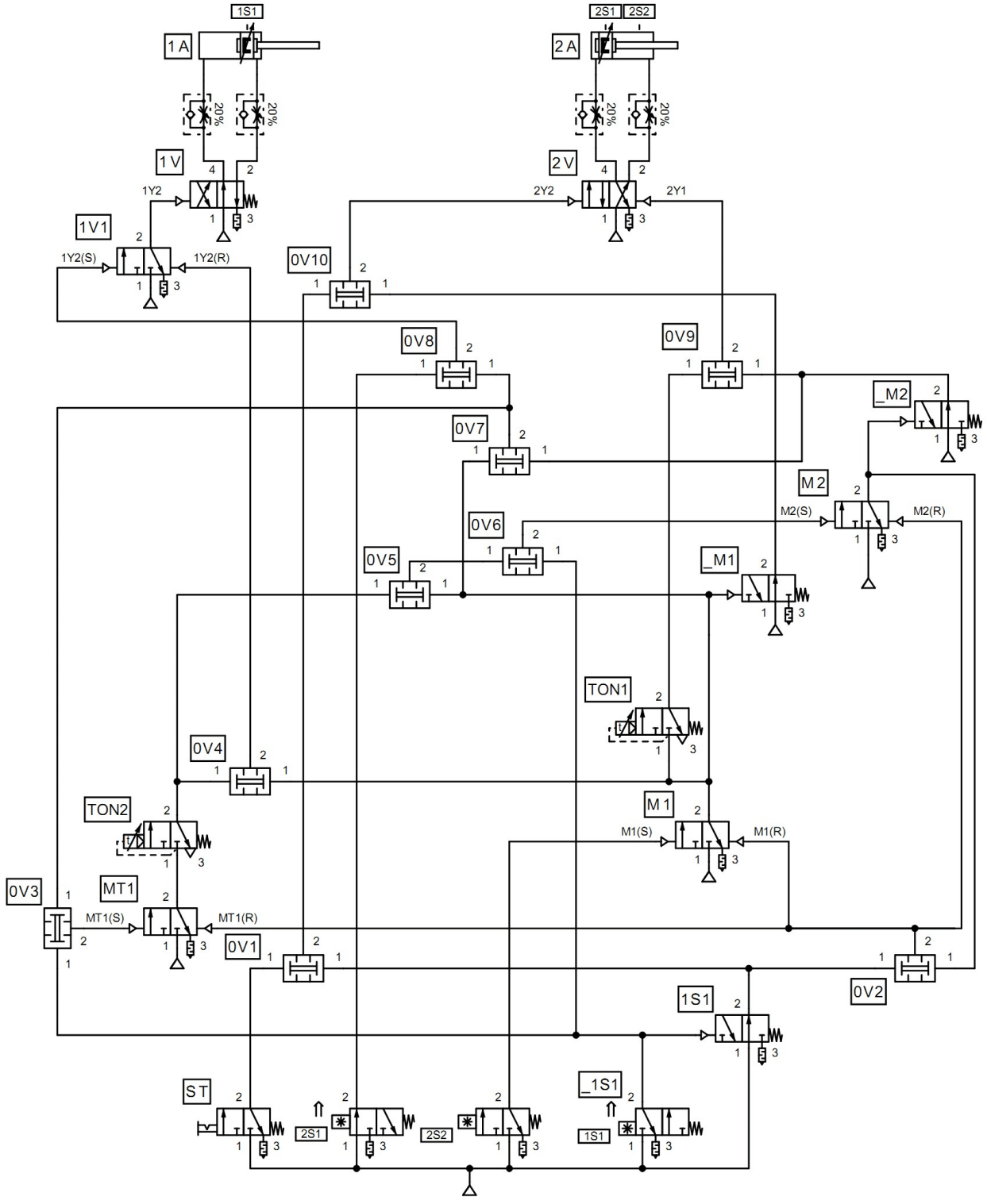


Rys. 9.15. Algorytmy przykładu 1 procedury sekwencyjnej z etapami czasowymi:
a) procesu; b) sterowania wraz ze zrealizowaną pamięcią

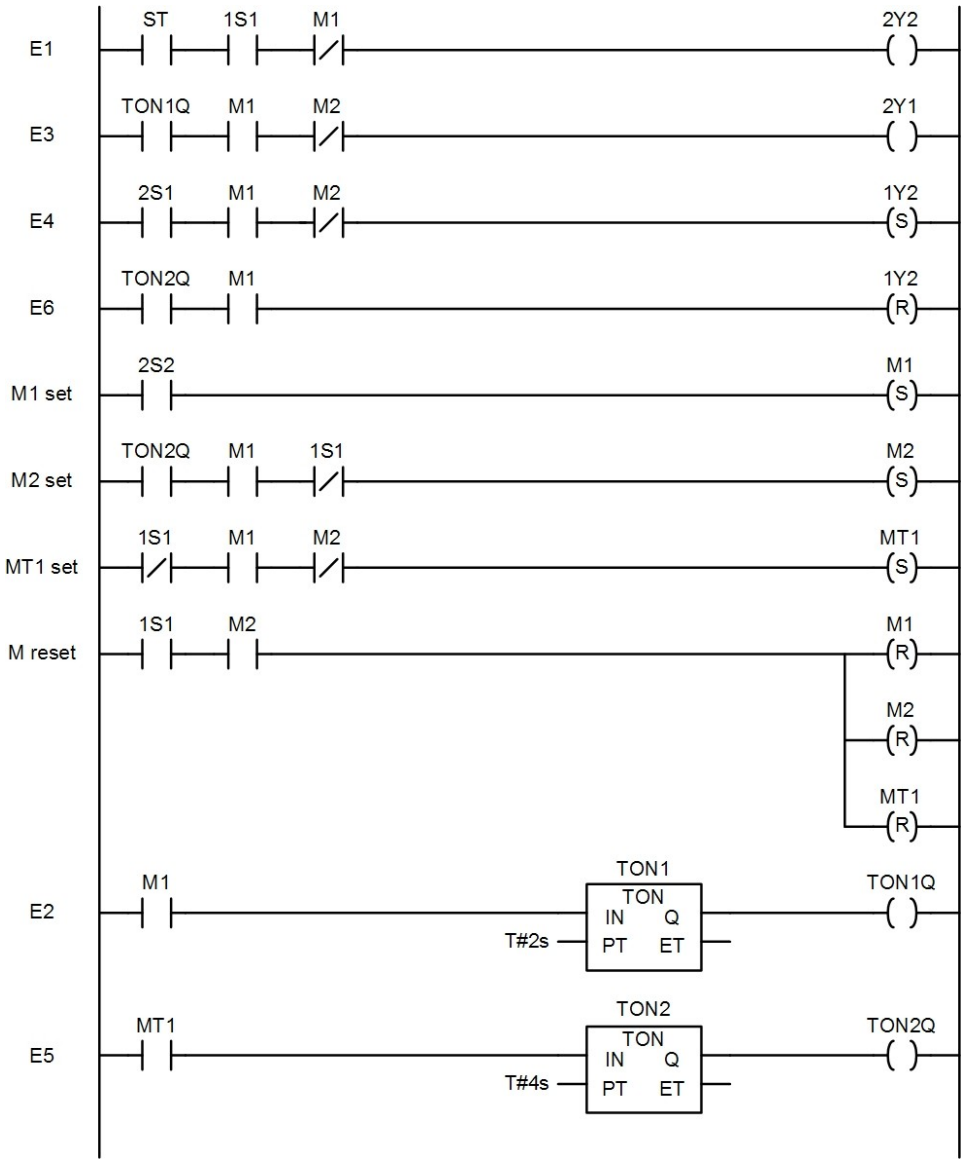
$$F(Y,M)=\sum\begin{cases} ST\cdot 1S1\cdot\overline{m_1}=Y_2^{(1)}, \\ TON_1Q\cdot m_1\cdot\overline{m_2}=Y_2^{(2)}, \\ 2S1\cdot m_1\cdot\overline{m_2}=Y_1^{(1)}(S), \\ TON_2Q\cdot m_1=Y_1^{(1)}(R), \\ \\ 2S2=M_1(S), \\ TON_2Q\cdot m_1\cdot\overline{1S1}=M_2(S), \\ \overline{1S1}\cdot m_1\cdot\overline{m_2}=MT_1(S), \\ 1S1\cdot m_2=M_1(R)+M_2(R)+MT_1(R) \\ \\ m_1=TON_1\,IN\,(2\mathrm{s}), \\ mt_1=TON_2\,IN\,(4\mathrm{s}), \end{cases} \tag{9.4}$$



Rys. 9.16. Schemat stykowo-przekaźnikowego układu sterowania wyznaczonego na podstawie równania schematowego (9.4)



Rys. 9.17. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.4)



Rys. 9.18. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.4)

9.2.2. Przykład 2

Schemat funkcjonalny napędów zaprezentowano na rys. 9.2. Opis słowny algorytmu procesu prezentuje się następująco:

ETAP E1: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E2: *odmierzanie czasu*
Realizacja: $Timer_1(5s)$
Sygnalizacja: $TON_1Q=1$

ETAP E3: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E4: *wsuw tłoczyska siłownika 1A przez zadany czas*
Realizacja: $1A^{(1)}$ (1Y2),
Sygnalizacja: $\overline{1S1}=1$

ETAP E5: *odmierzanie czasu*
Realizacja: $Timer_2(1s)$
Sygnalizacja: $TON_2Q=1$

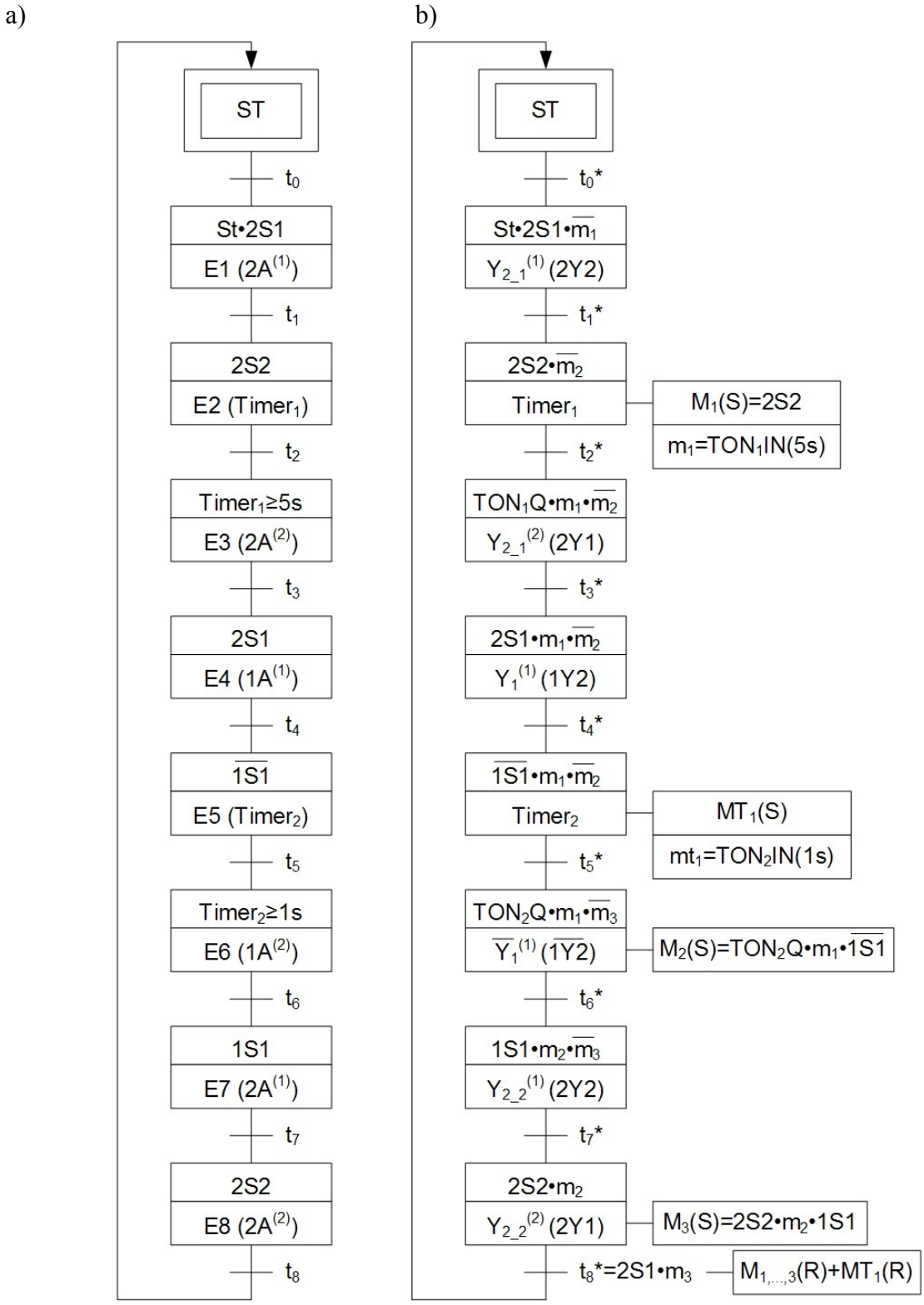
ETAP E6: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

ETAP E7: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E8: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

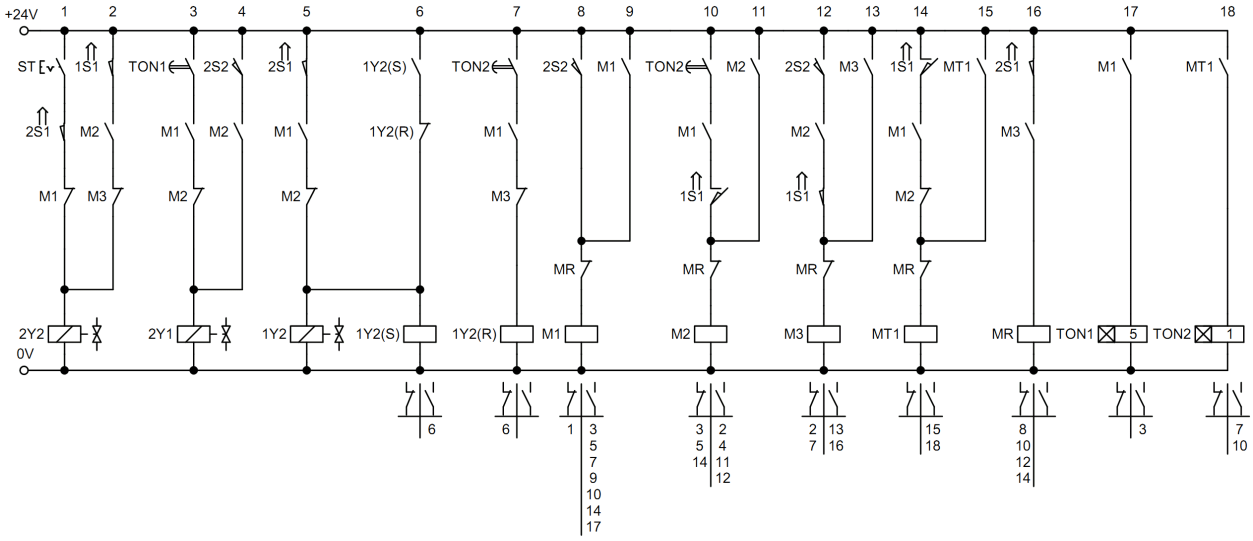
Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP(rys. 9.19a). Na podstawie sieci Grafpol GP, stosując opracowane zasady, zaprojektowano algorytm sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.19b). W oparciu o sieć Grafpol GS dokonano syntezy równania schematowego (9.5). Implementację równania schematowego dla stykowo-przekaźnikowego, pneumatycznego i elektronicznego (PLC) układu sterowania przestawiono na rys. 9.20, 9.21 i 9.22.

W wyniku przeprowadzonych prób stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

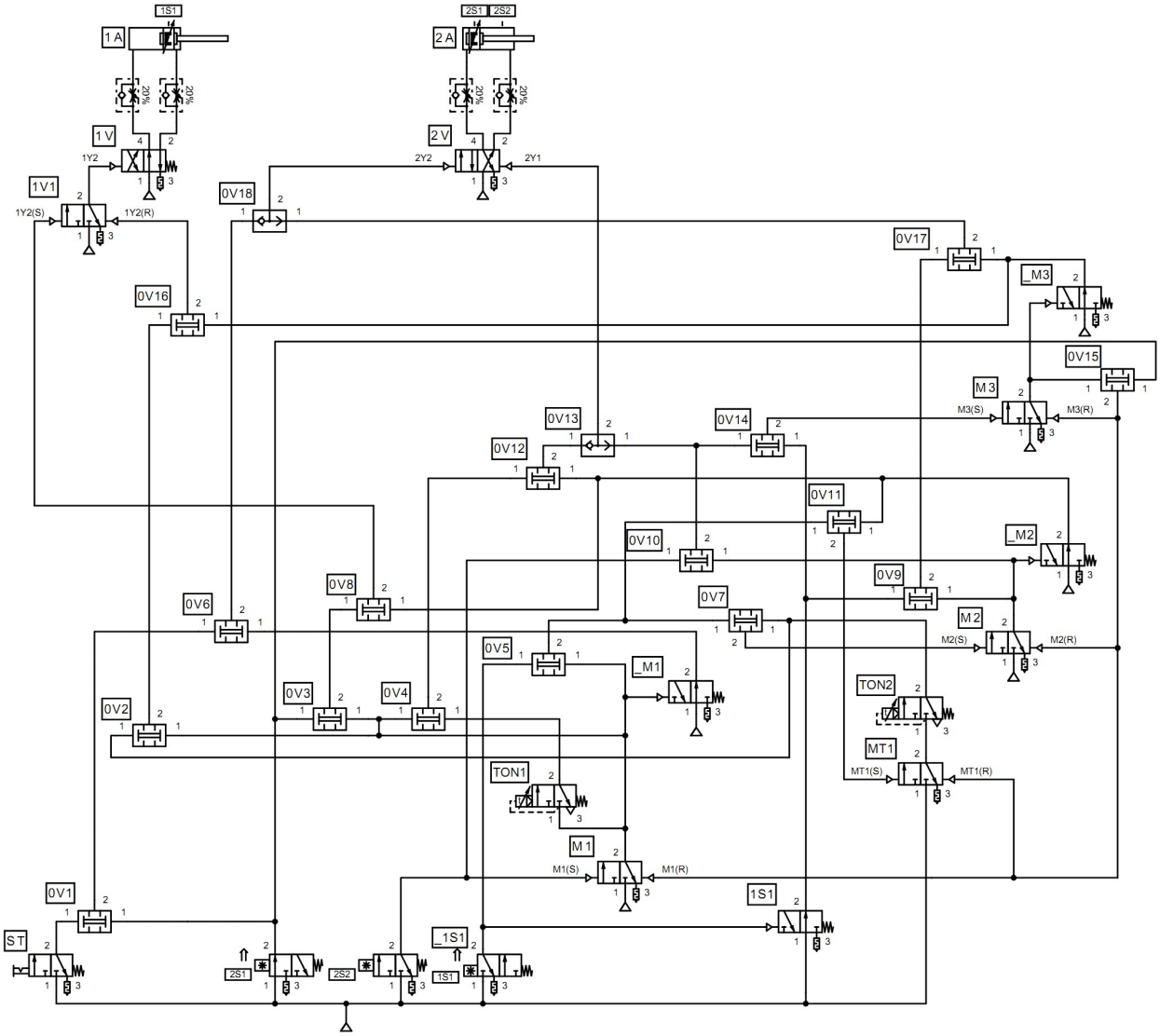


Rys. 9.19. Algorytmy przykładu 2 procedury sekwencyjnej z etapami czasowymi: a) procesu; b) sterowania wraz ze zrealizowaną pamięcią

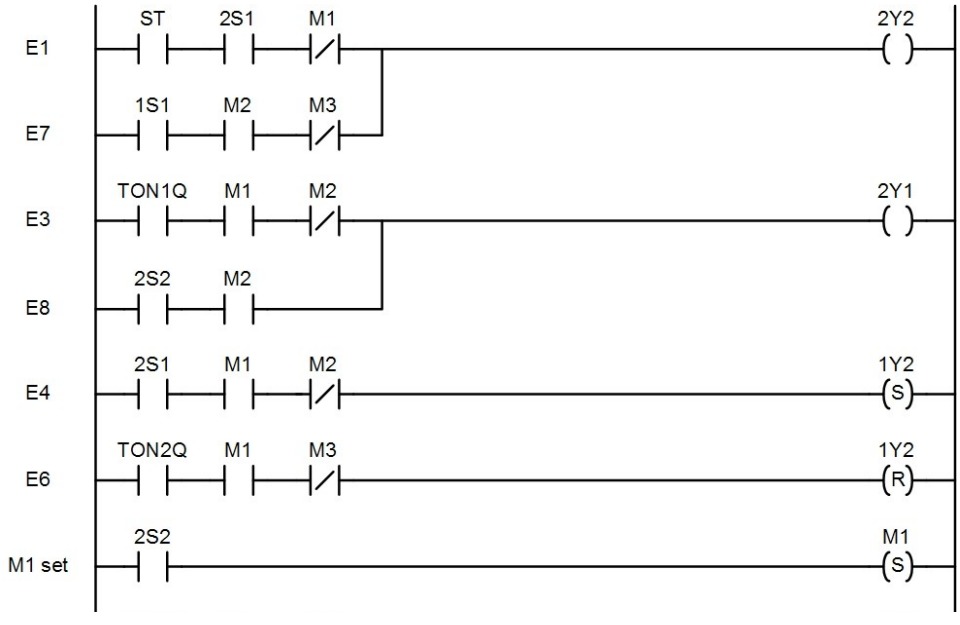
$$
F(Y,M)=\sum
\begin{array}{l}
\left.\begin{array}{l} ST\cdot 2S1\cdot\overline{m_1}=Y_{2\_1}^{(1)} \\ 1S1\cdot m_2\cdot\overline{m_3}=Y_{2\_2}^{(1)} \end{array}\right\} ST\cdot 2S1\cdot\overline{m_1}+1S1\cdot m_2\cdot\overline{m_3}=Y_2^{(1)}, \\
\left.\begin{array}{l} TON_1Q\cdot m_1\cdot\overline{m_2}=Y_{2\_1}^{(2)} \\ 2S2\cdot m_2=Y_{2\_2}^{(2)} \end{array}\right\} TON_1Q\cdot m_1\cdot\overline{m_2}+2S2\cdot m_2=Y_2^{(2)}, \\
2S1\cdot m_1\cdot\overline{m_2}=Y_1^{(1)}(S), \\
TON_2Q\cdot m_1\cdot\overline{m_3}=Y_1^{(1)}(R), \\
2S2=M_1(S), \\
TON_2Q\cdot m_1\cdot\overline{1S1}=M_2(S), \\
2S2\cdot m_2\cdot 1S1=M_3(S), \\
\overline{1S1}\cdot m_1\cdot\overline{m_2}=MT_1(S), \\
2S1\cdot m_3=M_1(R)+M_2(R)+M_3(R)+MT_1(R) \\
\\
m_1=TON_1\,IN(5s), \\
mt_1=TON_2\,IN(1s),
\end{array}
\tag{9.5}
$$

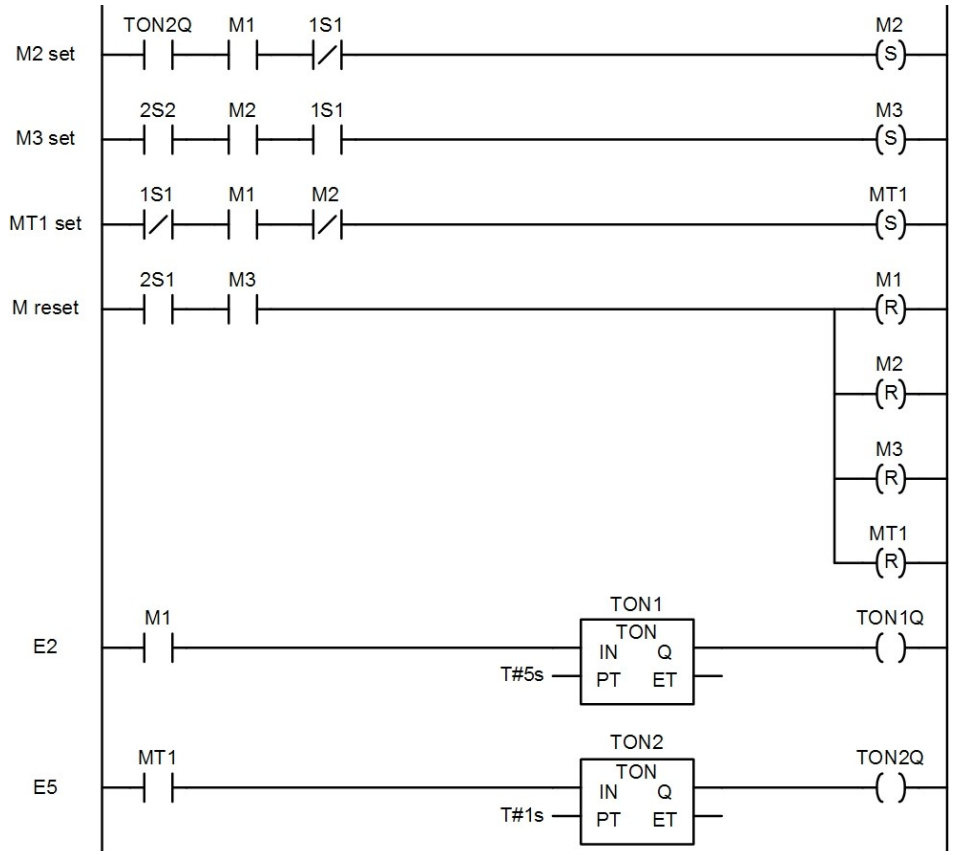


Rys. 9.20. Schemat stykowo-przekaźnikowego układu sterowania wyznaczonego na podstawie równania schematowego (9.5)



Rys. 9.21. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.5)





Rys. 9.22. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.5)

9.2.3. Przykład 3

Schemat funkcjonalny napędów zaprezentowano na rys. 9.2. Opis słowny algorytmu procesu prezentuje się następująco:

ETAP E1: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E2: *odmierzanie czasu*
Realizacja: $Timer_1$(3s)
Sygnalizacja: TON_1Q=1

ETAP E3: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

ETAP E4: *wysuw tłoczyska siłownika 2A przez zadany czas*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: $\overline{2S1}$=1

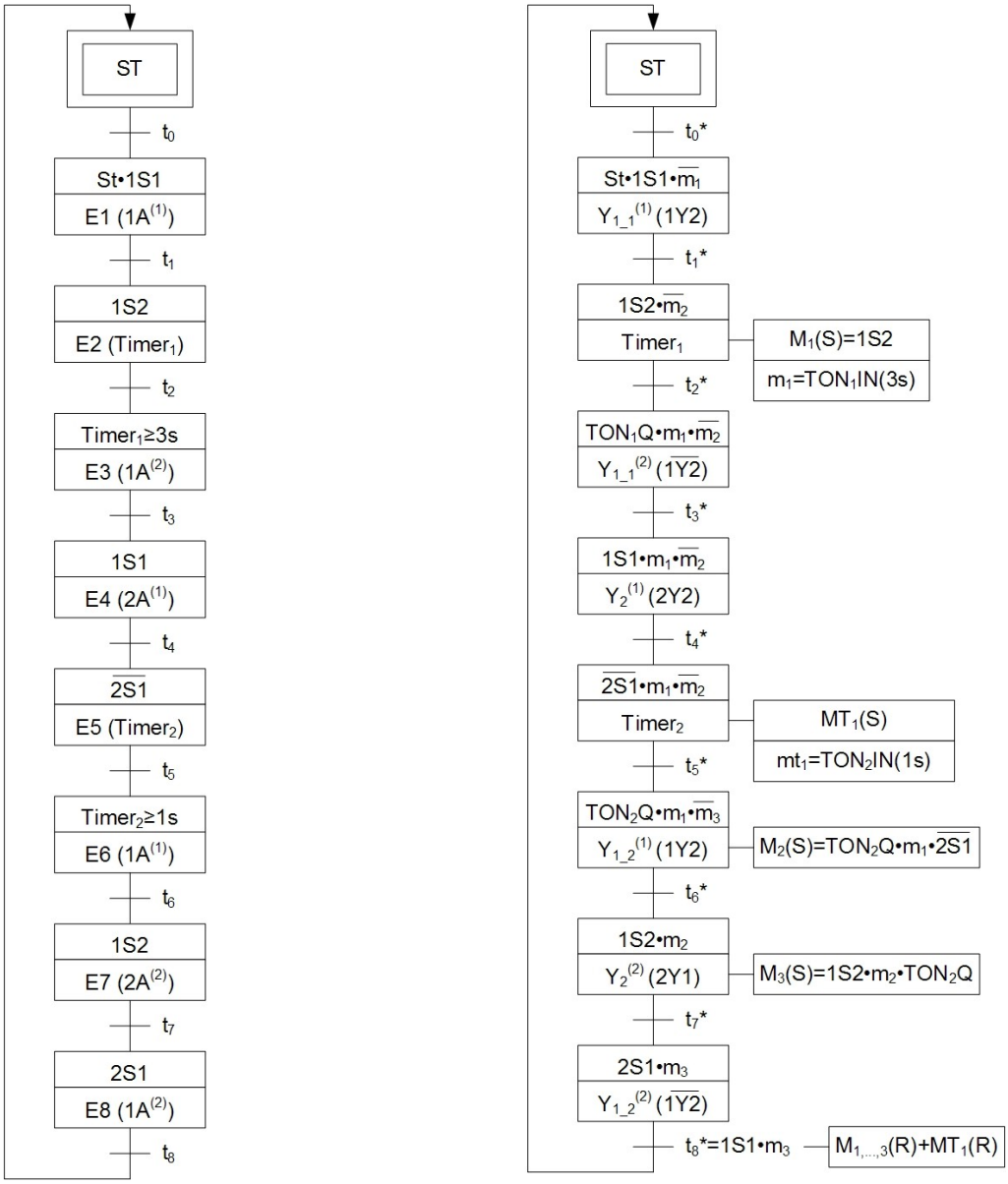
ETAP E5: *odmierzanie czasu*
Realizacja: $Timer_2$(1s)
Sygnalizacja: TON_2Q=1

ETAP E6: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E7: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E8 *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

a) b)

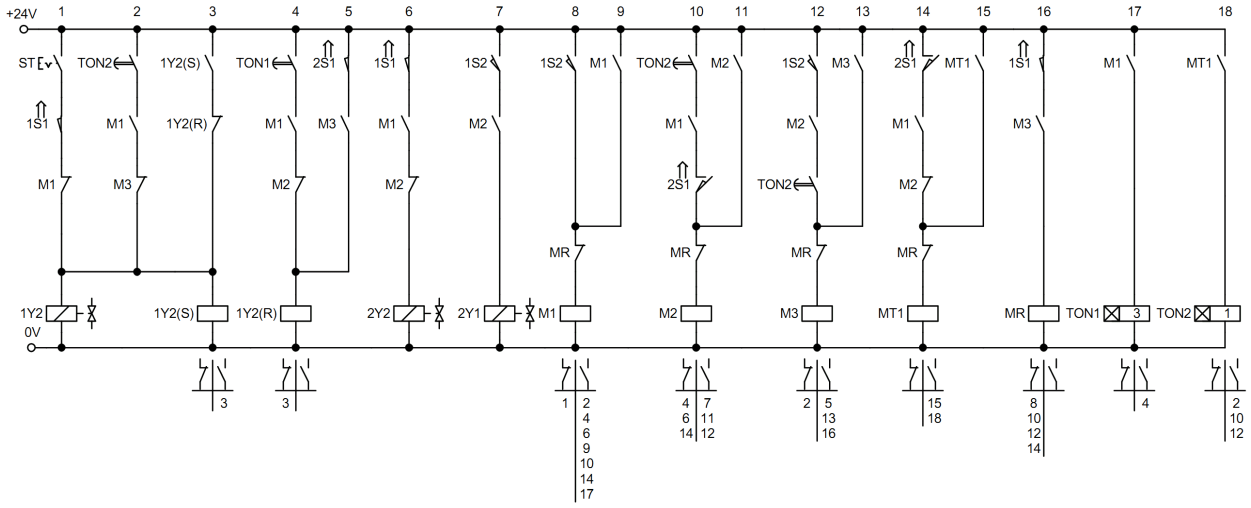


Rys. 9.23. Algorytmy przykładu 3 procedury sekwencyjnej z etapami czasowymi: a) procesu; b) sterowania wraz ze zrealizowaną pamięcią

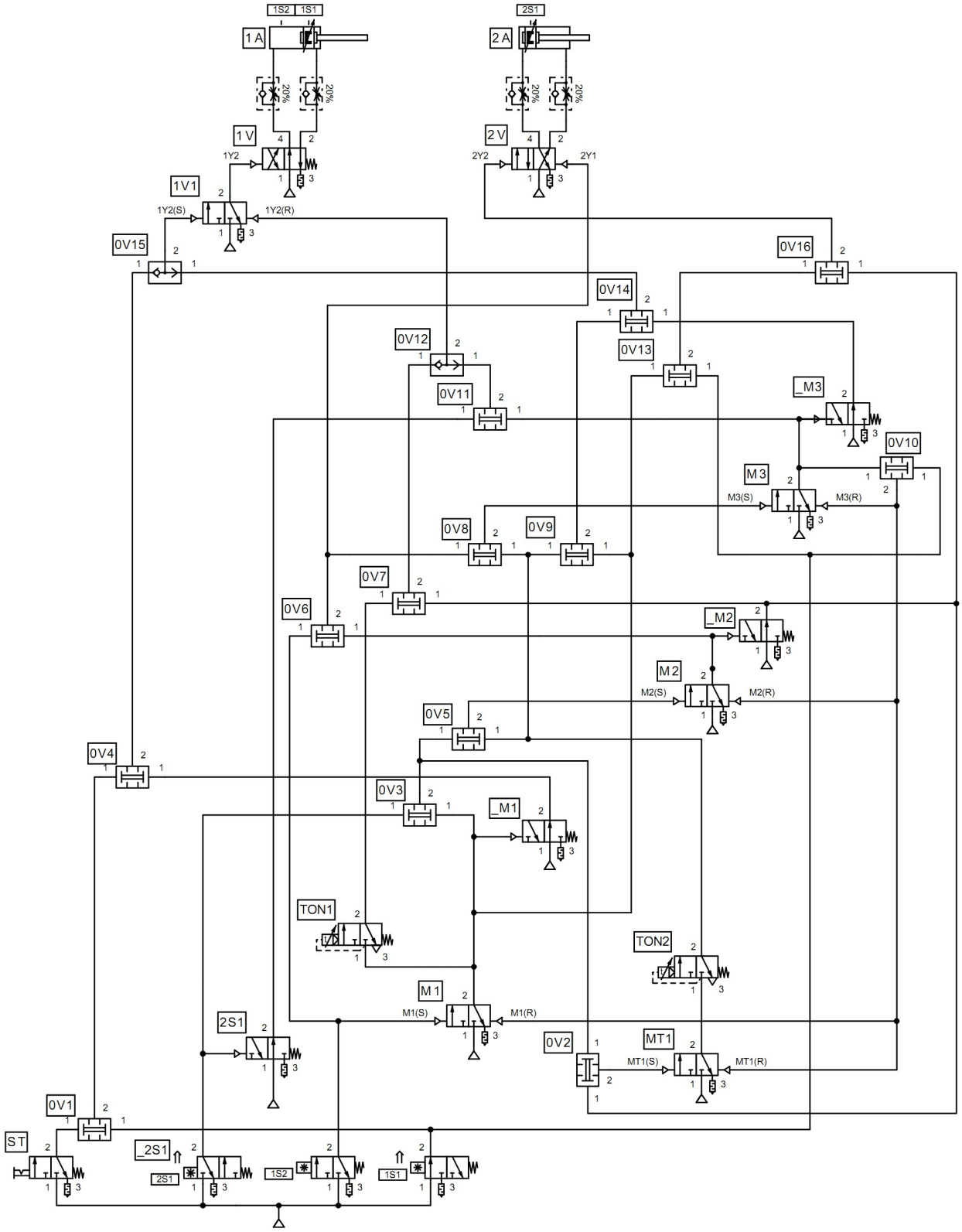
Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.23a). Na podstawie sieci Grafpol GP, stosując opracowane zasady, zaprojektowano algorytm sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.23b). W oparciu o sieć Grafpol GS dokonano syntezy równania schematowego (9.6). Implementację równania schematowego dla stykowo-przekaźnikowego, pneumatycznego i elektronicznego (PLC) układu sterowania przestawiono na rys. 9.24, 9.25 i 9.26.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

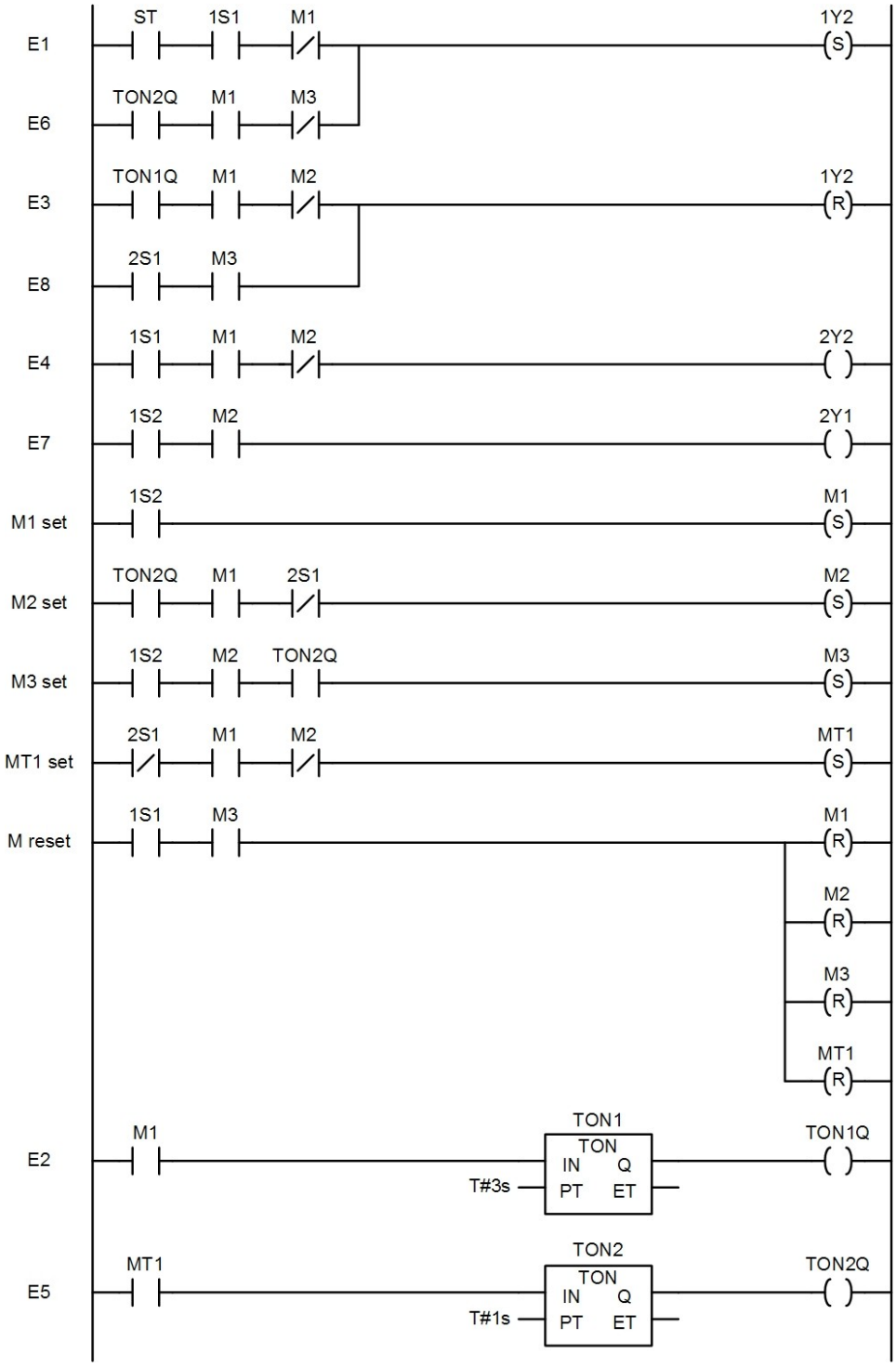
$$
F(Y,M)=\sum
\begin{cases}
ST\cdot 1S1\cdot \overline{m_1}=Y^{(1)}_{1\_1}(S)\\
TON_2Q\cdot m_1\cdot \overline{m_3}=Y^{(1)}_{1\_2}(S)
\end{cases}
ST\cdot 1S1\cdot \overline{m_1}+TON_2Q\cdot m_1\cdot \overline{m_3}=Y^{(1)}_{1}(S),\\
\begin{cases}
TON_1Q\cdot m_1\cdot \overline{m_2}=Y^{(1)}_{1\_1}(R)\\
2S1\cdot m_3=Y^{(1)}_{1\_2}(R)
\end{cases}
TON_1Q\cdot m_1\cdot \overline{m_2}+2S1\cdot m_3=Y^{(1)}_{1}(R),\\
1S1\cdot m_1\cdot \overline{m_2}=Y^{(1)}_{2},\\
1S2\cdot m_2=Y^{(2)}_{2},\\
1S2=M_1(S),\\
TON_2Q\cdot m_1\cdot \overline{2S1}=M_2(S),\\
1S2\cdot m_2\cdot TON_2Q=M_3(S),\\
\overline{2S1}\cdot m_1\cdot \overline{m_2}=MT_1(S),\\
1S1\cdot m_3=M_1(R)+M_2(R)+M_3(R)+MT_1(R)\\
\\
m_1=TON_1\,IN(3s),\\
mt_1=TON_2\,IN(1s),
\tag{9.6}
$$



Rys. 9.24. Schemat stykowo-przekaźnikowego układu sterowania wyznaczonego na podstawie równania schematowego (9.6)



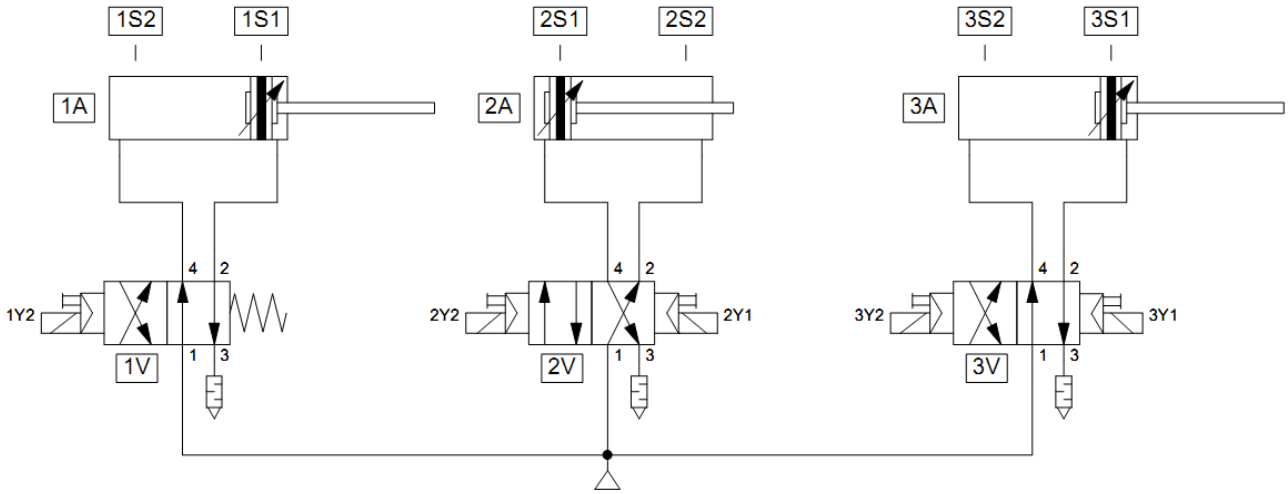
Rys. 9.25. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.6)



Rys. 9.26. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.6)

9.3. Procedury współbieżne

Na rysunku 9.27 zaprezentowano schemat funkcjonalny trzech napędów pneumatycznych 1A, 2A i 3A. Napędy te sterowane są odpowiednio zaworem 1V – monostabilnym i dwoma zaworami bistabilnymi 2V i 3V. Każdy z napędów pneumatycznych wyposażony jest w czujniki położeń krańcowych tłoka. Na poniżej zaprezentowanych przykładach, odnoszących się do procedur współbieżnych, schemat funkcjonalny uzupełniono o opis słowny jednoznacznie określający algorytm pracy poszczególnych napędów. Opis słowny składa się z nazwy symbolicznej etapu (np. E2) i jego opisu (*wsuw tłoczyska siłownika 1A*). Ponadto zawiera informacje odnośnie do działania wykonywanego w danym etapie (np. $1A^{(1)}$ - pierwszy ruch napędu 1A), sposobu jego realizacji (np. podanie prądu na cewkę 1Y1) i oznaczenia czujników (np. 1S1) których sygnały informują o zakończeniu wykonania poszczególnych etapów.



Rys. 9.27. Schemat funkcjonalny trzech napędów pneumatycznych

9.3.1. Przykład 1

Schemat funkcjonalny napędów zaprezentowano na rys. 9.27. Algorytm procesu stanowią dwie procedury sekwencyjne (PS1 i PS2) realizowane równocześnie o opisie słownym:

Procedura sekwencyjna 1 (PS1)
ETAP E1: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E2: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

ETAP E3: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

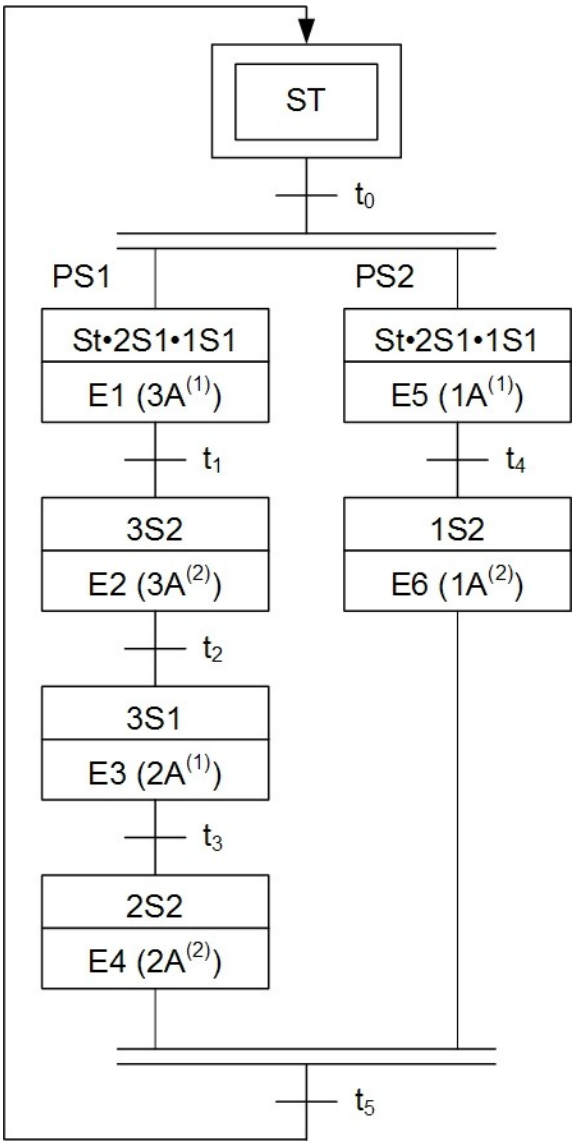
ETAP E4: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

Procedura sekwencyjna 2 (PS2)
ETAP E5: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

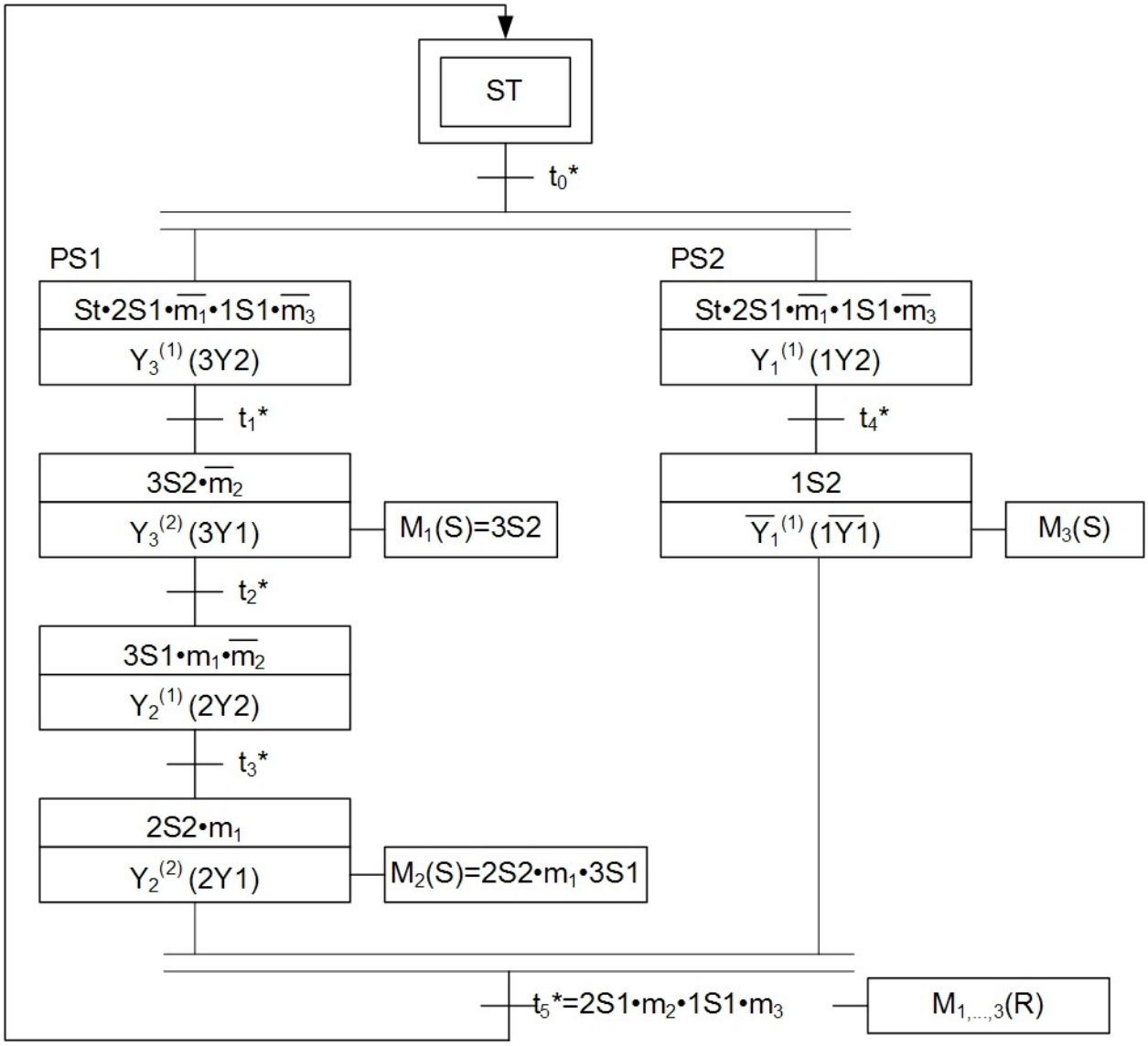
ETAP E6: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.28). Stosując opracowane zasady syntezy algorytmów sekwencyjnych, zaprojektowano algorytmy sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.29). Na podstawie tej sieci dokonano syntezy równania schematowego (9.7). Implementację równania schematowego dla realizacji za pomocą elementów pneumatycznych, stykowo-przekaźnikowych i sterownika PLC przestawiono kolejno na rys. 9.30, 9.31 i 9.32.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

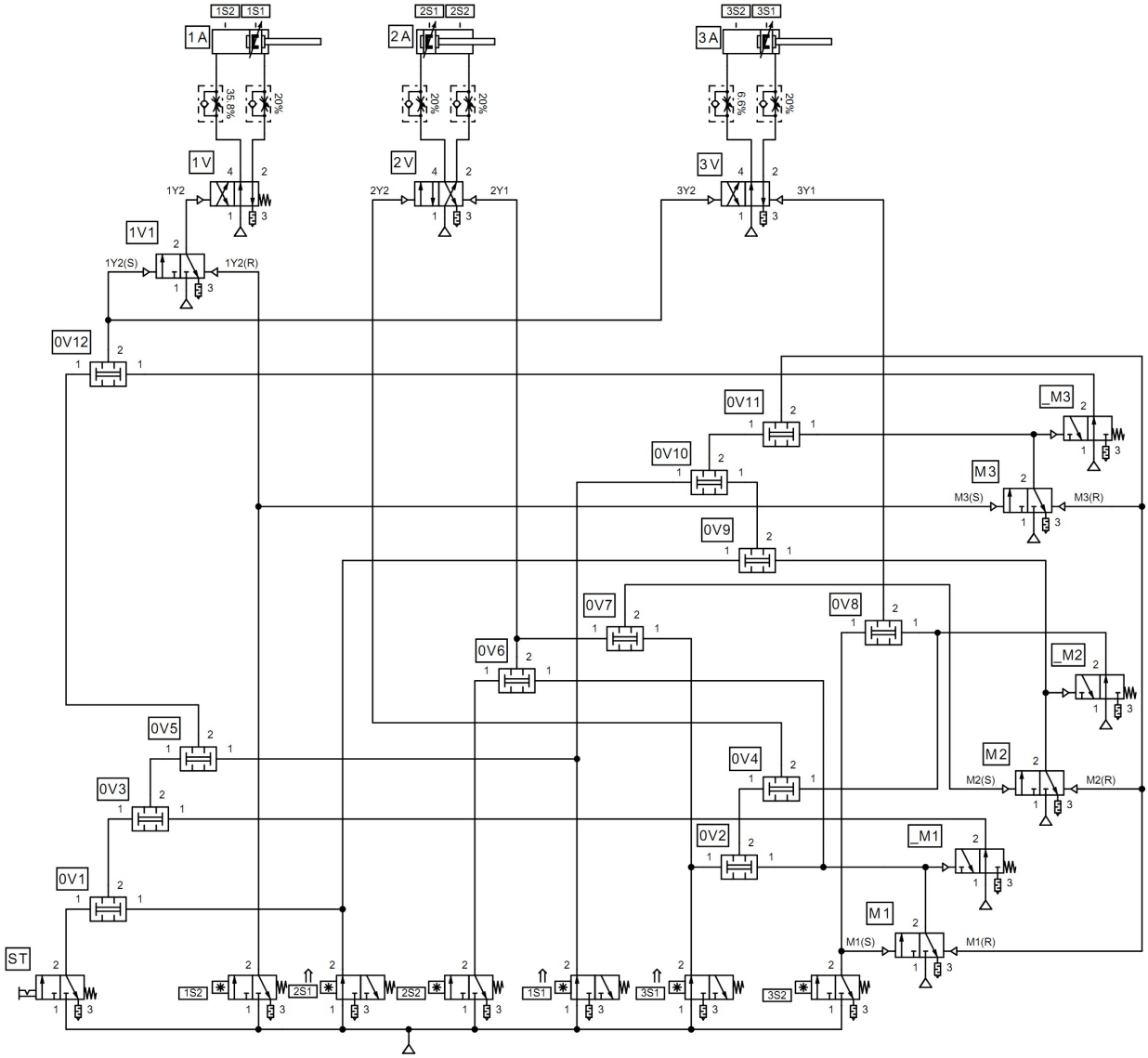


Rys. 9.28. Algorytm procesu przykładu 1 procedur współbieżnych

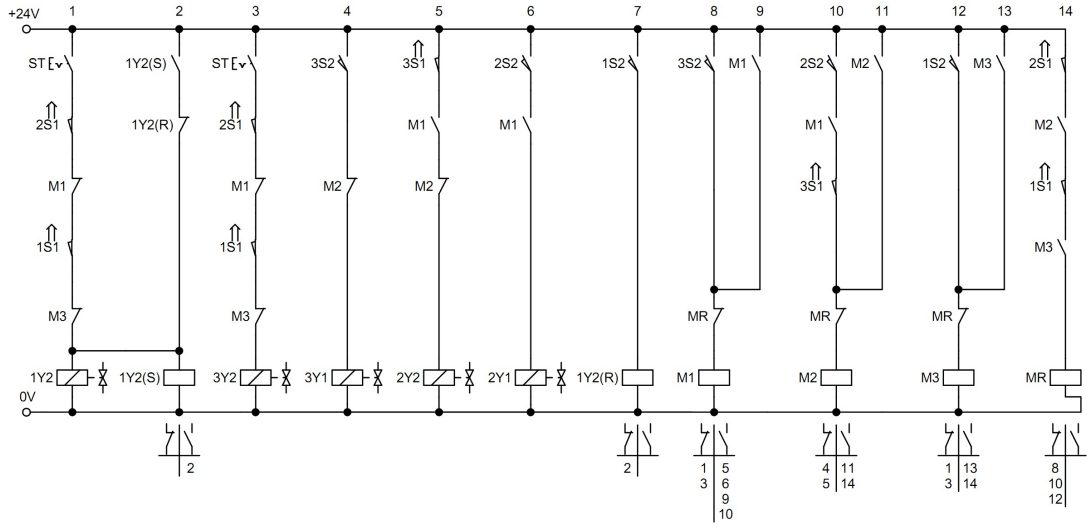


Rys. 9.29. Algorytm sterowania wraz ze zrealizowaną pamięcią przykładu 1 procedur współbieżnych

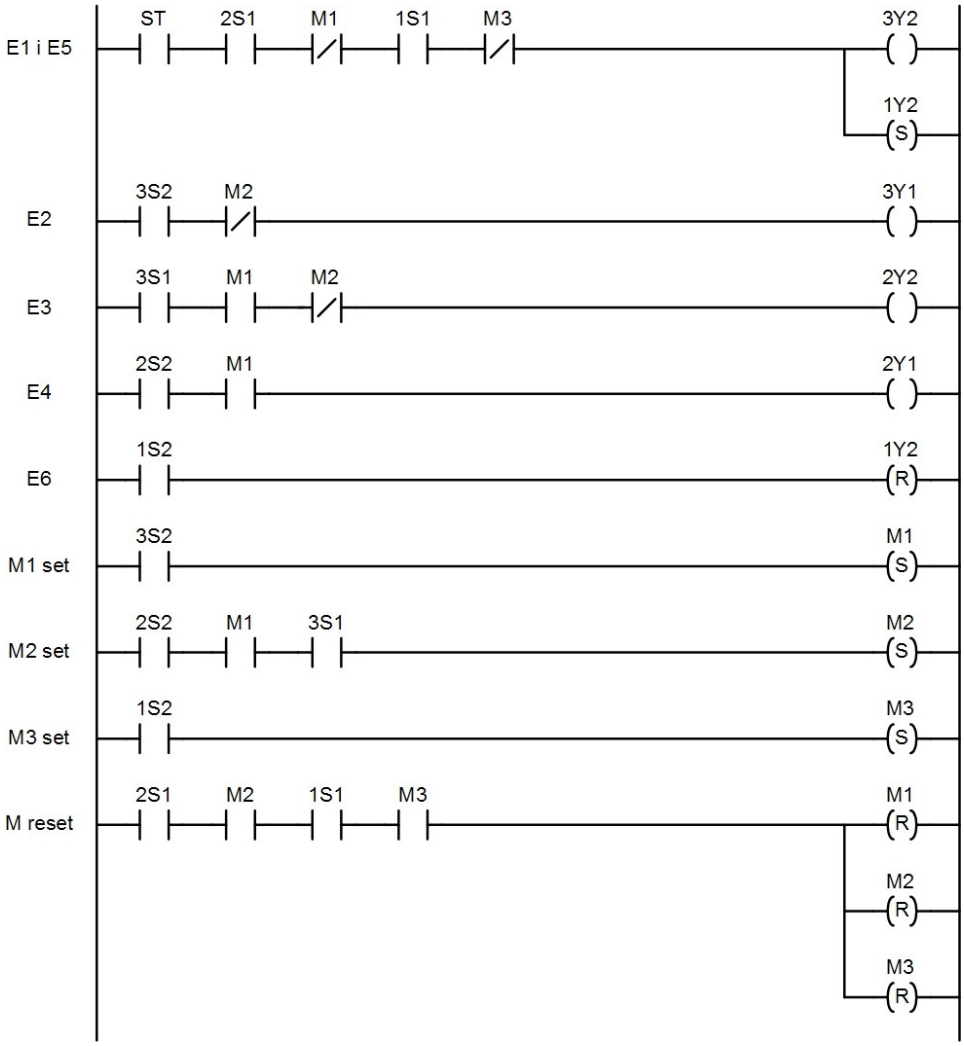
$$F(Y,M)=\sum \begin{cases} ST\cdot 2S1\cdot \overline{m_1}\cdot 1S1\cdot \overline{m_3}=Y_3^{(1)}+Y_1^{(1)}(S), \\ 3S2\cdot \overline{m_2}=Y_3^{(2)}, \\ 3S1\cdot m_1\cdot \overline{m_2}=Y_2^{(1)}, \\ 2S2\cdot m_1=Y_2^{(2)}, \\ 1S2=Y_1^{(1)}(R), \\ \\ 3S2=M_1(S), \\ 2S2\cdot m_1\cdot 3S1=M_2(S), \\ 1S2=M_3(S), \\ 2S1\cdot m_2\cdot 1S1\cdot m_3=M_1(R)+M_2(R)+M_3(R) \end{cases} \tag{9.7}$$



Rys. 9.30. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.7)



Rys. 9.31. Schemat stykowo-przekaźnikowego układu sterowania wyznaczony na podstawie równania schematowego (9.7)



Rys. 9.32. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.7)

9.3.2. Przykład 2

Schemat funkcjonalny napędów zaprezentowano na rys. 9.27. Algorytm procesu stanowią dwie procedury sekwencyjne (PS1) i (PS2) realizowane równocześnie, o opisie słownym:

Procedura sekwencyjna 1 (PS1)
ETAP E1: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E2: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

ETAP E3: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

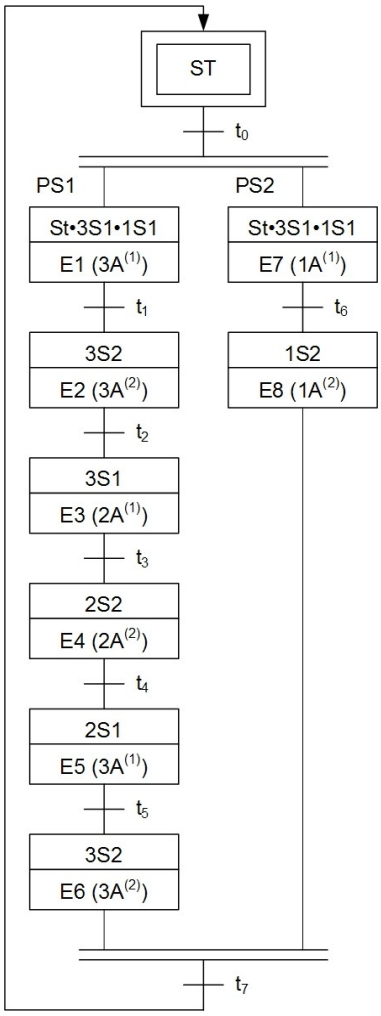
ETAP E4: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E5: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

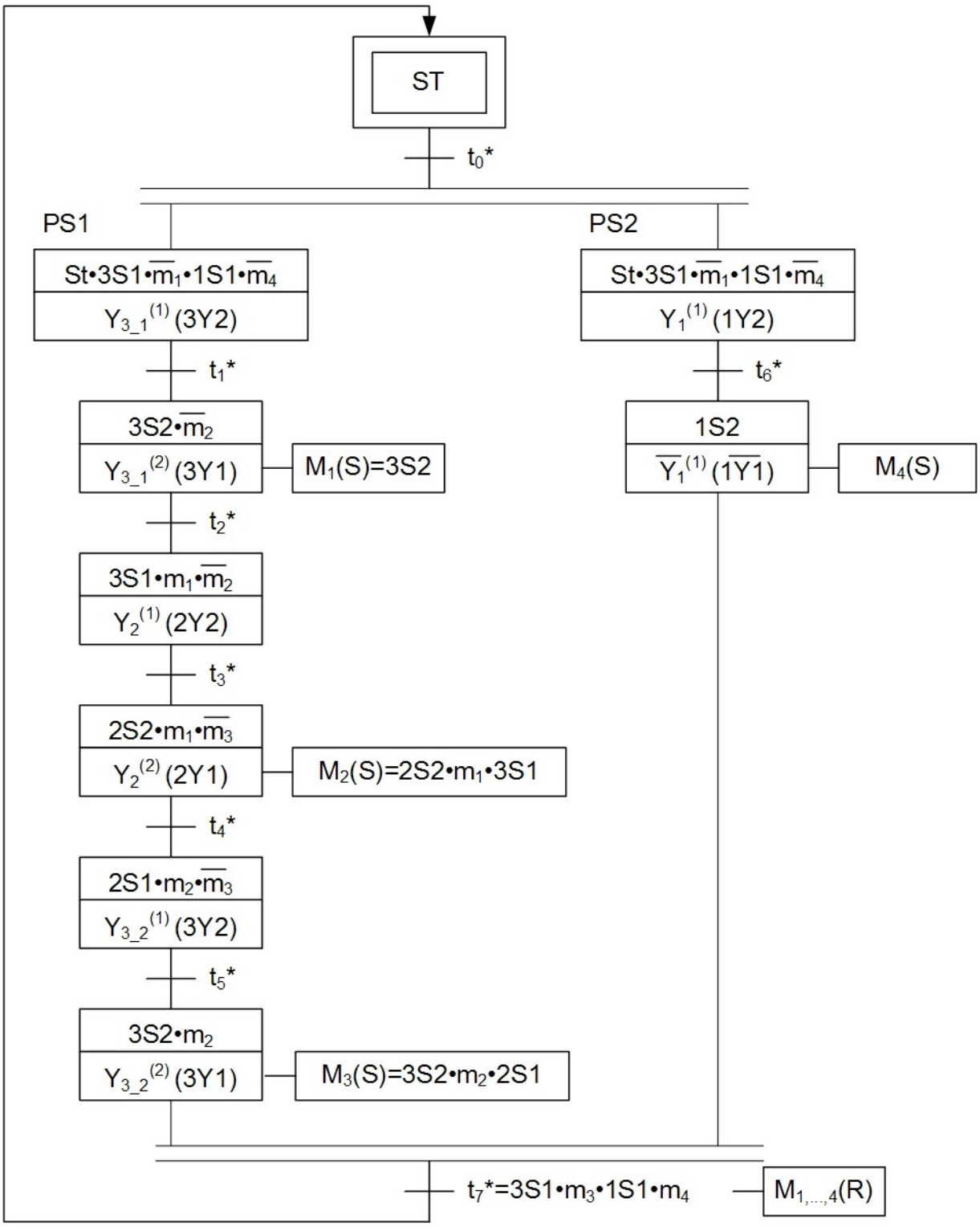
ETAP E6: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

Procedura sekwencyjna 2 (PS2)
ETAP E7: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E8: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1



Rys. 9.33. Algorytm procesu przykładu 2 procedur współbieżnych

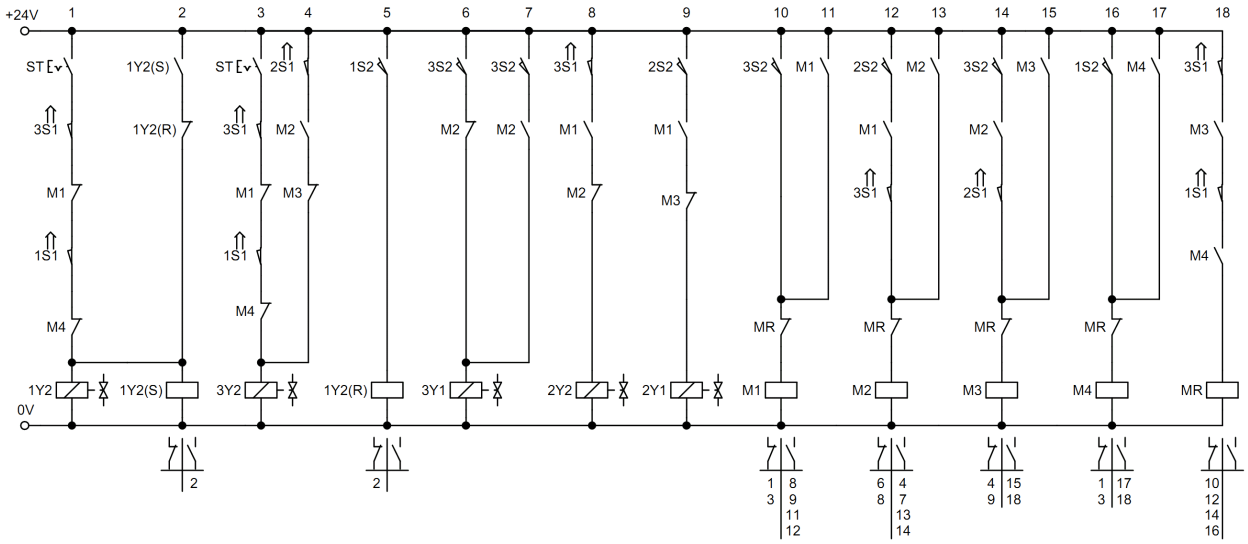


Rys. 9.34. Algorytm sterowania wraz ze zrealizowaną pamięcią przykładu 2 procedur współbieżnych

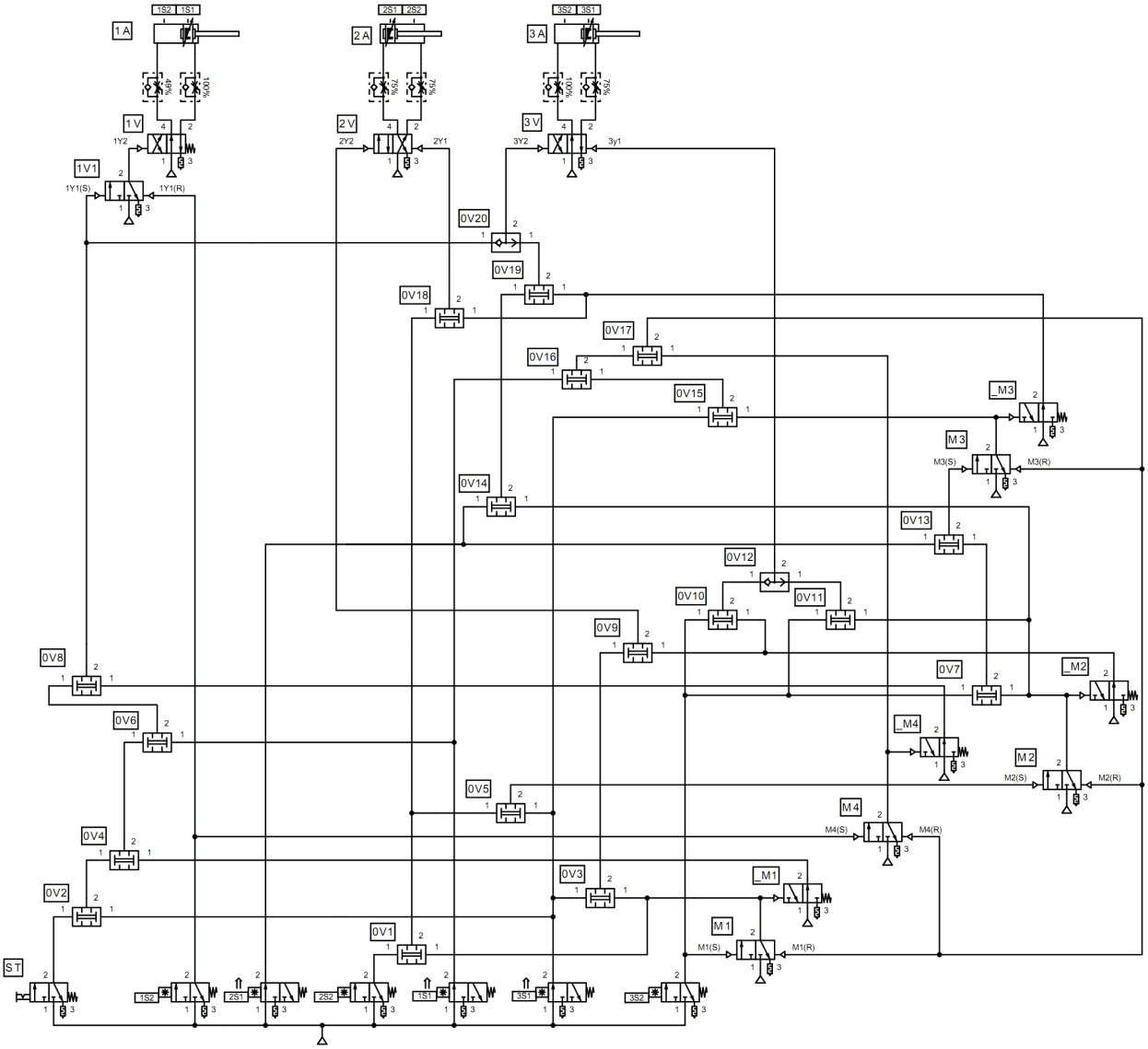
Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.33). Stosując opracowane zasady zaprojektowano algorytmy sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.34). Na podstawie tej sieci dokonano syntezy równania schematowego (9.8). Implementację równania schematowego dla realizacji za pomocą elementów stykowo-przekaźnikowych, pneumatycznych i sterownika PLC przestawiono kolejno na rys. 9.35, 9.36 i 9.37.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

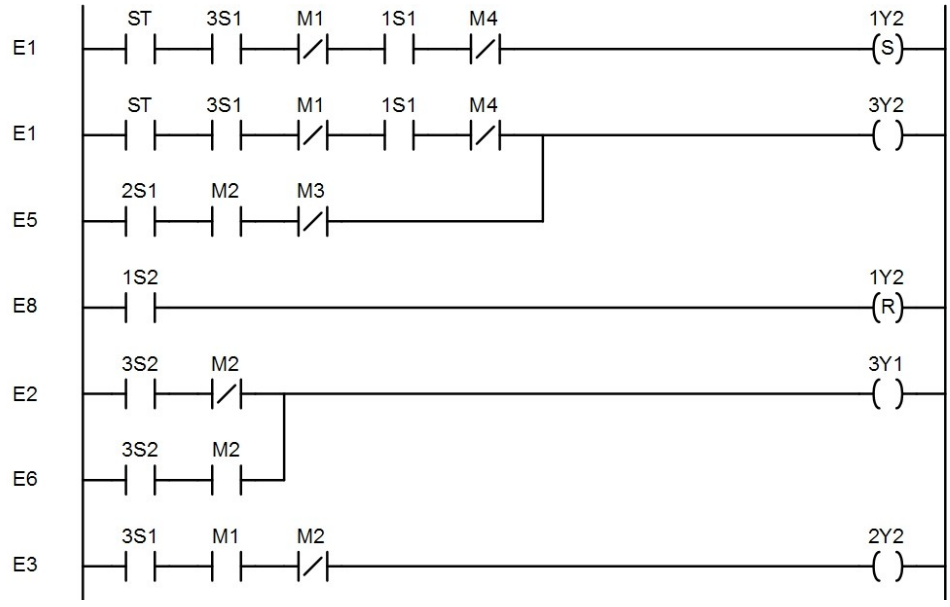
$$
F(Y,M)=\sum
\begin{array}{l}
ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_1^{(1)}(S),\\
\left.\begin{array}{l} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_{3\_1}^{(1)} \\ 2S1\cdot m_2\cdot\overline{m_3}=Y_{3\_2}^{(1)} \end{array}\right\} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}+2S1\cdot m_2\cdot\overline{m_3}=Y_3^{(1)},\\
1S2=Y_1^{(1)}(R),\\
\left.\begin{array}{l} 3S2\cdot\overline{m_2}=Y_{3\_1}^{(2)} \\ 3S2\cdot m_2=Y_{3\_2}^{(2)} \end{array}\right\} 3S2\cdot\overline{m_2}+3S2\cdot m_2=Y_3^{(2)},\\
3S1\cdot m_1\cdot\overline{m_2}=Y_2^{(1)},\\
2S2\cdot m_1\cdot\overline{m_3}=Y_2^{(2)},\\
\\
3S2=M_1(S),\\
2S2\cdot m_1\cdot 3S1=M_2(S),\\
3S2\cdot m_2\cdot 2S1=M_3(S),\\
1S2=M_4(S),\\
3S1\cdot m_3\cdot 1S1\cdot m_4=M_1(R)+M_2(R)+M_3(R)+M_4(R)
\end{array}
\tag{9.8}
$$

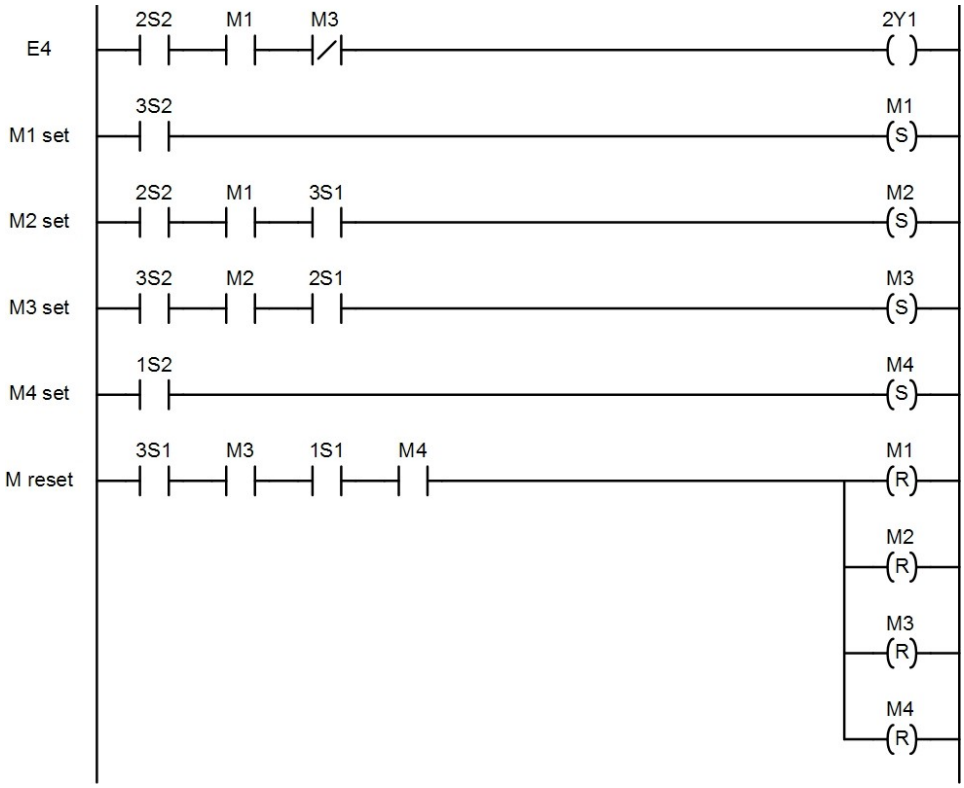


Rys. 9.35. Schemat stykowo-przekaźnikowego układu sterowania wyznaczonego na podstawie równania schematowego (9.8)



Rys. 9.36. Schemat układu sterowania realizowanego za pomocą elementów pneumatycznych wyznaczony na podstawie równania schematowego (9.8)





Rys. 9.37. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.8)

9.3.3. Przykład 3

Schemat funkcjonalny napędów zaprezentowano na rys. 9.27. Algorytm procesu stanowią dwie procedury sekwencyjne (PS1 i PS2) realizowane równocześnie o opisie słownym:

Procedura sekwencyjna 1 (PS1)
ETAP E1: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E2: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

ETAP E3: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

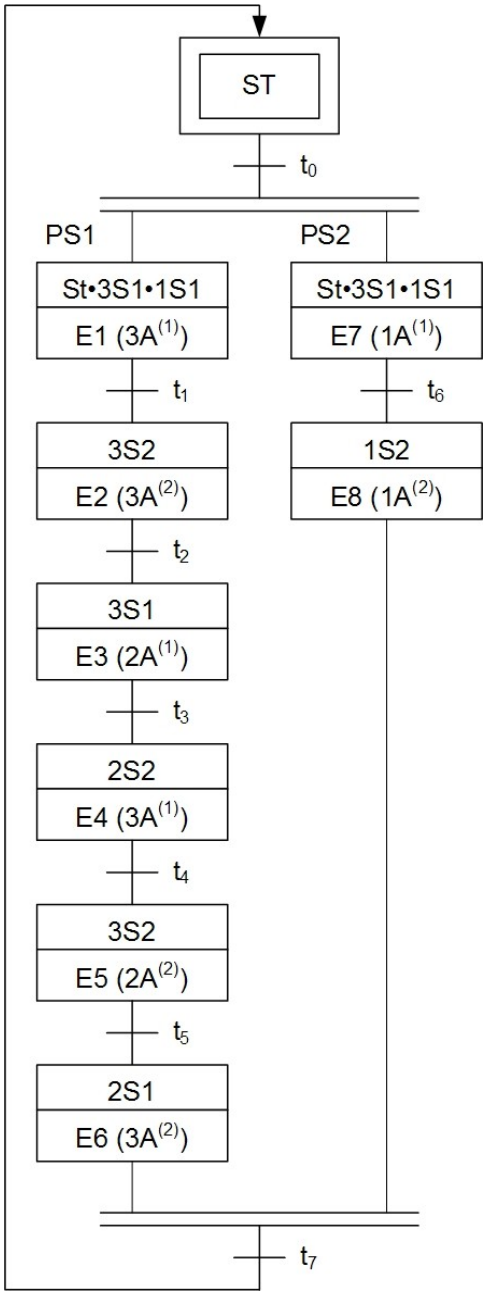
ETAP E4: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E5: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

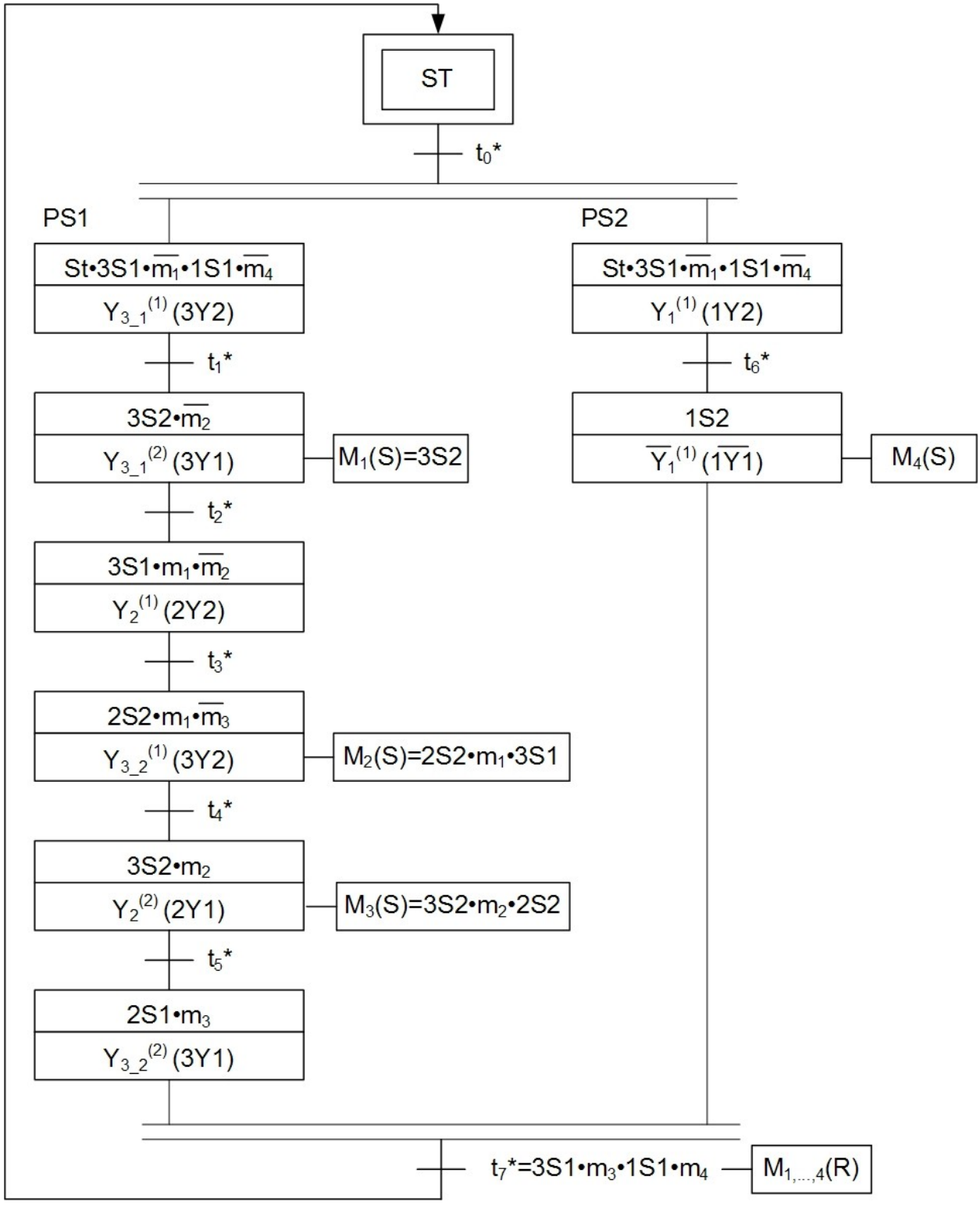
ETAP E6: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

Procedura sekwencyjna 2 (PS2)
ETAP E7: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: 1S2=1

ETAP E8: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1



Rys. 9.38. Algorytm procesu przykładu 3 procedur współbieżnych



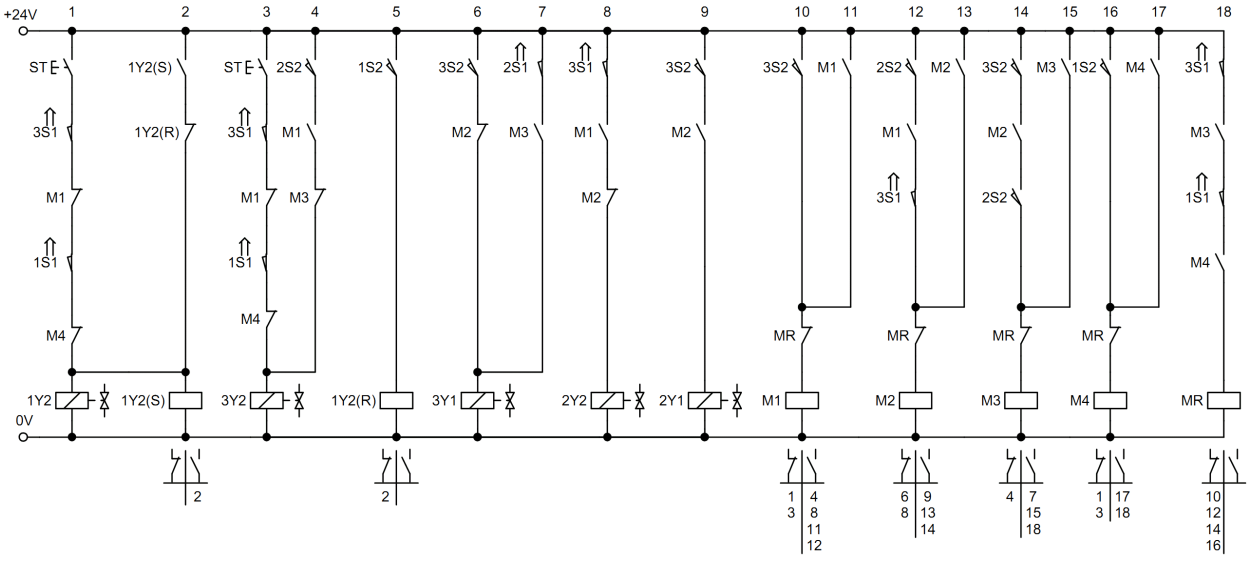
Rys. 9.39. Algorytm sterowania wraz ze zrealizowaną pamięcią przykładu 3 procedur współbieżnych

Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.38). Stosując opracowane zasady, zaprojektowano algorytmy sterowania wraz ze zrealizowaną pamięcią reprezentowane siecią Grafpol GS (rys. 9.39). Na podstawie tej sieci dokonano syntezy równania schematowego (9.9). Implementację równania schematowego dla realizacji za pomocą elementów stykowo-przekaźnikowych, pneumatycznych i sterownika PLC przestawiono kolejno na rys. 9.40, 9.41 i 9.42.

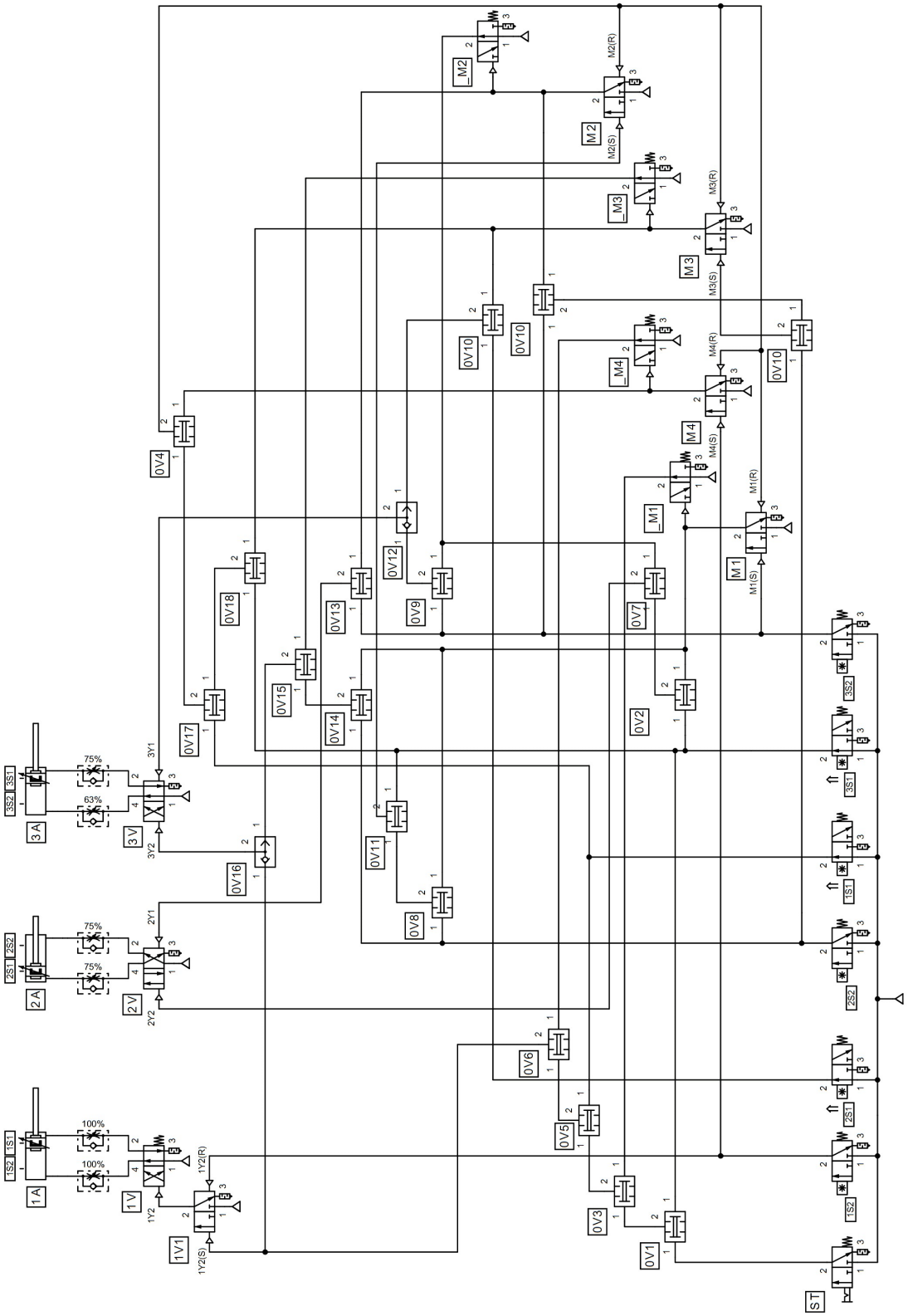
W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

$$
F(Y,M)=\sum
\begin{cases}
ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_1^{(1)}(S),\\
\left.\begin{array}{l} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_{3\_1}^{(1)}\\ 2S1\cdot m_1\cdot\overline{m_3}=Y_{3\_2}^{(1)}\end{array}\right\} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}+2S2\cdot m_1\cdot\overline{m_3}=Y_3^{(1)},\\
1S2=Y_1^{(1)}(R),\\
\left.\begin{array}{l} 3S2\cdot\overline{m_2}=Y_{3\_1}^{(2)}\\ 2S1\cdot m_3=Y_{3\_2}^{(2)}\end{array}\right\} 3S2\cdot\overline{m_2}+2S1\cdot m_3=Y_3^{(2)},\\
3S1\cdot m_1\cdot\overline{m_2}=Y_2^{(1)},\\
3S2\cdot m_2=Y_2^{(2)},
\end{cases}
\qquad (9.9)
$$

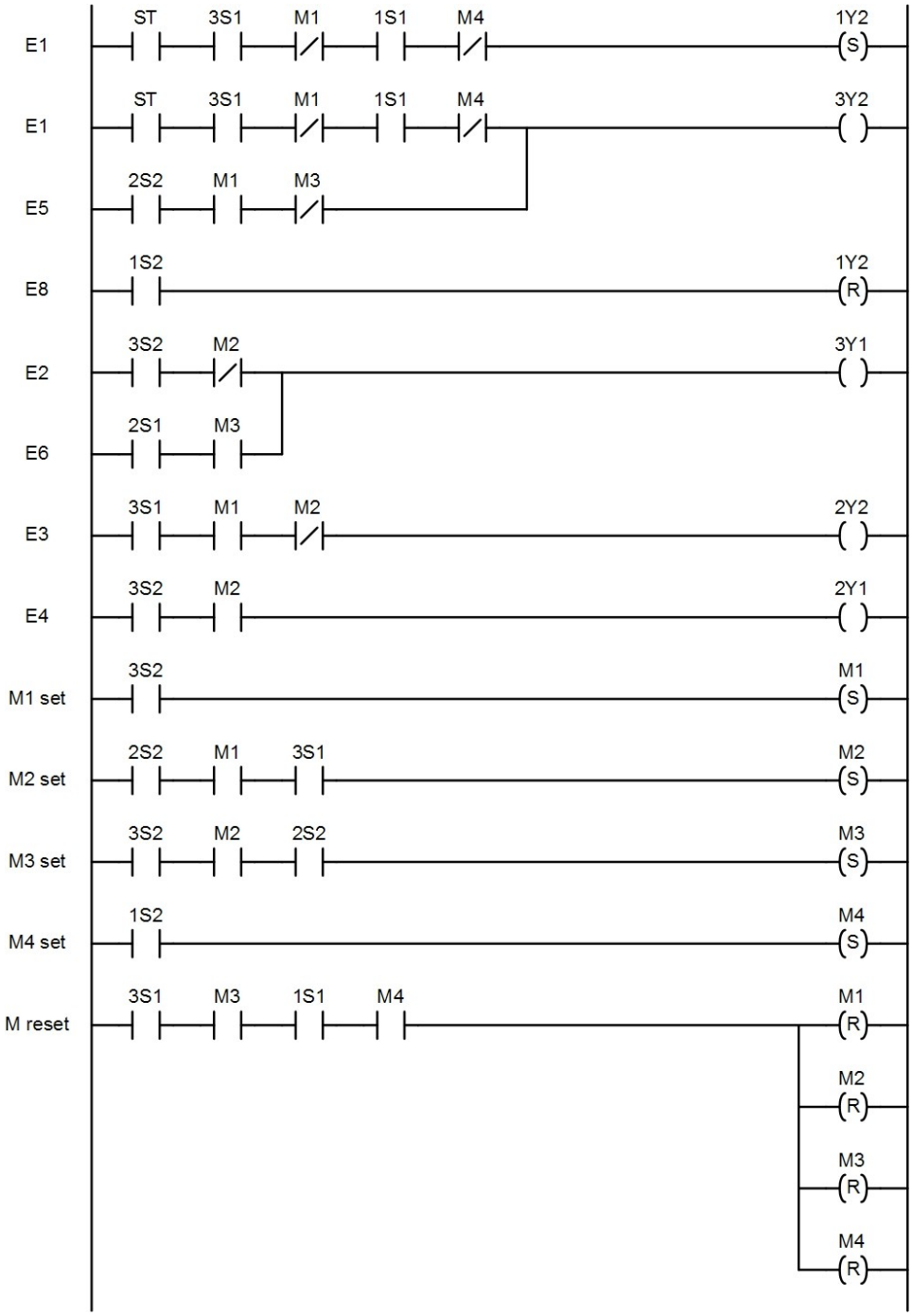
$$
\begin{cases}
3S2=M_1(S),\\
2S2\cdot m_1\cdot 3S1=M_2(S),\\
3S2\cdot m_2\cdot 2S2=M_3(S),\\
1S2=M_4(S),\\
3S1\cdot m_3\cdot 1S1\cdot m_4=M_1(R)+M_2(R)+M_3+M_4(R)
\end{cases}
$$



Rys. 9.40. Schemat układu stykowo-przekaźnikowego wyznaczony na podstawie równania schematowego (9.9)



Rys. 9.41. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.9)



Rys. 9.42. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.9)

9.4. Procedury współbieżne z etapami czasowymi

9.4.1. Przykład 1

Schemat funkcjonalny pracy napędów zaprezentowano na rys. 9.27. Algorytm procesu stanowią dwie procedury sekwencyjne (PS1 i PS2) realizowane równocześnie o opisie słownym:

Procedura sekwencyjna 1 (PS1)
ETAP E1: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E2: *odmierzanie czasu*
Realizacja: $Timer_1$(2s)
Sygnalizacja: TON_1Q=1

ETAP E3: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

ETAP E4: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E5: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

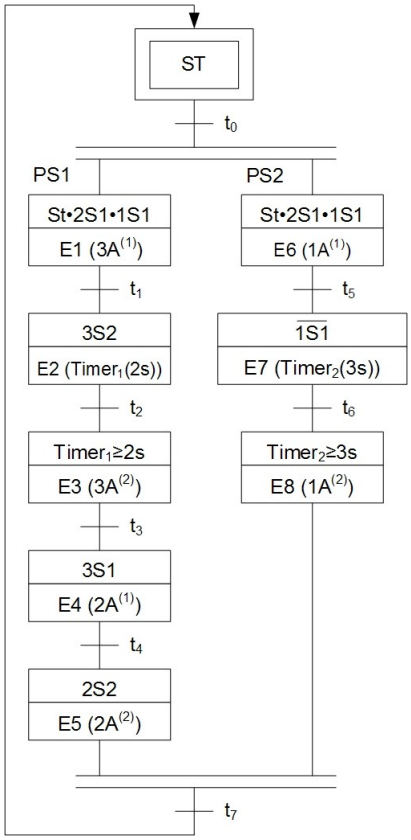
Procedura sekwencyjna 2 (PS2)
ETAP E6: *wsuw tłoczyska siłownika 1A przez zadany czas*
Realizacja: $1A^{(1)}$ (1Y2)
Sygnalizacja: $\overline{1S1}$=1

ETAP E7: *odmierzanie czasu*
Realizacja: $Timer_2$(3s)
Sygnalizacja: TON_2Q=1

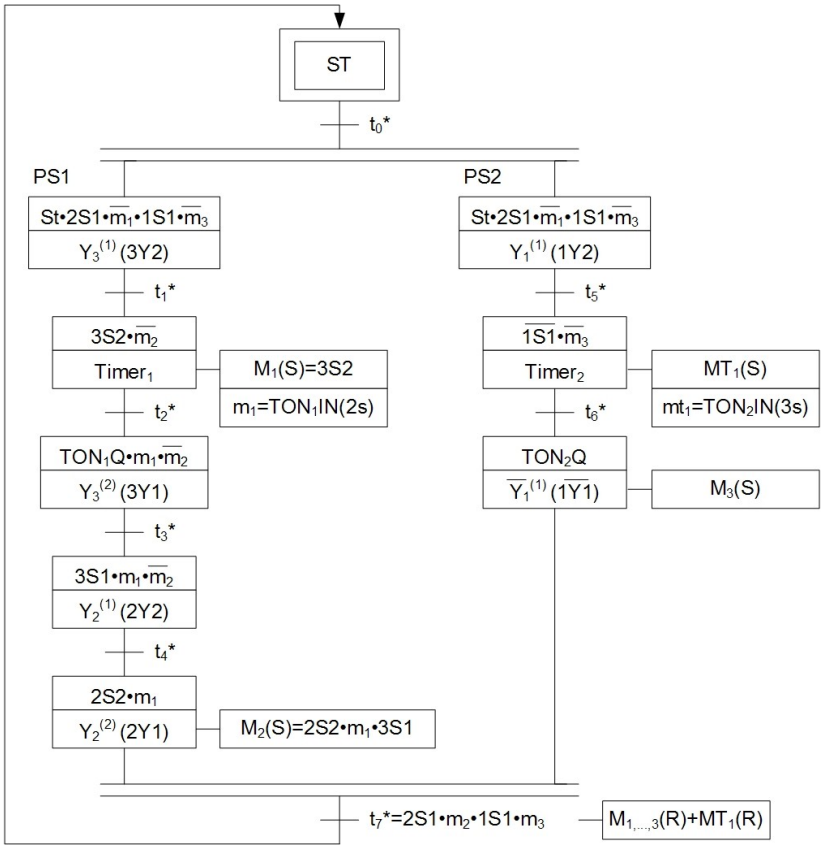
ETAP E8: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.43). Stosując opracowane zasady, zaprojektowano algorytmy sterowania wraz ze zrealizowaną pamięcią reprezentowane siecią Grafpol GS (rys. 9.44). Na podstawie tej sieci dokonano syntezy równania schematowego (9.10). Implementację równania schematowego dla realizacji za pomocą elementów stykowo-przekaźnikowych, pneumatycznych i sterownika PLC przestawiono kolejno na rys. 9.45, 9.46 i 9.47.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

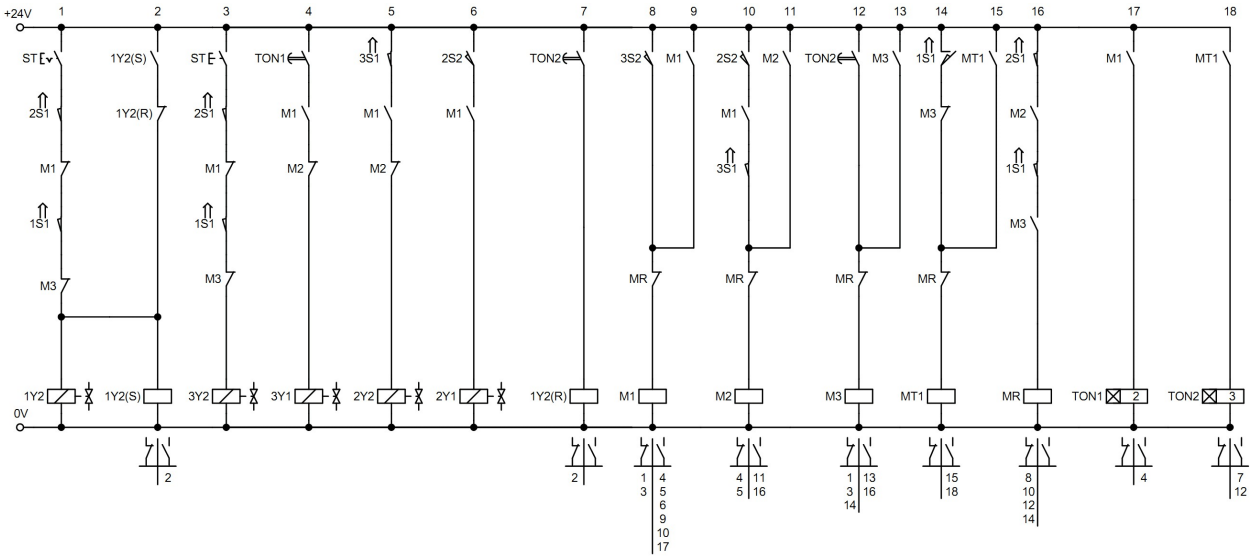


Rys. 9.43. Algorytm procesu przykładu 1 procedur współbieżnych z etapami czasowymi

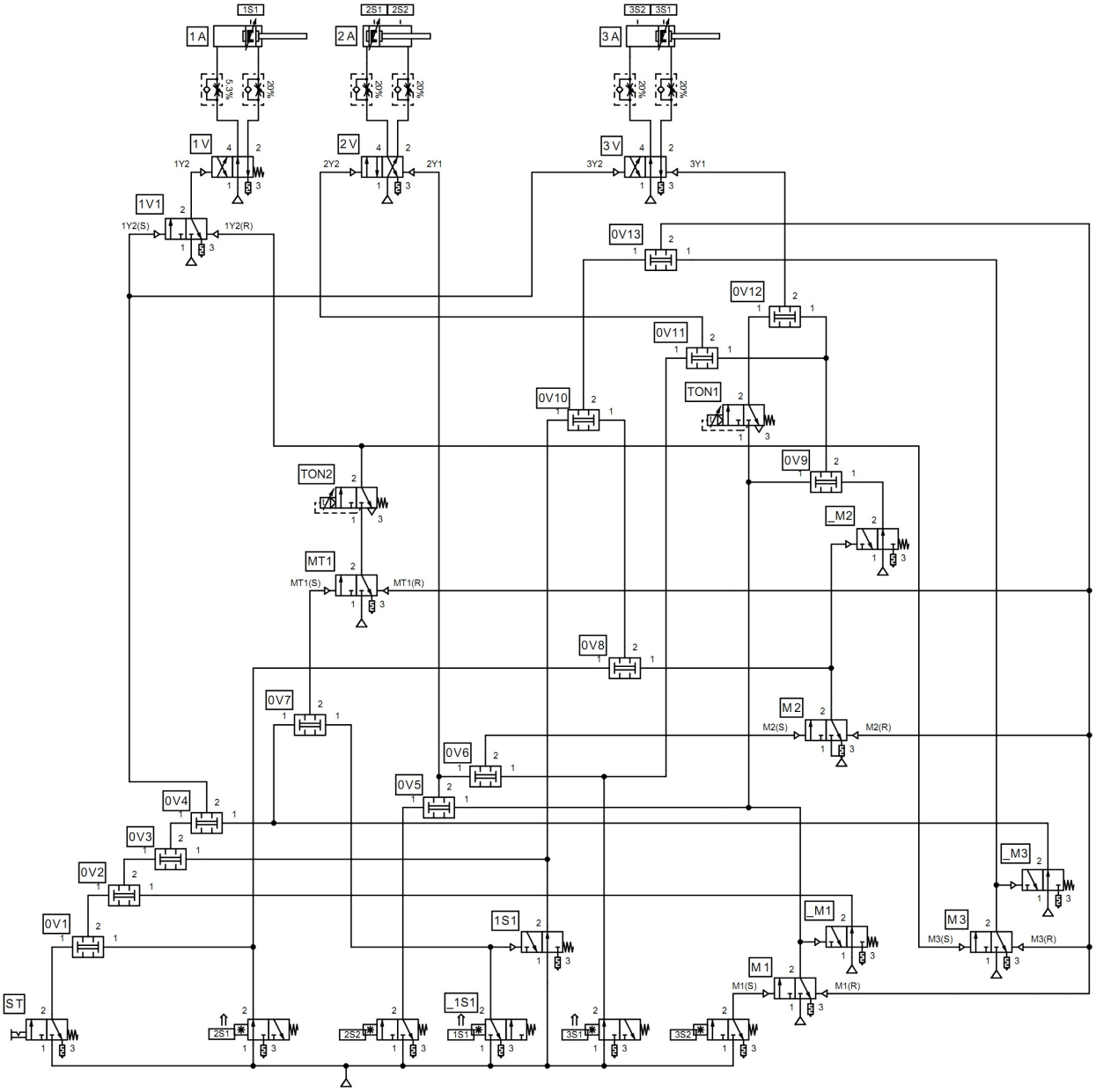


Rys. 9.44. Algorytm sterowania wraz ze zrealizowaną pamięcią przykładu 1 procedur współbieżnych z etapami czasowymi

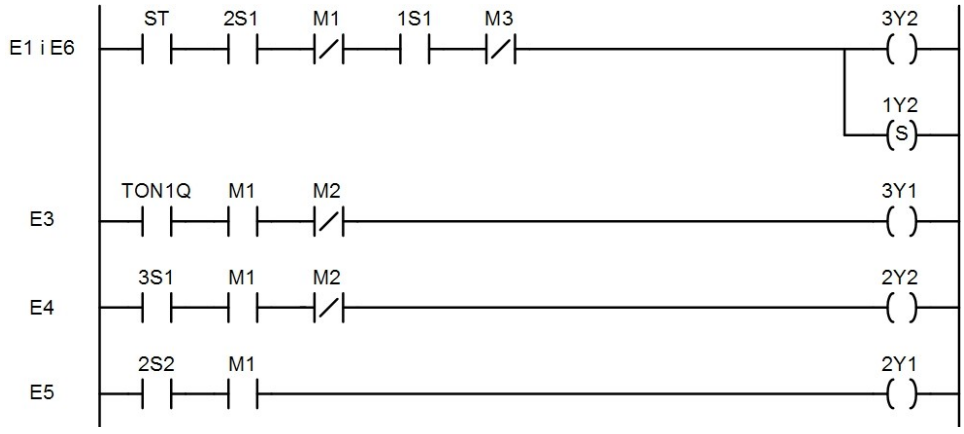
$$
\begin{aligned}
& ST \cdot 2\mathrm{S1} \cdot \overline{m_1} \cdot 1\mathrm{S1} \cdot \overline{m_3} = Y_3^{(1)} + Y_1^{(1)}(S), \\
& TON_1 Q \cdot m_1 \cdot \overline{m_2} = Y_3^{(2)}, \\
& 3\mathrm{S1} \cdot m_1 \cdot \overline{m_2} = Y_2^{(1)}, \\
& 2\mathrm{S2} \cdot m_1 = Y_2^{(2)}, \\
& TON_2 Q = Y_1^{(1)}(R), \\
F(Y,M) = \sum \; & 3\mathrm{S2} = M_1(S), \\
& 2\mathrm{S2} \cdot m_1 \cdot 3\mathrm{S1} = M_2(S), \\
& TON_2 Q = M_3(S), \\
& \overline{1\mathrm{S1}} \cdot \overline{m_3} = MT_1(S), \\
& 2\mathrm{S1} \cdot m_2 \cdot 1\mathrm{S1} \cdot m_3 = M_1(R) + M_2(R) + M_3(R) + MT_1(R) \\
& \\
& m_1 = TON_1\, IN(2\mathrm{s}), \\
& mt_1 = TON_2\, IN(3\mathrm{s}),
\end{aligned}
\tag{9.10}
$$

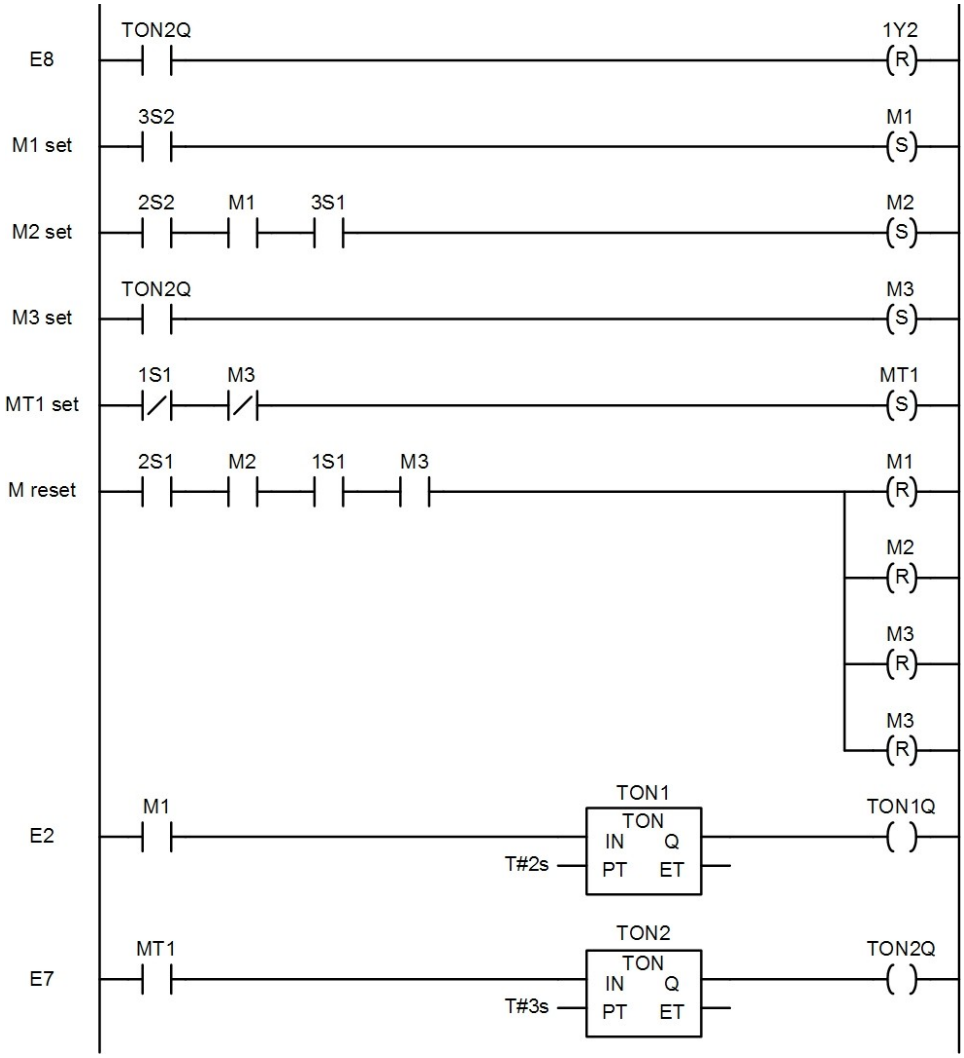


Rys. 9.45. Schemat układu stykowo-przekaźnikowego wyznaczony na podstawie równania schematowego (9.10)



Rys. 9.46. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.10)





Rys. 9.47. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.10)

9.4.2. Przykład 2

Schemat funkcjonalny napędów zaprezentowano na rys. 9.27. Algorytm procesu stanowią dwie procedury sekwencyjne (PS1 i PS2) realizowane równocześnie o opisie słownym:

Procedura sekwencyjna 1 (PS1)
ETAP E1: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E2: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

ETAP E3: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E4: *odmierzanie czasu*
Realizacja: $Timer_1(2s)$
Sygnalizacja: $TON_1Q=1$

ETAP E5: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E6: *wsuw tłoczyska siłownika 3A przez zadany czas*
Realizacja: $3A^{(1)}$ ($\overline{3Y2}$)
Sygnalizacja: $\overline{3S1}$=1

ETAP E7: *odmierzanie czasu*
Realizacja: $Timer_2(1s)$
Sygnalizacja: $TON_2Q=1$

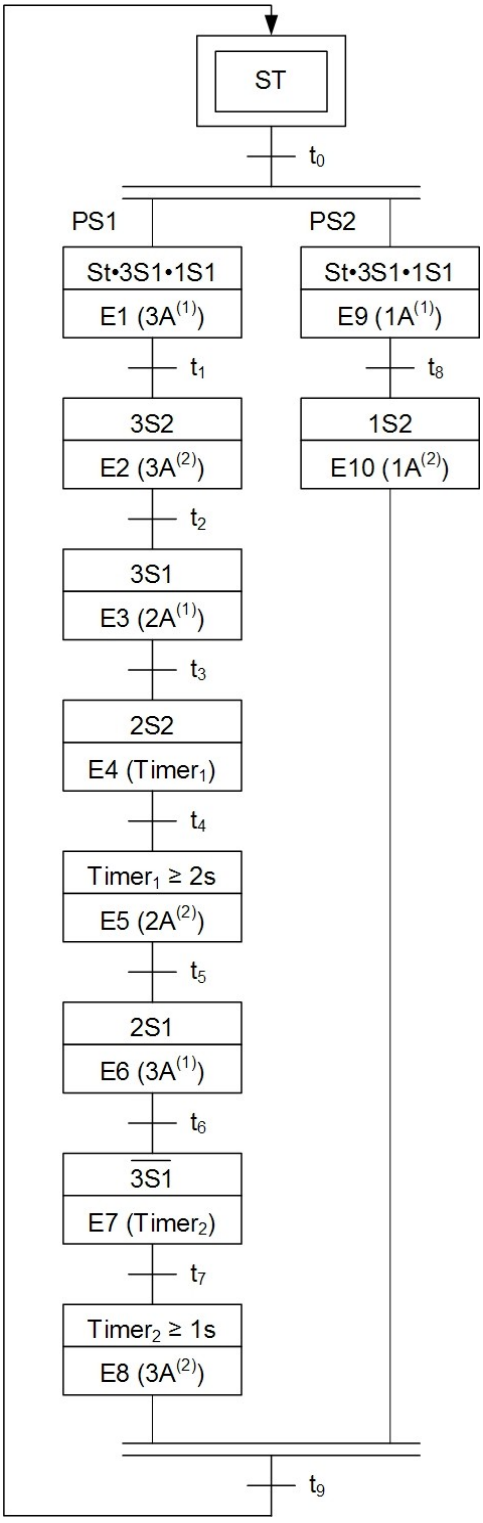
ETAP E8: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

Procedura sekwencyjna 2 (PS2)
ETAP E9: *wsuw tłoczyska siłownika 1A*
Realizacja: $1A^{(1)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S2=1

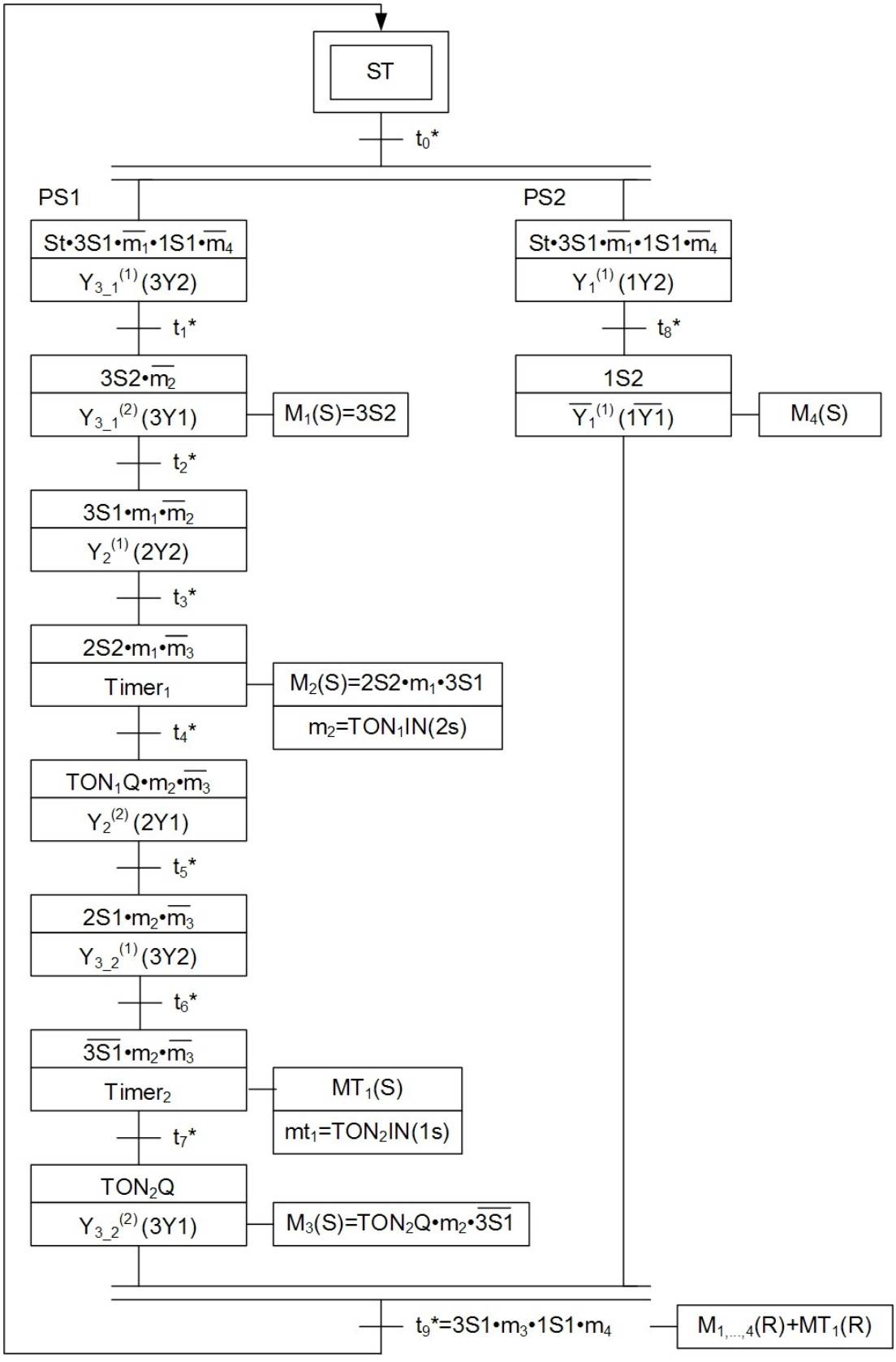
ETAP E10: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.48). Stosując opracowane zasady zaprojektowano algorytmy sterowania wraz ze zrealizowaną pamięcią reprezentowane siecią Grafpol GS (rys. 9.49). Na podstawie tej sieci dokonano syntezy równania schematowego (9.11). Implementację równania schematowego dla realizacji za pomocą elementów stykowo-przekaźnikowych, pneumatycznych i sterownika PLC przestawiono kolejno na rys. 9.50, 9.51 i 9.52.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

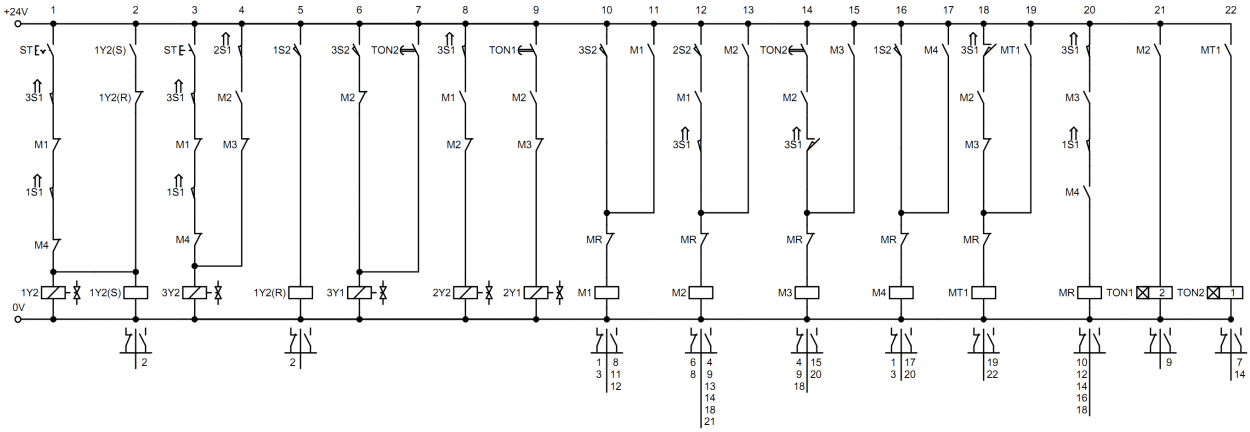


Rys. 9.48. Algorytm procesu przykładu 2 procedur współbieżnych z etapami czasowymi

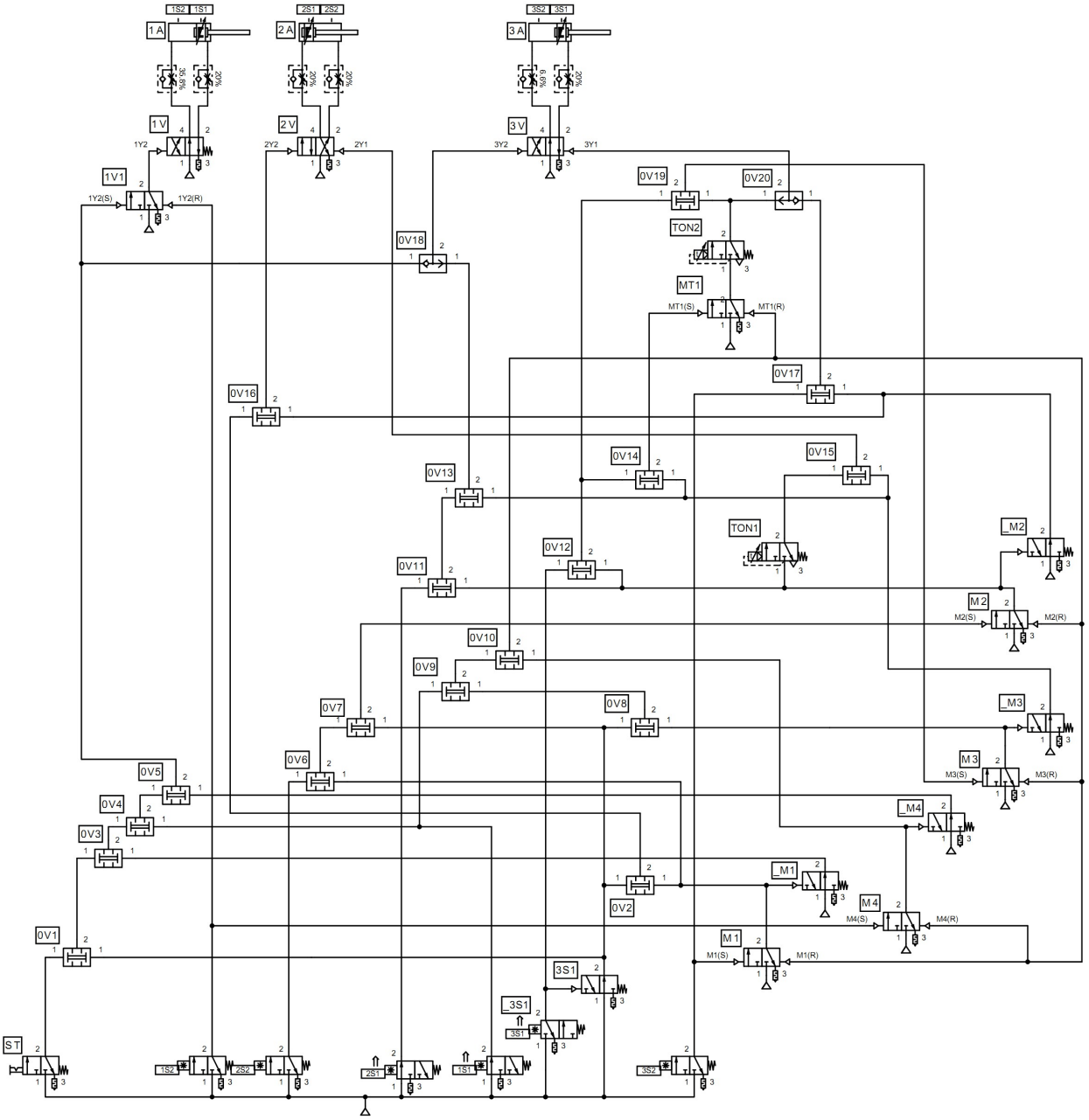


Rys. 9.49. Algorytm sterowania wraz ze zrealizowaną pamięcią przykładu 2 procedur współbieżnych z etapami czasowymi

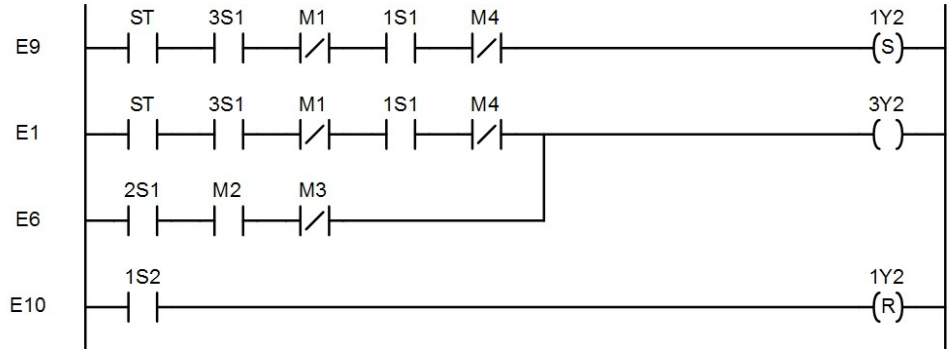
$$
F(Y,M)=\sum
\begin{array}{l}
ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_1^{(1)}(S),\\
\left.\begin{array}{l} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_{3\_1}^{(1)} \\ 2S1\cdot m_2\cdot\overline{m_3}=Y_{3\_2}^{(1)} \end{array}\right\} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}+2S1\cdot m_2\cdot\overline{m_3}=Y_3^{(1)},\\
1S2=Y_1^{(1)}(R),\\
\left.\begin{array}{l} 3S2\cdot\overline{m_2}=Y_{3\_1}^{(2)} \\ TON_2Q=Y_{3\_2}^{(2)} \end{array}\right\} 3S2\cdot\overline{m_2}+TON_2Q=Y_3^{(2)},\\
3S1\cdot m_1\cdot\overline{m_2}=Y_2^{(1)},\\
TON_1Q\cdot m_2\cdot\overline{m_3}=Y_2^{(2)},\\
\\
3S2=M_1(S),\\
2S2\cdot m_1\cdot 3S1=M_2(S),\\
TON_2Q\cdot m_2\cdot\overline{3S1}=M_3(S),\\
1S2=M_4(S),\\
\overline{3S1}\cdot m_2\cdot\overline{m_3}=MT_1(S),\\
3S1\cdot m_3\cdot 1S1\cdot m_4=M_1(R)+M_2(R)+M_3(R)+M_4(R)+MT_1(R)\\
\\
m_2=TON_1IN(2\mathrm{s}),\\
mt_1=TON_2IN(1\mathrm{s}),
\end{array}
\tag{9.11}
$$

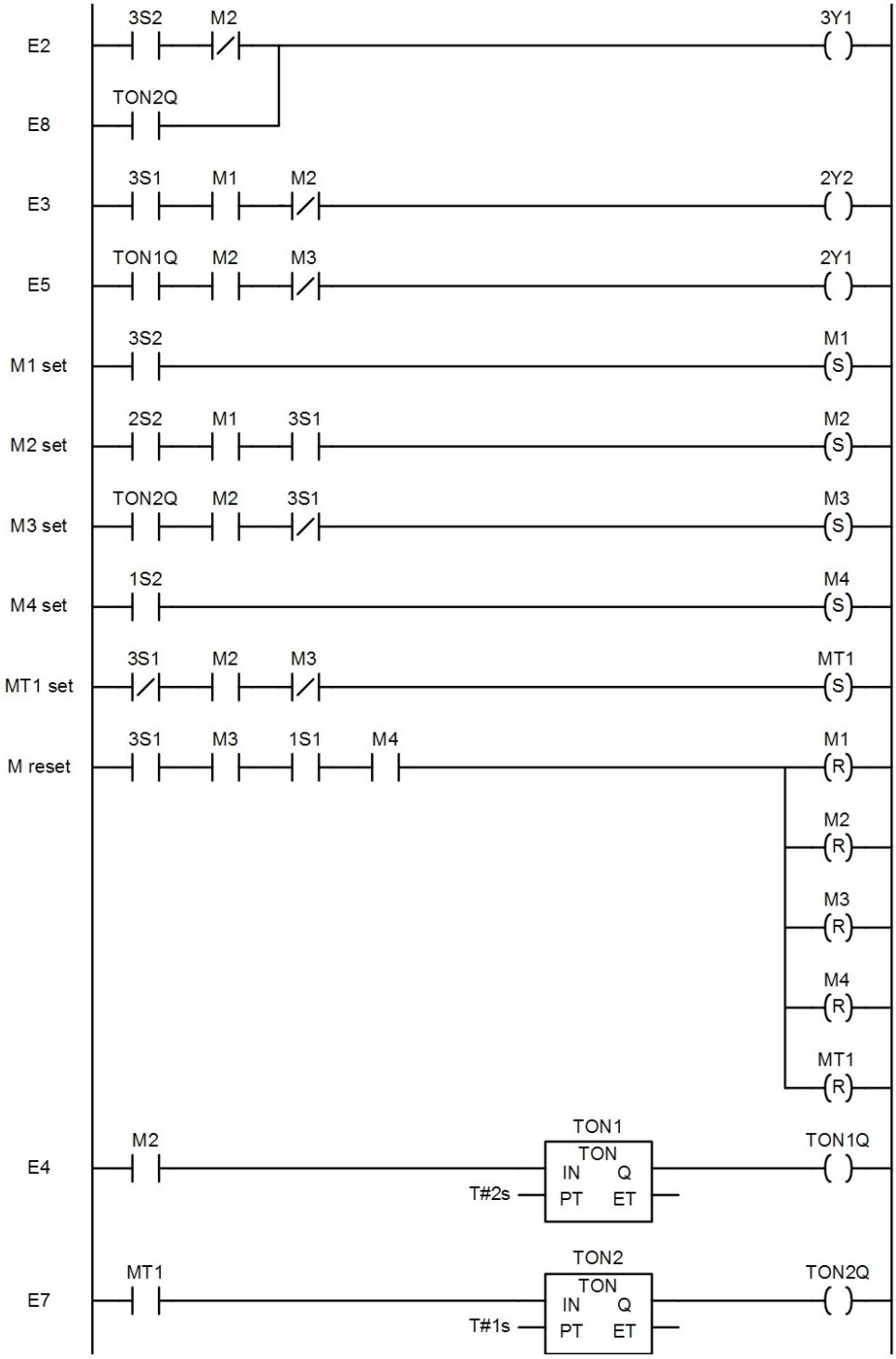


Rys. 9.50. Schemat układu stykowo-przekaźnikowego wyznaczony na podstawie równania schematowego (9.11)



Rys. 9.51. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.11)





Rys. 9.52. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.11)

9.4.3. Przykład 3

Schemat funkcjonalny napędów zaprezentowano na rys. 9.27. Algorytm procesu stanowią dwie procedury sekwencyjne (PS1 i PS2) realizowane równocześnie o opisie słownym:

Procedura sekwencyjna 1 (PS1)
ETAP E1: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E2: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

ETAP E3: *wysuw tłoczyska siłownika 2A*
Realizacja: $2A^{(1)}$ (2Y2)
Sygnalizacja: 2S2=1

ETAP E4: *odmierzanie czasu*
Realizacja: $Timer_1$(3s)
Sygnalizacja: TON_1Q=1

ETAP E5: *wsuw tłoczyska siłownika 3A*
Realizacja: $3A^{(1)}$ (3Y2)
Sygnalizacja: 3S2=1

ETAP E6: *odmierzanie czasu*
Realizacja: $Timer_2$(2s)
Sygnalizacja: TON_2Q=1

ETAP E7: *wsuw tłoczyska siłownika 2A*
Realizacja: $2A^{(2)}$ (2Y1)
Sygnalizacja: 2S1=1

ETAP E8: *wysuw tłoczyska siłownika 3A*
Realizacja: $3A^{(2)}$ (3Y1)
Sygnalizacja: 3S1=1

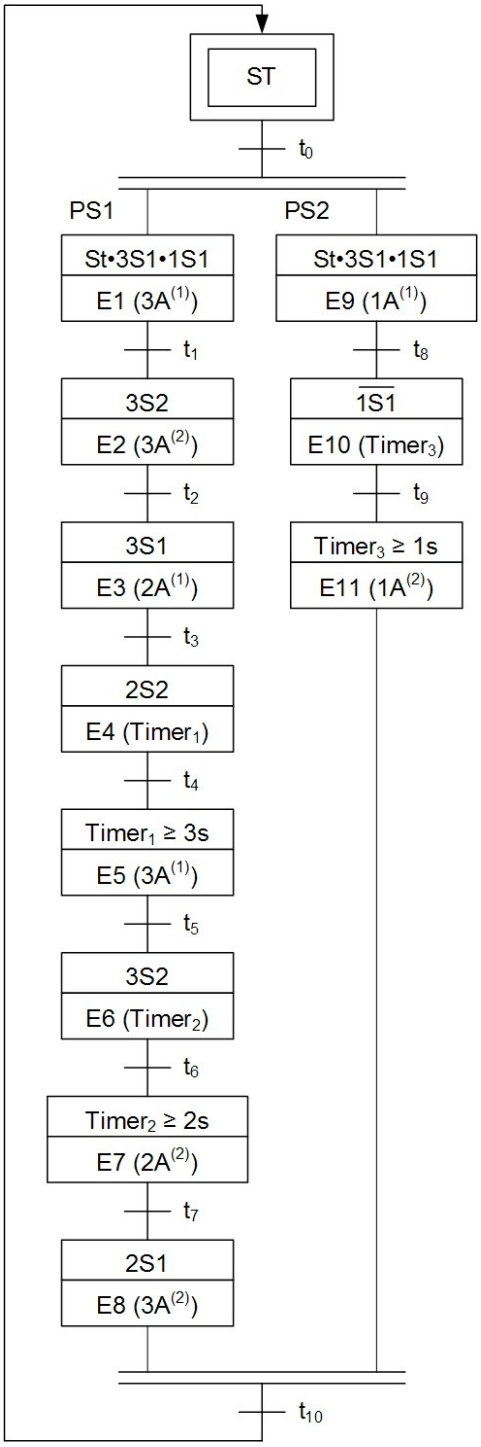
Procedura sekwencyjna 2 (PS2)
ETAP E9: *wsuw tłoczyska siłownika 1A przez zadany czas*
Realizacja: $1A^{(1)}$ ($\overline{1Y2}$)
Sygnalizacja: $\overline{1S1}$=1

ETAP E10: *odmierzanie czasu*
Realizacja: $Timer_3$(1s)
Sygnalizacja: TON_3Q=1

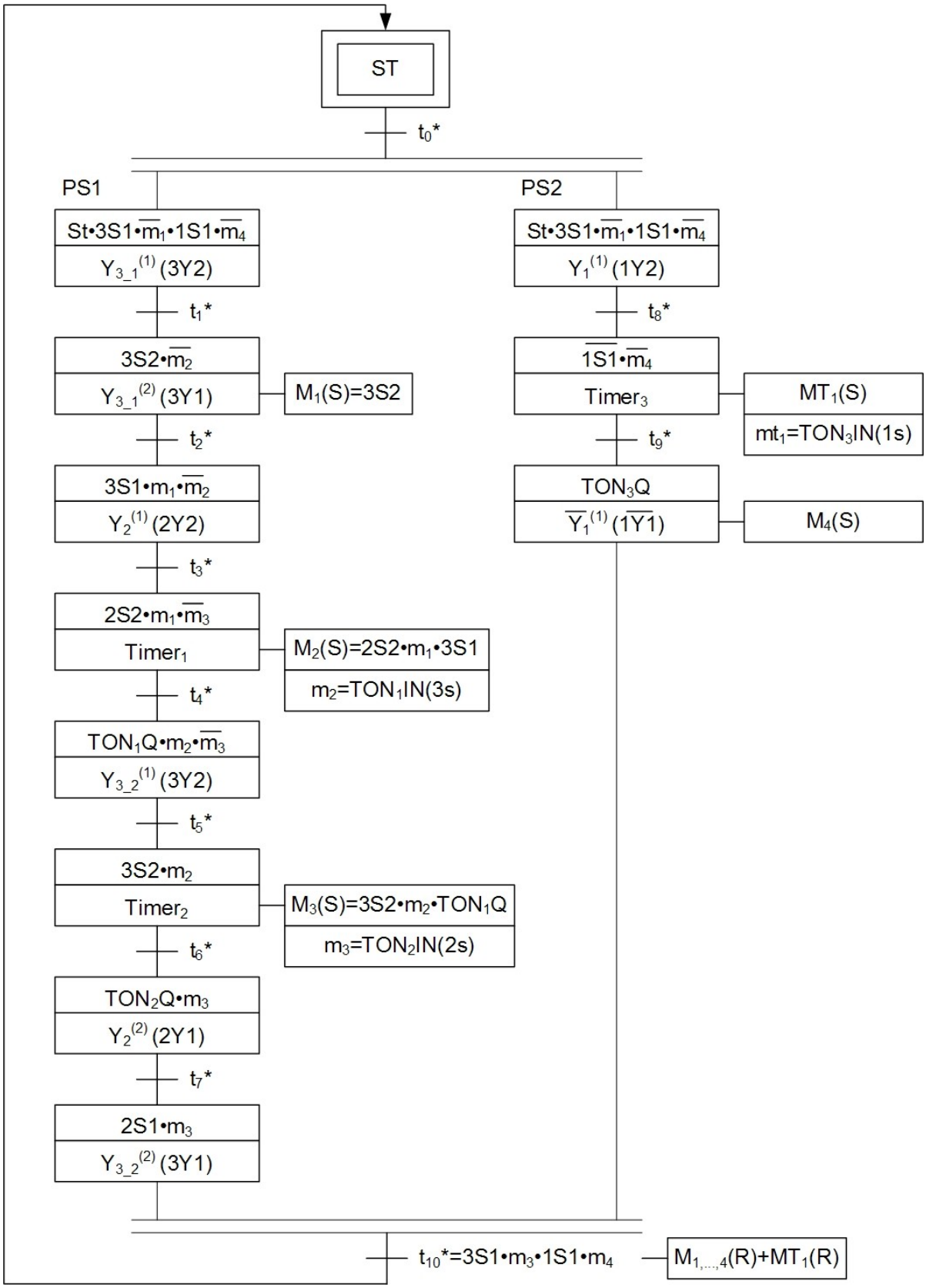
ETAP E11: *wysuw tłoczyska siłownika 1A*
Realizacja: $1A^{(2)}$ ($\overline{1Y2}$)
Sygnalizacja: 1S1=1

Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.53). Stosując opracowane zasady, zaprojektowano algorytmy sterowania wraz ze zrealizowaną pamięcią reprezentowane siecią Grafpol GS (rys. 9.54). Na podstawie tej sieci dokonano syntezy równania schematowego (9.12). Implementację równania schematowego dla realizacji za pomocą elementów stykowo-przekaźnikowch, pneumatycznych oraz sterownika PLC przestawiono kolejno na rys. 9.55, 9.56 i 9.57.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.

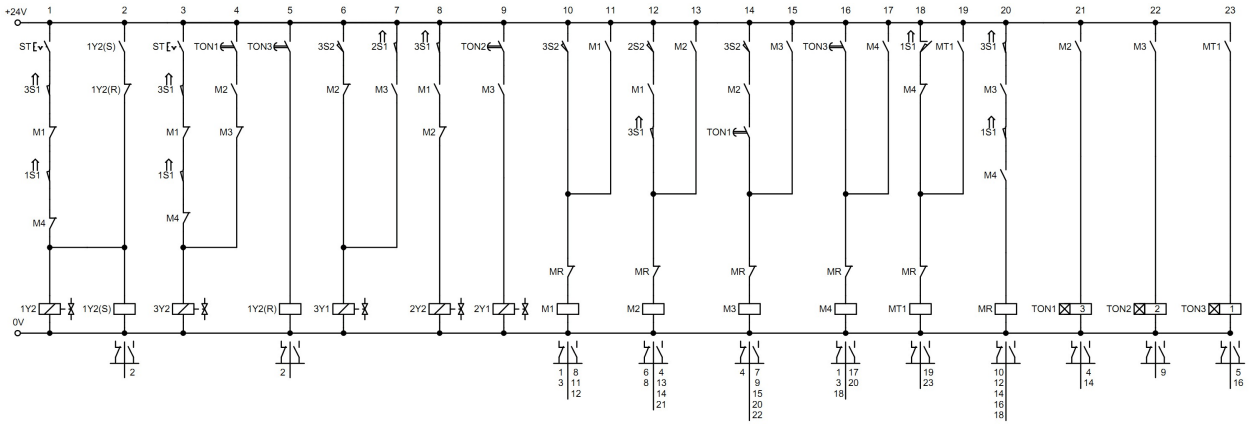


Rys. 9.53. Algorytm procesu przykładu 3 procedur współbieżnych z etapami czasowymi

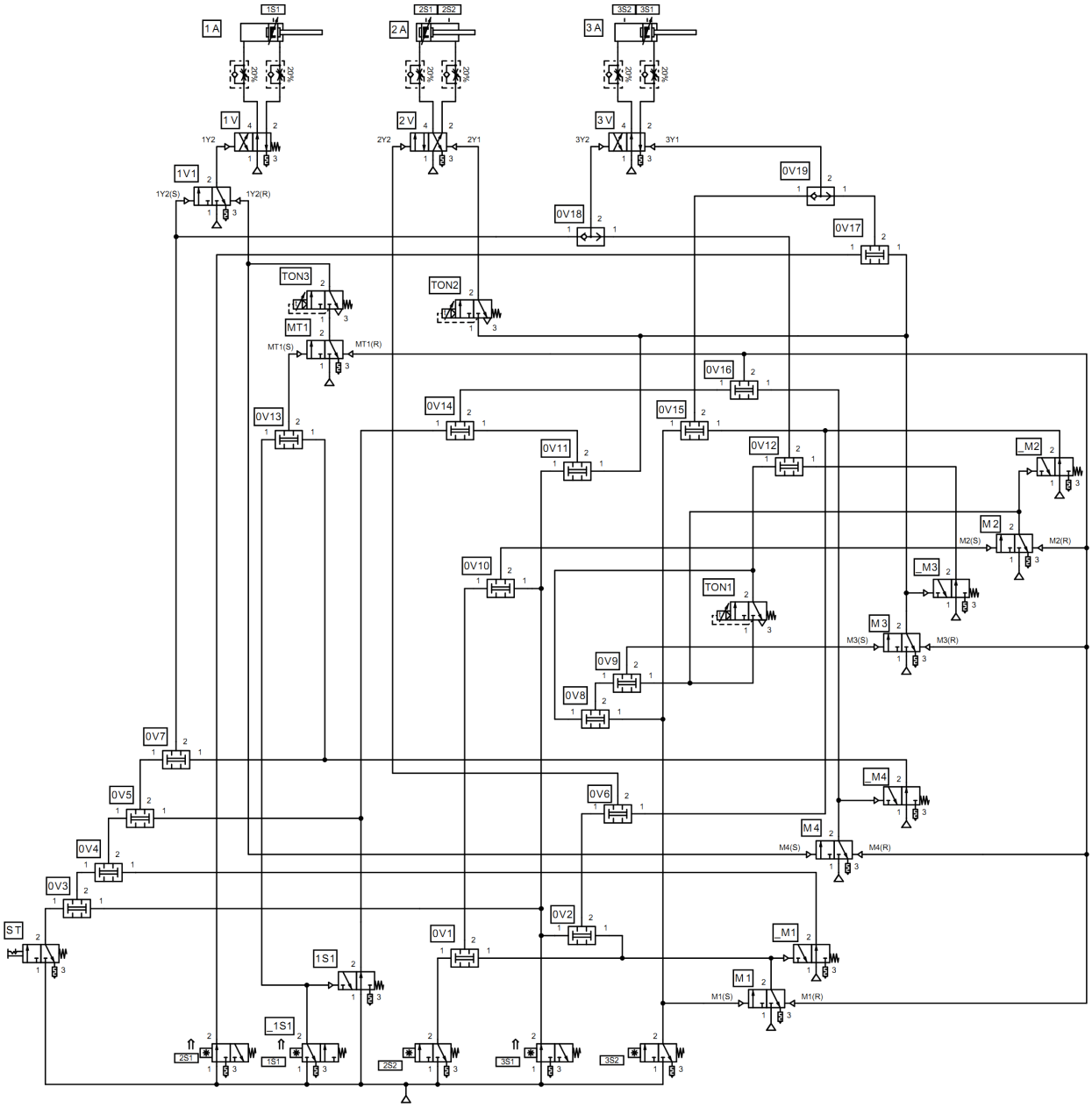


Rys. 9.54. Algorytm sterowania wraz ze zrealizowaną pamięcią przykładu 3 procedur współbieżnych z etapami czasowymi

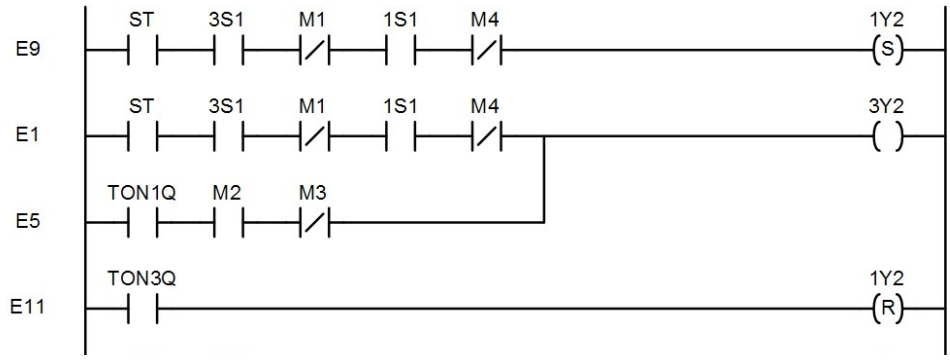
$$
F(Y,M)=\sum
\begin{cases}
ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_1^{(1)}(S),\\
\left.\begin{array}{l} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}=Y_{3\_1}^{(1)}\\ TON_1Q\cdot m_2\cdot\overline{m_3}=Y_{3\_2}^{(1)}\end{array}\right\} ST\cdot 3S1\cdot\overline{m_1}\cdot 1S1\cdot\overline{m_4}+TON_1Q\cdot m_2\cdot\overline{m_3}=Y_3^{(1)},\\
TON_3Q=Y_1^{(1)}(R),\\
\left.\begin{array}{l} 3S2\cdot\overline{m_2}=Y_{3\_1}^{(2)}\\ 2S1\cdot m_3=Y_{3\_2}^{(2)}\end{array}\right\} 3S2\cdot\overline{m_2}+2S1\cdot m_3=Y_3^{(2)},\\
3S1\cdot m_1\cdot\overline{m_2}=Y_2^{(1)},\\
TON_2Q\cdot m_3=Y_2^{(2)},\\
\\
3S2=M_1(S),\\
2S2\cdot m_1\cdot 3S1=M_2(S),\\
3S2\cdot m_2\cdot TON_1Q=M_3(S),\\
TON_3Q=M_4(S),\\
\overline{1S1}\cdot\overline{m_4}=MT_1(S),\\
3S1\cdot m_3\cdot 1S1\cdot m_4=M_1(R)+M_2(R)+M_3+M_4(R)+MT_1(R)\\
\\
m_2=TON_1\,IN\,(3s),\\
m_3=TON_2\,IN\,(2s),\\
mt_1=TON_3\,IN\,(1s),
\end{cases}
\tag{9.12}
$$

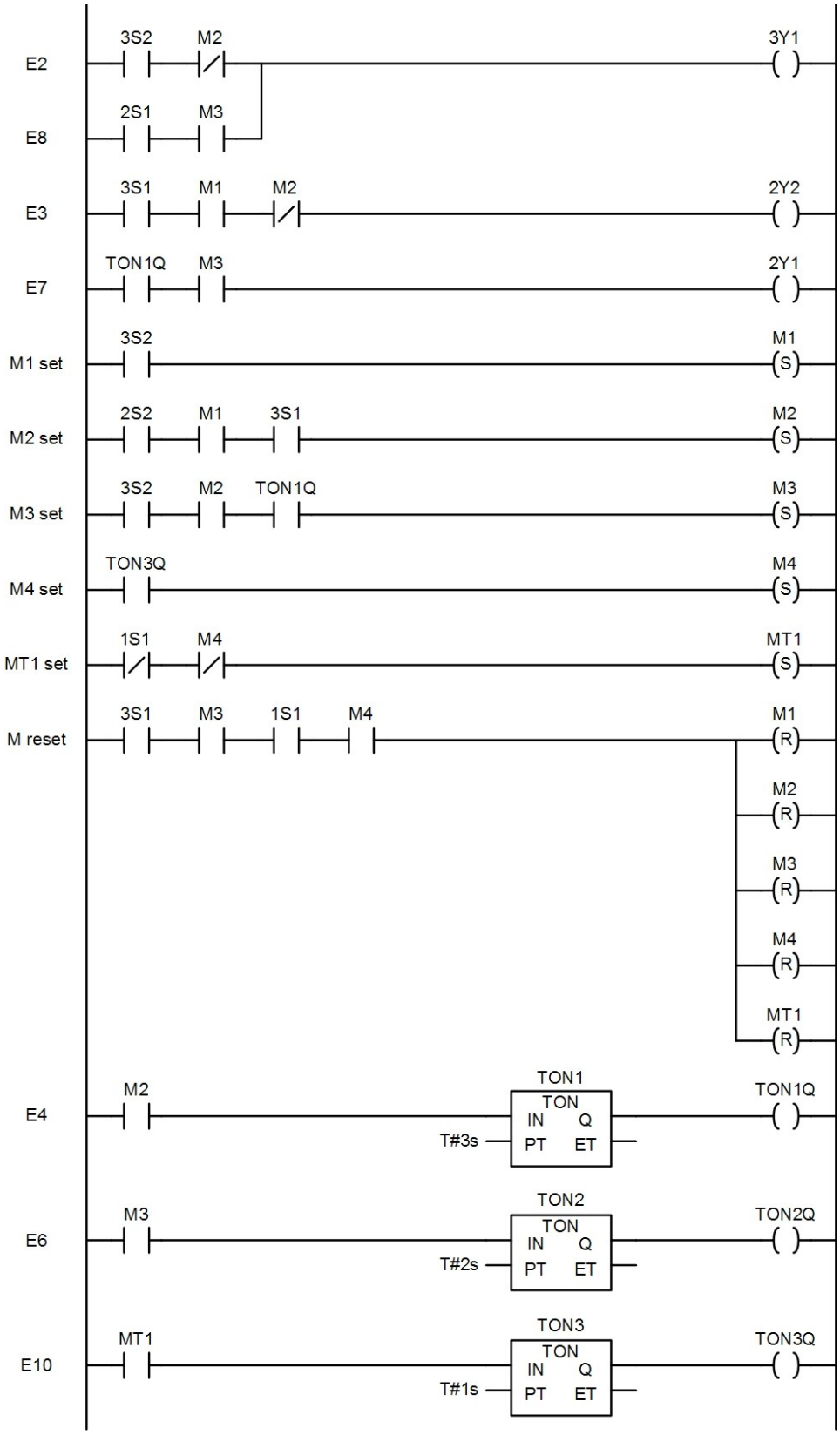


Rys. 9.55. Schemat układu stykowo-przekaźnikowego wyznaczony na podstawie równania schematowego (9.12)



Rys. 9.56. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.12)



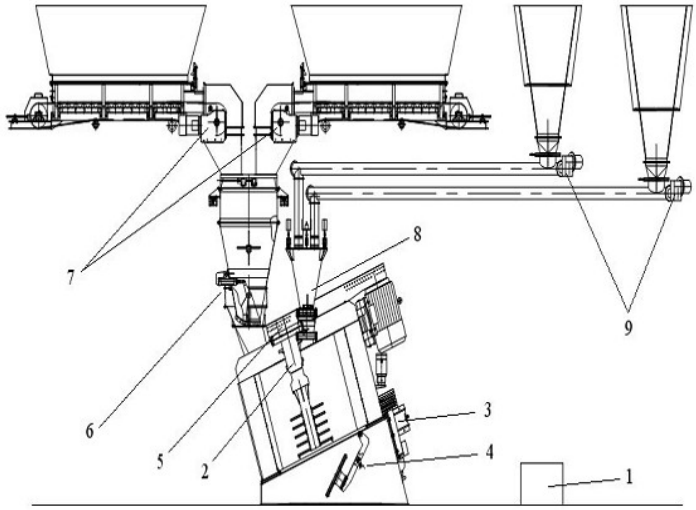


Rys. 9.57. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.12)

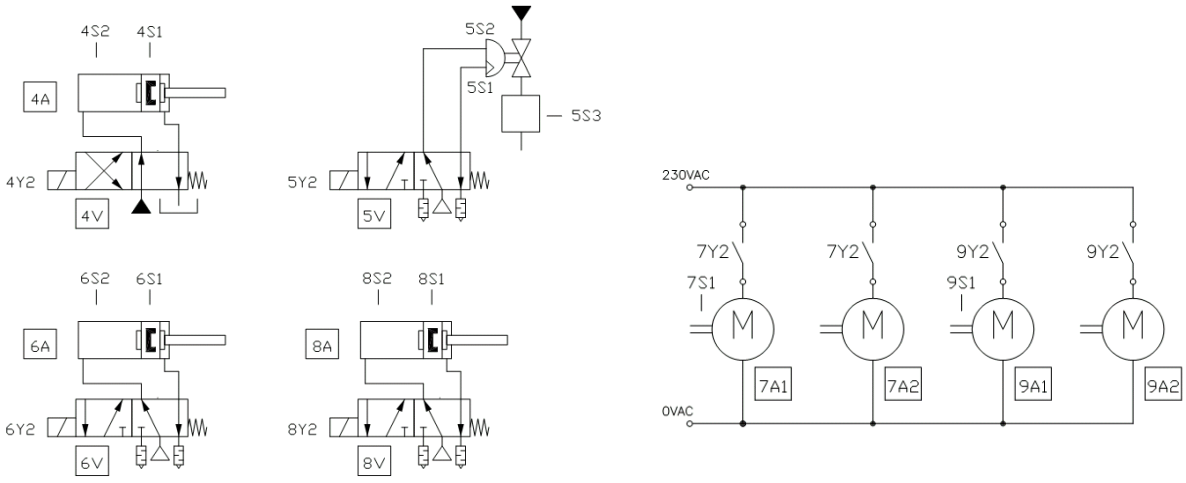
9.5. Procedury złożone

9.5.1. Przykład 1

Schemat funkcjonalny mieszarki turbinowej zaprezentowano na rys. 9.58. Na rysunku 9.59 przedstawiono sposób sterowania i sygnalizacji stanu poszczególnych elementów wykonawczych. Ze względu na ciągłą pracę, od momentu włączenia zasilania mieszarki, elementy oznaczone 1, 2 i 3 nie są brane pod uwagę na etapie modelowania procesu. Na schemacie nie przedstawiono czujników 6S3 i 8S3, które informują o zważeniu odpowiedniej ilości dozowanej do mieszarki masy i dodatków.



Rys. 9.58. Rozmieszczenie poszczególnych elementów wykonawczych na schemacie funkcjonalnym mieszarki turbinowej: 1 - stacja hydrauliczna, 2 - turbina z napędem, 3 - misa z napędem, 4 - klapa wysypowa, 5 - dozownik wody, 6 - waga tensometryczna masy, 7 - dozownik masy do wagi, 8 - waga tensometryczna dodatków, 9 - dozownik dodatków do wagi



Rys. 9.59. Sposób sterowania i sygnalizacji stanu poszczególnych elementów wykonawczych: 5S3 - sygnał dwustanowy z przepływomierza informujący o przepływie zadanej ilości cieczy, 7S1 i 9S1 - sygnały z czujników informujące o zatrzymaniu silników 7A1 i 7A2 oraz 9A1 i 9A2

Algorytm procesu stanowi osiem procedur sekwencyjnych (PS-PS8) o opisie słownym:

Procedura sekwencyjna 1 (PS1)
ETAP E1: *rozpoczęcie dozowania masy do wagi*
Realizacja: $7A^{(1)}$ (7Y2)
Sygnalizacja: 6S3=1

ETAP E2: *zakończenie dozowania masy do wagi*
Realizacja: $7A^{(2)}$ ($\overline{7Y2}$)
Sygnalizacja: 7S1=1

Procedura sekwencyjna 2 (PS2)
ETAP E3: *rozpoczęcie dozowania dodatków do wagi*
Realizacja: $9A^{(1)}$ (9Y2)
Sygnalizacja: 8S3=1

ETAP E4: *zakończenie dozowania dodatków do wagi*
Realizacja: $9A^{(2)}$ ($9\overline{Y2}$)
Sygnalizacja: 9S1=1

Procedura sekwencyjna 3 (PS3)
ETAP E5: *otwarcie (opróżnienie) wagi tensometrycznej masy*
Realizacja: $6A^{(1)}$ (6Y2)
Sygnalizacja: 6S2=1

Procedura sekwencyjna 4 (PS4)
ETAP E6: *otwarcie zaworu dozowania wody*
Realizacja: $5A^{(1)}$ (5Y2)
Sygnalizacja: 5S2=1

ETAP E7: *dozowanie wody*
Realizacja: brak
Sygnalizacja: 5S3=1

ETAP E8: *zamknięcie zaworu dozowania wody*
Realizacja: $5A^{(2)}$ ($\overline{5Y2}$)
Sygnalizacja: 5S1=1

Procedura sekwencyjna 5 (PS5)
ETAP E9: *otwarcie (opróżnienie) wagi tensometrycznej dodatków*
Realizacja: $8A^{(1)}$ (8Y2)
Sygnalizacja: 8S2=1

ETAP E10: *odmierzanie czasu*
Realizacja: $Timer_1$(4s)
Sygnalizacja: TON_1Q=1

Procedura sekwencyjna 6 (PS6)
ETAP E11: *zamknięcie wagi tensometrycznej masy*
Realizacja: $6A^{(2)}$ ($\overline{6Y2}$)
Sygnalizacja: 6S1=1

Procedura sekwencyjna 7 (PS7)
ETAP E12: *zamknięcie wagi tensometrycznej dodatków*
Realizacja: $8A^{(2)}$ ($\overline{8Y2}$)
Sygnalizacja: 8S1=1

Procedura sekwencyjna 8 (PS8)
ETAP E13: *odmierzanie czasu*
Realizacja: $Timer_2(28s)$
Sygnalizacja: $TON_2Q=1$

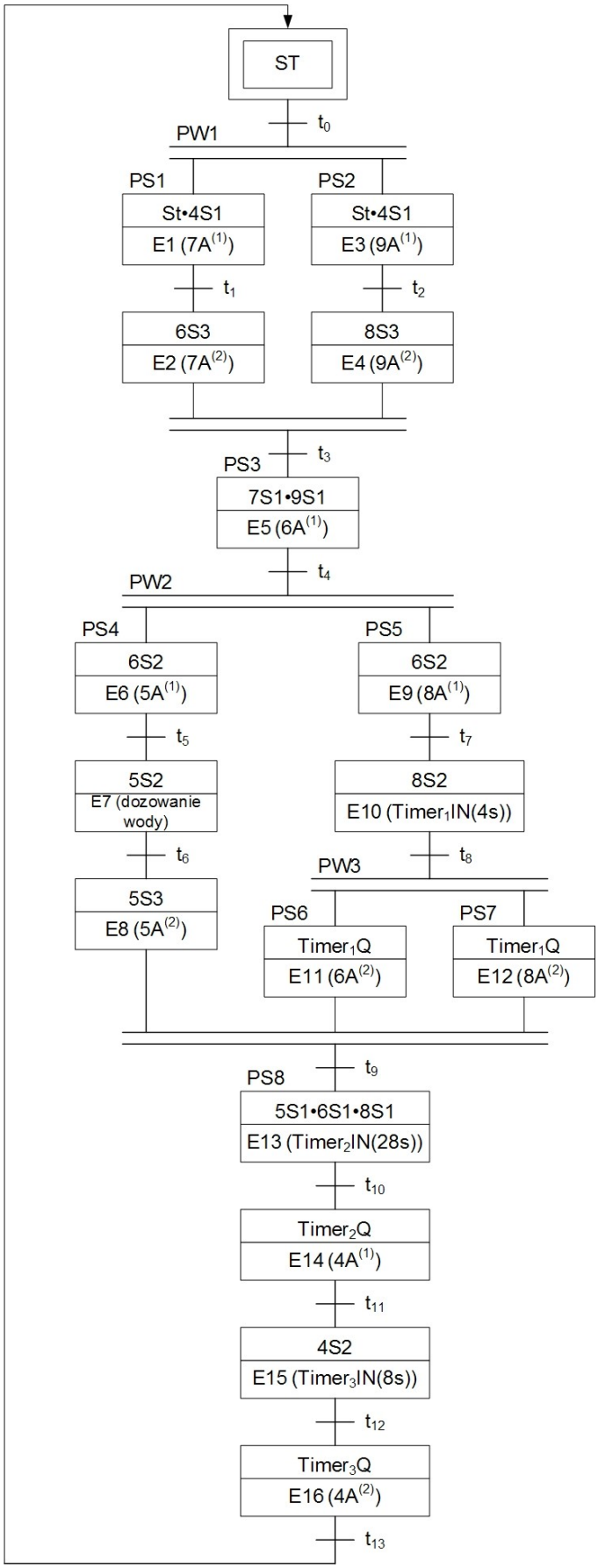
ETAP E14: *otwarcie klapy wysypowej (opróżnienie mieszarki)*
Realizacja: $4A^{(1)}$ (4Y2)
Sygnalizacja: 4S2=1

ETAP E15: *odmierzanie czasu*
Realizacja: $Timer_3(8s)$
Sygnalizacja: $TON_3Q=1$

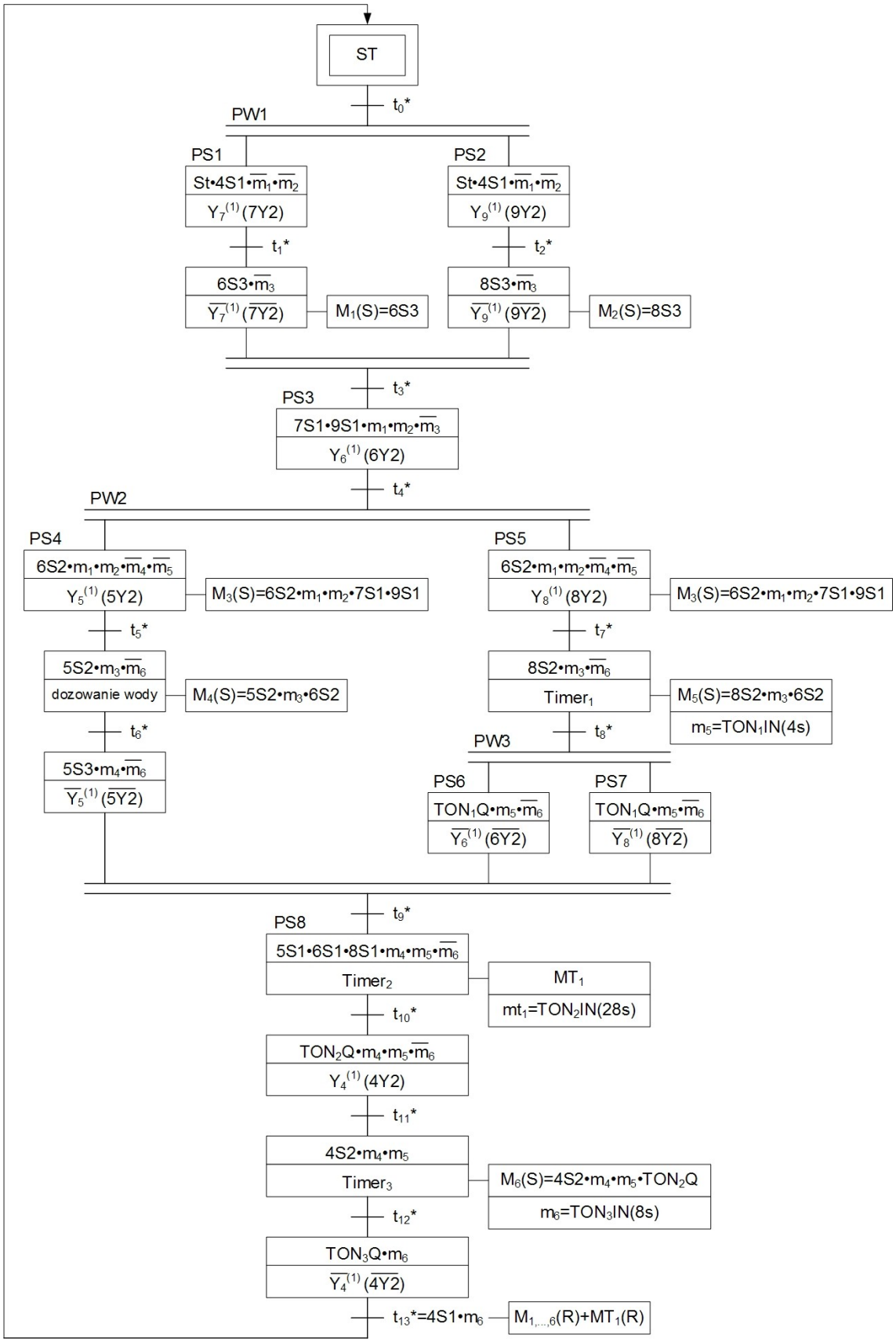
ETAP E16: *zamknięcie klapy wysypowej*
Realizacja: 4A(2) (4Y2)
Sygnalizacja: 4S1=1

Przedstawiony opis słowny pracy napędów zapisano w postaci sieci Grafpol GP (rys. 9.60). Stosując opracowane zasady zaprojektowano algorytm sterowania wraz ze zrealizowaną pamięcią reprezentowany siecią Grafpol GS (rys. 9.61). Na podstawie tej sieci dokonano syntezy równania schematowego (9.13). Implementację równania schematowego dla realizacji za pomocą sterownika PLC, układu stykowo-przekaźnikowego oraz elementów pneumatycznych przestawiono kolejno na rys. 9.62, 9.63 i 9.64.

W wyniku przeprowadzonych badań stwierdzono poprawność opracowanego algorytmu sterowania względem zakładanego algorytmu procesu.



Rys. 9.60. Algorytm procesu przykładu 1 procedur złożonych

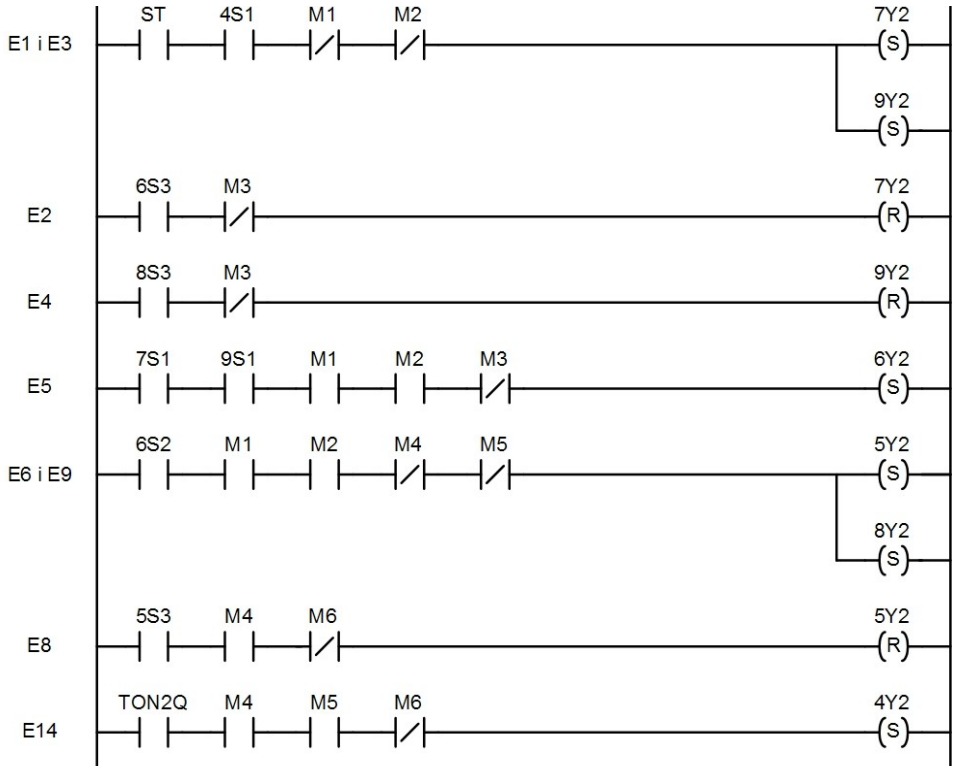


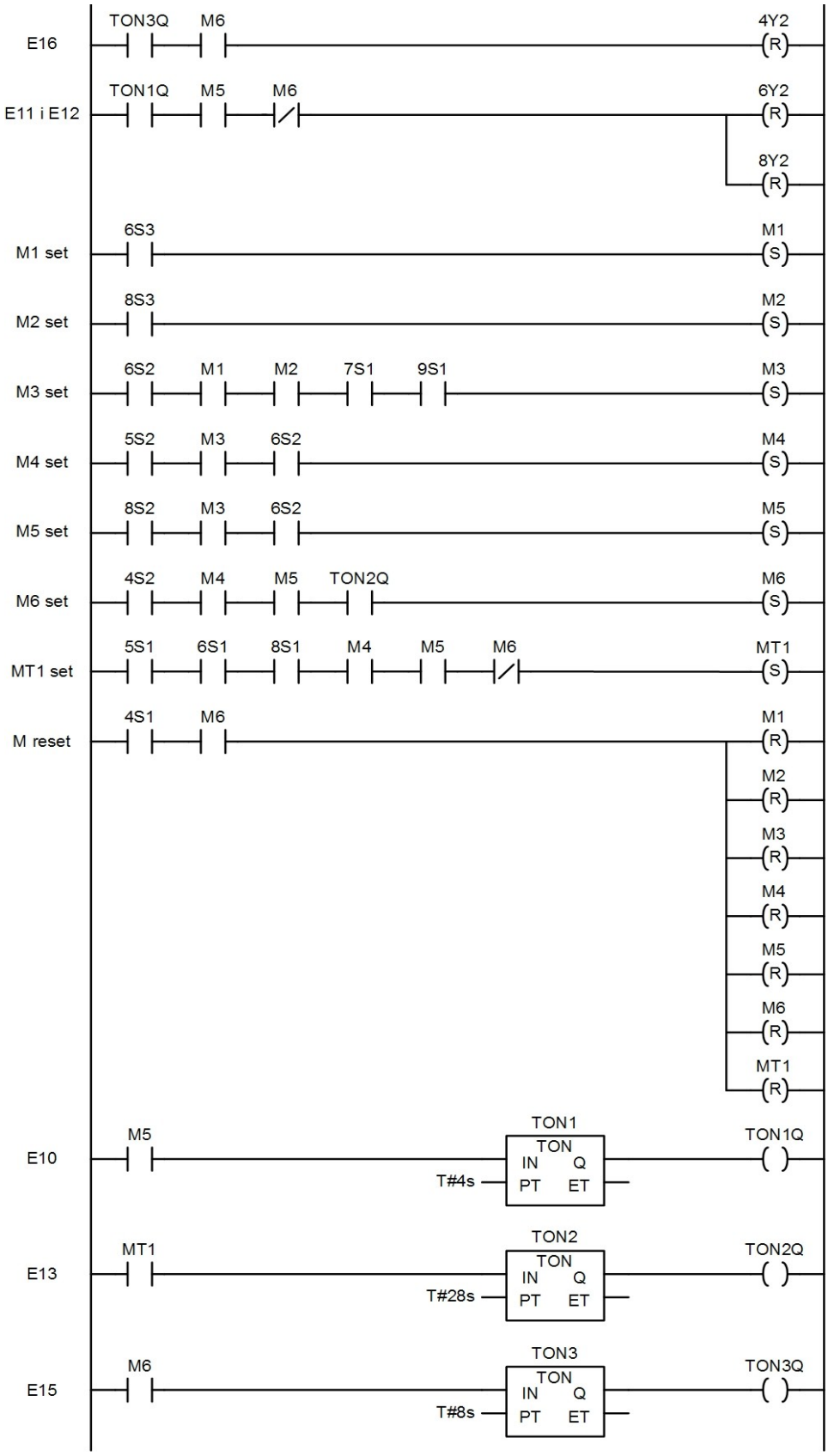
Rys. 9.61. Algorytm sterowania przykładu 1 procedur złożonych

$$
\begin{aligned}
& ST \cdot 4\mathrm{S1} \cdot \overline{m_1} \cdot \overline{m_2} = Y_7^{(1)}(S) + Y_9^{(1)}(S), \\
& 6\mathrm{S3} \cdot \overline{m_3} = Y_7^{(1)}(R), \\
& 8\mathrm{S3} \cdot \overline{m_3} = Y_9^{(1)}(R), \\
& 7\mathrm{S1} \cdot 9\mathrm{S1} \cdot m_1 \cdot m_2 \cdot \overline{m_3} = Y_6^{(1)}(S), \\
& 6\mathrm{S2} \cdot m_1 \cdot m_2 \cdot \overline{m_4} \cdot \overline{m_5} = Y_5^{(1)}(S) + Y_8^{(1)}(S), \\
& 5\mathrm{S3} \cdot m_4 \cdot \overline{m_6} = Y_5^{(1)}(R), \\
& TON_1 Q \cdot m_5 \cdot \overline{m_6} = Y_6^{(1)}(R) + Y_8^{(1)}(R), \\
& TON_2 Q \cdot m_4 \cdot m_5 \cdot \overline{m_6} = Y_4^{(1)}(S), \\
& TON_3 Q \cdot m_6 = Y_4^{(1)}(R),
\end{aligned}
$$

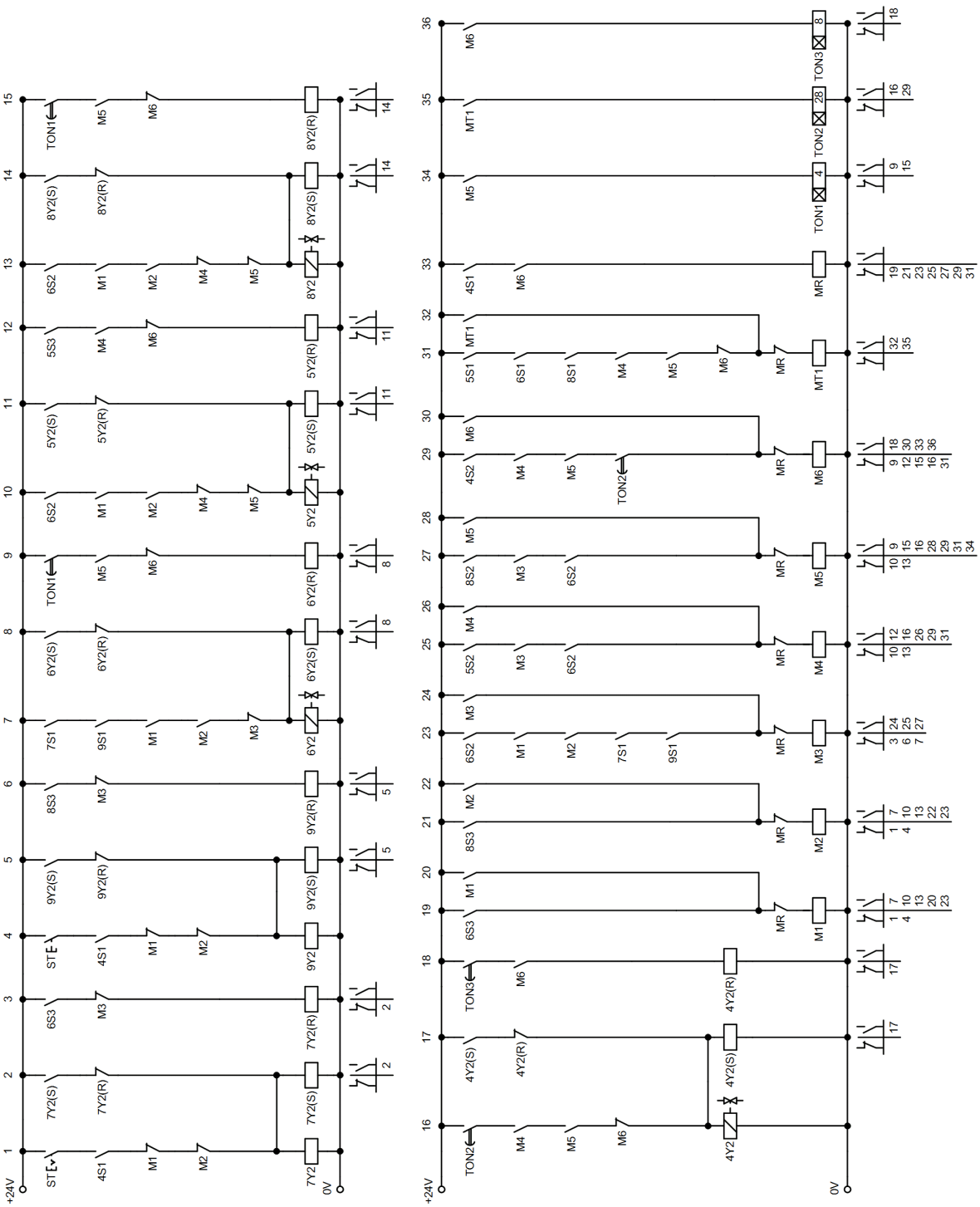
$$
F(Y, M) = \sum
\begin{aligned}
& 6\mathrm{S3} = M_1(S), \\
& 8\mathrm{S3} = M_2(S), \\
& 6\mathrm{S2} \cdot m_1 \cdot m_2 \cdot 7\mathrm{S1} \cdot 9\mathrm{S1} = M_3(S), \\
& 5\mathrm{S2} \cdot m_3 \cdot 6\mathrm{S2} = M_4(S), \\
& 8\mathrm{S2} \cdot m_3 \cdot 6\mathrm{S2} = M_5(S), \\
& 4\mathrm{S2} \cdot m_4 \cdot m_5 \cdot TON_2 Q = M_6(S), \\
& 5\mathrm{S1} \cdot 6\mathrm{S1} \cdot 8\mathrm{S1} \cdot m_4 \cdot m_5 \cdot \overline{m_6} = MT_1(S), \\
& 4\mathrm{S1} \cdot m_6 = M_1(R) + M_2(R) + M_3(R) + M_4(R) + M_5(R) + M_6(R) + MT_1(R)
\end{aligned}
\tag{9.13}
$$

$$
\begin{aligned}
& m_5 = TON_1\, IN\,(4\mathrm{s}), \\
& mt_1 = TON_2\, IN\,(28\mathrm{s}), \\
& m_6 = TON_3\, IN\,(8\mathrm{s}),
\end{aligned}
$$

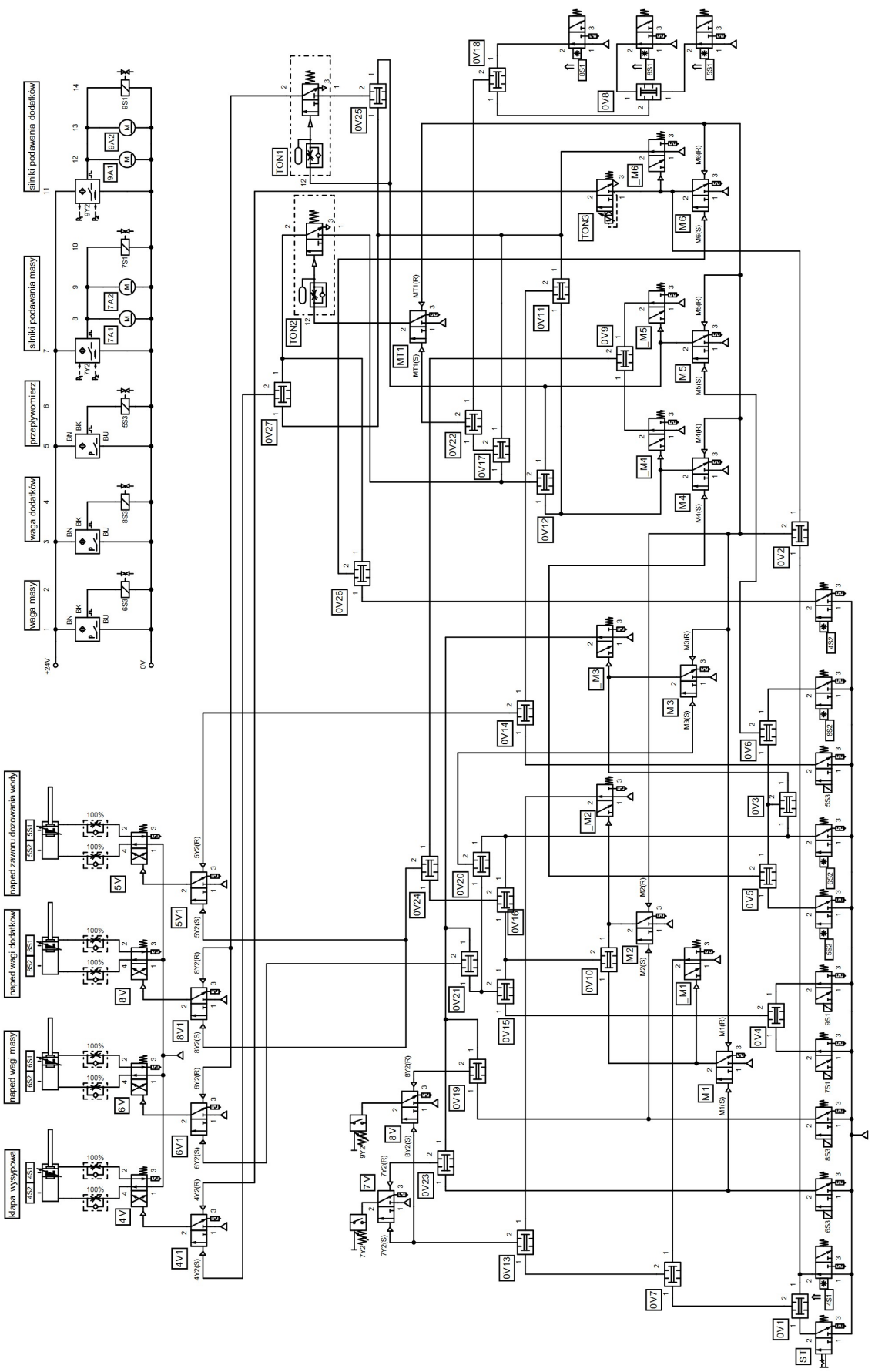




Rys. 9.62. Program sterownika PLC, zapisany za pomocą języka LD, wyznaczony na podstawie równania schematowego (9.13)



Rys. 9.63. Schemat układu stykowo-przekaźnikowego wyznaczony na podstawie równania schematowego (9.13)



Rys. 9.64. Schemat układu sterowania, realizowanego za pomocą elementów pneumatycznych, wyznaczony na podstawie równania schematowego (9.13)

10. PODSUMOWANIE I WNIOSKI

W pracy zaprezentowano nowe zasady syntezy metodą Grafpol równań schematowych sekwencyjnych układów sterowania. Opracowane zasady pozwalają na projektowanie algorytmów sterowania procedur sekwencyjnych, współbieżnych oraz zawierających etapy czasowe. Do zalet opracowanych nowych zasad syntezy sekwencyjnych układów sterowania metodą Grafpol należy zaliczyć:

- Przejrzystość stosowanych na schemacie funkcjonalnym oraz sieciach Grafpol GP i GS oznaczeń dzięki zastosowaniu symboli i ich systematyki zdefiniowanej w normie PN-ISO 1219-2. Przekłada się to bezpośrednio na skrócenie czasu syntezy rozwiązania, gdyż oznaczenia elementów sygnalizacyjnych i wykonawczych bezpośrednio definiują ich funkcje. Zmniejsza to zdecydowanie, a wręcz eliminuje, konieczność wielokrotnego powrotu i interpretacji schematu funkcjonalnego oraz opisu słownego w trakcie syntezy równania schematowego.
- Brak konieczności analizy przebiegu sygnałów wejściowych i wyjściowych układu sterowania, w celu syntezy równania schematowego, w porównaniu z dotychczas stosowanymi w metodzie Grafpol zasadami używanymi w Metodzie Transformacji Sieci (MTS).
- Ułatwiona w porównaniu z metodą MTS analiza równań schematowych. W opracowanym rozwiązaniu sekwencyjna realizacja etapów procesu zapewnione jest przez odpowiedni zapis komórek pamięci układu sterowania i właściwe stosowanie ich sygnałów w tranzycjach t_i. Nie są stosowane pętle sprzężeń zwrotnych niezbędne w układach sterowania projektowanych metodą MTS. Projektowane według nowych zasad równanie schematowe ściśle określa warunki zapisu komórek pamięci oraz użycia ich sygnałów w tranzycjach t_i. Dzięki temu po zapisie danej komórki pamięci od razu wiadomo, które zmienne sterujące Y_i mogą przyjąć wartość logiczną '1' , jeśli spełniona będzie tranzycja t_i, a które nie, pomimo spełnienia tranzycji t_i. Innymi słowy komórki pamięci zezwalają lub zabraniają na przypisanie wartości logicznej '1' zmiennym sterującym Y_i.
- Stosowanie zdecydowanie mniejszej liczby elementarnych komórek pamięci w porównaniu z algorytmami projektowanymi metodą Grafcet lub zapisywanymi językiem programowania SFC. W przeciwieństwie do Grafcet i SFC, gdzie komórki pamięci przyporządkowane są do każdego z kroków algorytmu sterowania, w opracowanym rozwiązaniu komórki pamięci przechowują informacje o zakończeniu wykonania ruchu danego napędu z pozycji wyjściowej. Dzięki temu ich liczba może być mniejsza nawet o 50% w stosunku do algorytmów projektowanych metodą Grafcet i SFC.
- Projektowane, według nowych zasad metodą Grafpol, równanie schematowe może być zaimplementowane w układach sterowania realizowanych przez elementy pneumatyczne, stykowo-przekaźnikowe oraz sterowniki PLC. Istotną cechą opracowanego równania schematowego, w odniesieniu do implementacji w sterowniku PLC, jest jego uniwersalność. Przejawia się ona w możliwości zapisu programu użytkowego za pomocą jednego dowolnego języka programowania zdefiniowanego w normie PN-EN 61131-3.
- Prostota i szybkość modyfikacji istniejącego równania schematowego w przypadku zmiany elementu sterującego, np. zaworu bistabilnego na zawór monostabilny, względem pierwotnego schematu funkcjonalnego.

Metoda Grafpol, której dotyczą nowo opracowane zasady syntezy równań schematowych, składa się z pięciu etapów.

- Etap I - w tym etapie realizowany jest schemat funkcjonalny, który przedstawia proces w stanie początkowym (wyjściowym) oraz zawiera wszystkie elementy wykonawcze

i sygnalizacyjne. Etap ten zawiera również opis słowny algorytmu modelowanego procesu.

- Etap II - na podstawie schematu funkcjonalnego i opisu słownego modelowany jest algorytm procesu, stanowiący graficzne ujęcie przebiegu etapów elementarnych procesu. W metodzie Grafpol algorytm procesu jest reprezentowany przy pomocy sieci Grafpol GP. Sieć ta stanowi podstawę do budowy algorytmu sterowania w etapie trzecim.
- Etap III - algorytm sterowania, reprezentowany siecią Grafpol GS, otrzymuje się poprzez odwzorowanie etapów algorytmu procesu krokami algorytmu sterowania w których występują zmienne sterujące Y_i.
- Etap IV - na podstawie sieci Grafpol GS następuje realizacja pamięci wg opracowanych nowych zasad. Realizacja pamięci dotyczy trzech aspektów - zapisu elementarnych komórek pamięci, użycia informacji przechowywanych w elementarnych komórkach pamięci w algorytmie sterowania oraz kasowania elementarnych komórek pamięci. Zapis elementarnych komórek pamięci następuje zawsze po zakończeniu wykonania pierwszego ruchu danego elementu wykonawczego z pozycji początkowej. Ruch taki następuje pod wpływem sygnału wyjściowego $Y_i^{(1)}$. Zakończenie wykonania ruchu sygnalizowane jest przez warunek reprezentowany tranzycją t_i. Znając stany układu sterowania w których powinien następować zapis elementarnych komórek pamięci, na podstawie opracowanych nowych zasad, możliwe jest wyznaczenie zależności opisujących warunki zapisu poszczególnych komórek pamięci. Kolejnym krokiem jest zastosowanie informacji przechowywanych w elementarnych komórkach pamięci w algorytmie sterowania. Ponieważ na tym etapie określono już tranzycje w których następuje zapis elementarnych komórek pamięci opracowane nowe zasady umożliwiają określenie postaci tranzycji t_i^*. Tranzycje t_i^* to tranzycje z uwzględnionymi informacjami przechowywanymi w elementarnych komórkach pamięci. Tak stworzony algorytm sterowania z uwzględnionymi komórkami pamięci jest prawie kompletny. Ze względu jednak na fakt iż procedury sekwencyjne realizowane są cyklicznie, konieczne jest kasowanie elementarnych komórek pamięci po zakończeniu realizacji procesu. Kasowanie elementarnych komórek pamięci następuje zawsze w ostatnim stanie algorytmu sterowania. Tranzycja t_n^*, będąca ostatnią tranzycją algorytmu sterowania, określa warunki kasowania wszystkich elementarnych komórek pamięci. Powyżej prezentowany sposób postępowania, oparty na opracowanych nowych zasadach realizacji pamięci metodą Grafpol sekwencyjnych układów sterowania, znajduje zastosowanie zarówno w procedurach sekwencyjnych, jak również współbieżnych i złożonych. W przypadku występowania w algorytmie etapów czasowych konieczne jest zastosowanie liczników czasu typu TON wg PN-EN 61131-3. Liczniki czasu, w zależności od położenia etapu czasowego względem etapów w których następuje ruch danego napędu z pozycji początkowej, inicjowane są przez wspomniane powyżej sygnały elementarnych komórek pamięci, nazywane również sterującymi, lub przez dodatkowe komórki pamięci nazywane skojarzonymi. Szczegółowy sposób inicjowania liczników czasu oraz zastosowania ich sygnału wyjściowego przedstawiono w rozdziale 8.4.
- Etap V - implementacja opracowanego równania schematowego w układzie sterowania. Ponieważ postawiono tezę, iż równanie schematowe stanowi podstawę do realizacji dowolnych sekwencyjnych układów sterowania każdorazowo równanie implementowano w trzech typach układów sterowania:
  - realizowanym przez elementy pneumatyczne,
  - stykowo-przekaźnikowym,
  - sterowniku PLC.

Poprawność opracowanych nowych zasad realizacji pamięci i syntezy równania

schematowego metodą Grafpol zweryfikowano na podstawie badań symulacyjnych. Badania te polegały na:

- projektowaniu równań schematowych przykładowych procedur,
- implementacji opracowanych równań schematowych w układach sterowania sterujących modelowanymi w programie FluidSim napędami,
- porównaniu poprawności pracy modelowanych napędów z wymaganymi algorytmami procesu.

Badania obejmowały swoim zakresem następujące algorytmy:

- sekwencyjne,
- sekwencyjne z etapami czasowymi,
- współbieżne,
- współbieżne z etapami czasowymi,
- złożone.

Wyniki przeprowadzonych badań potwierdziły obie tezy niniejszej pracy.

Podsumowując przeprowadzone prace oraz badania można stwierdzić, że opracowane zasady realizacji pamięci i syntezy równania schematowego metodą Grafpol stanowią rozwiązanie postawionego w pracy celu badawczego. Ponadto można stwierdzić że:

1. Przejrzysty system oznaczeń oraz brak konieczności graficznej analizy przebiegu sygnałów wejściowo-wyjściowych układu sterowania wynikający z nowych zasad realizacji pamięci i syntezy równania schematowego sprawiają, że metoda Grafpol jest prosta w praktycznym stosowaniu.
2. Równania schematowe projektowane metodą Grafpol mogą być implementowane w różnych realizacjach układów sterowania (stykowo-przekaźnikowych, realizowanych za pomocą elementów pneumatycznych, sterowniku PLC).
3. Projektowane równania schematowe można łatwo modyfikować w przypadku zmiany typu elementu sterującego danym napędem.

Na zakończenie istotne jest podkreślenie, że opracowane zasady syntezy programu sterującego metodą Grafpol odnoszą się jedynie do meritum programu sterującego rzeczywistym urządzeniem. Dlatego też kolejnym krokiem, stanowiącym kontynuację wykonanej pracy, mogłoby być rozszerzenie opracowanych zasad o procesy związane z uzyskiwaniem pozycji referencyjnej i diagnostyki bezpośrednio po zasileniu urządzenia, obsługi sytuacji awaryjnych związanych z wciśnięciem przycisku "STOP AWARYJNY", sygnalizacji i obsługi diagnostyki w trakcie pracy urządzenia.

LITERATURA

1 Aumiaux M.: Methodologie d'implantation d'un reseau de Petri ou d'un grafcet dans des circuit logiques programmables, RAIRO Automatique. vol. 23 no 6, Paris 1989.

2 Baier A., Kost G., Świder J., Zdanowicz R.: Sterowanie i automatyzacja procesów technologicznych i układów mechatronicznych, Układy pneumatyczne i elektropneumatyczne ze sterowaniem logicznym (PLC). Wydawnictwo Politechniki Śląskiej, Gliwice 2008

3 Banaszak Z., Drzazga A., Kuś J.: Metody interakcyjnego modelowania i programowania procesów dyskretnych. Wydawnictwo Politechniki Wrocławskiej, Wrocław, 1993

4 Cieniak I.: Polski rynek sterowników programowalnych - Zastój na rynku PLC. Control Engineering Polska, nr 8 (61), październik 2009 r., str. 28-39

5 Cieniak I.: Polski rynek sterowników programowalnych PLC - PLC idzie do przodu. Control Engineering Polska, nr 8 (71), październik 2010 r., str. 36-47

6 Ciskowski S., Sysak Z., Dworzak Ł., Ganczarek M., Mikulczyński T.: Modelowanie i programowanie metodą Grafpol pracy napędów pneumatycznych. Napędy i sterowania hydrauliczne i pneumatyczne '2007. Krajowy sektor wobec wyzwań konkurencyjności. Międzynarodowa konferencja naukowo-techniczna, Wrocław 10-12 października 2007. Ośrodek Doskonalenia Kadr SIMP, str. 56-63

7 Dworzak Ł., Ciskowski S., Mikulczyński T., Bogdan M.: Synteza metodą Grafpol sekwencyjnych algorytmów sterowania. Pomiary, Automatyka, Robotyka. 2009, R. 13, nr 2, str. 686-694

8 Dworzak Ł., Ciskowski S., Mikulczyński T.: Modelowanie i programowanie metodą Grafpol procedur współbieżnych procesów dyskretnych. Pneumatyka, nr 3, 2007 r., str. 24-27

9 Dworzak Ł., Ciskowski S., Mikulczyński T.: Modelowanie i programowanie procedur sekwencyjnych analityczną metodą Grafpol uwzględniającą możliwość wielokrotnych ruchów poszczególnych napędów sterowanych impulsowo. Pneumatyka, nr 1, 2008 r., str. 46-49

10 Dworzak Ł., Ciskowski S., Mikulczyński T.: Nowa metoda modelowania i programowania procedur sekwencyjnych. Pneumatyka, nr 4, 2007 r., str. 12-14

11 Dworzak Ł., Mikulczyński T., Barycki J.: Synteza metodą Grafpol współbieżnych algorytmów sterowania napędami pneumatycznymi sterowanymi zaworami bistabilnymi. Napędy i sterowania hydrauliczne i pneumatyczne 2009: krajowy sektor w warunkach turbulentnego rynku, międzynarodowa konferencja naukowo-techniczna, Wrocław 7-9 października 2009. Ośrodek Doskonalenia Kadr SIMP, str. 66-73

12 Dworzak Ł., Mikulczyński T.: Automation of foundry processes with Grafpol method. Archives of Metallurgy and Materials. 2010, vol. 55, iss. 3, str. 803-811

13 Dworzak Ł., Mikulczyński T.: Programowanie metodą Grafpol sterowników PLC sterujących procesami technologicznymi w odlewniach. Odlewnictwo XXI wieku - technologie, maszyny i urządzenia odlewnicze: XII Konferencja Odlewnicza TECHNICAL 2010, biuletyn konferencyjny, Nowa Sól, 27-29 maj 2010.str. 135-142

14 Dworzak Ł., Mikulczyński T.: Realizacja pamięci w sekwencyjnych algorytmach sterowania. Interdyscyplinarność badań naukowych: praca zbiorowa pod red. Jarosława Szreka. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2009. str. 75-80

15 Dworzak Ł., Mikulczyński T.: Synteza metodą Grafpol procedur współbieżnych w których realizacja kroków jest sygnalizowana warunkami logicznymi i czasowymi. Acta Mechanica et Automatica. 2010, vol. 4, nr 1, s. 25-29

16 Dworzak Ł., Mikulczyński T.: Synteza sekwencyjnych algorytmów sterowania zawierających kroki czasowe. Pomiary, Automatyka, Robotyka. 2010, R. 14, nr 2, s. 662-669,

17 Dworzak Ł., Mikulczyński T.: Synthesis of sequential control algorithms for pneumatic drives controlled by monostable valves. Archives of Foundry Engineering, 2009, vol. 9, iss. 3, str. 35-40

18 Horowitz P., Hill W.: Sztuka elektroniki – część I. Wydawnictwa Komunikacji i Łączności, Warszawa 1995

19 http://fr.wikipedia.org/wiki/Grafcet

20 http://japura.lurpa.ens-cachan.fr/grafcet/generalites/presentation_uk.html

21 http://pl.wikipedia.org/wiki/Kwantyzacja_(technika)

22 http://pl.wikipedia.org/wiki/Próbkowanie

23 http://www.automatyka.siemens.pl/docs/docs_ia/DOC_AS_PL_S7300x_HW.PDF

24 http://www.automatyka.siemens.pl/docs/docs_ia/S7-300_CPU31xC_and_CPU31x_e.pdf

25 Karnaugh M.: The map method for synthesis of combinational logic circuits. A IEE trans. Comm. and Electronics. vol. 72 part I, November 1953

26 Kwaśniewski J.: Programowalny sterownik SIMATIC S7-300 w praktyce inżynierskiej. Wydawnictwo BTC, Legionowo 2009

27 Kwaśniewski J.: Sterowniki PLC w praktyce inżynierskiej. Wydawnictwo BTC, Legionowo 2008

28 Le Grafcet, ontil de representation du cahier de charges d'un automatisme logique, Rapport final de la Commision AFCET. Paris 1977.

29 Łukowicz M.: Układy logiczne, ćwiczenia laboratoryjne pod redakcją Mirosława Łukowicza. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2002

30 Mikulczyński T., Dworzak Ł., Nowak D.: Synteza równania schematowego sekwencyjnych algorytmów sterowania. Pneumatyka, nr 6, 2006 r., str. 39-43

31 Mikulczyński T., Dworzak Ł.: Metodyka modelowania i programowania procedur współbieżnych bez stosowania elementarnych komórek pamięci. Pneumatyka, nr 4, 2008 r., str. 6-9

32 Mikulczyński T., Samsonowicz Z.: Automatyzacja dyskretnych procesów produkcyjnych: metody modelowania procesów dyskretnych i programowania PLC. WNT, Warszawa 1997

33 Mikulczyński T.: Automatyzacja procesów produkcyjnych – metoda modelowania procesów dyskretnych i programowania sterowników PLC. WNT, Warszawa 2006

34 Mikulczyński T.: Podstawy Automatyki pod redakcją Tadeusza Mikulczyńskiego. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 1998

35 Misiurewicz P.: Układy automatyki cyfrowej. Wydawnictwa Szkolne i Pedagogiczne, Warszawa 1978

36 Norma PN-EN 60848:2003 Język specyfikacyjny GRAFCET do schematów funkcji sekwencyjnych

37 Norma PN-EN 61131-3 Sterowniki programowalne - Część 3: Języki programowania

38 Pełna lista urządzeń programowanych przy pomocy oprogramowania CoDeSys - http://www.3s-software.com/index.shtml?en_Company_ref

39 Pomoc programu CoDeSys 2.3.9.19 dostarczonego przez FESTO (tłumaczenie własne)

40 Specyfikacja techniczna Unitronics V200-18-E2B

41 Veith E.W.: A chart method of simplifing Boolean functions. Proc. of the ACM, May 1952