Zbiór zadań dla sterowników GE-Fanuc serii 90-30/VersaMax/Micro wraz z przykładami rozwiązań

Uwaga:

Zadania zamieszczone w tym zbiorze można zrealizować używając już najmniejszego sterownika serii VersaMax Nano (np. IC200NDR001), jedynie zadanie 5 - ze względu na liczbę wejść wymaga zastosowania modelu IC200UDR005 lub większego i zadanie 9 - ze względu na konieczność zastosowania wejść i wyjść analogowych przy podłączeniu do rzeczywistego obiektu regulacji wymaga zastosowania sterownika IC200UAL006. Każde zadanie da się oczywiście zrealizować na sterowniku serii VersaMax lub 90-30. Do napisania programów sterujących dla sterowników wykorzystano oprogramowanie Cimplicity Machine Edition – Logic Developer PLC. Wszystkie zamieszczone programy zostały przetestowane w działaniu.

Materiały szkoleniowe opracowane przez:

Grzegorza Faracika, Grzegorza Dubiela, Sławomira Dzierżka i Tomasza Michałka

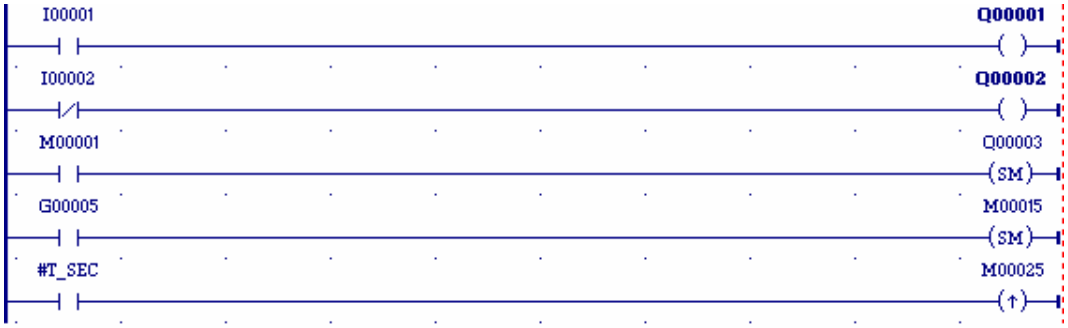
Wydano nakładem
ASTOR sp. z o.o.
31-112 Kraków
ul. Smoleńsk 29
tel. 428-63-00, 428-63-09

Spis treści

Przykłady użycia bloków funkcyjnych

Przykład 1. Elementy logiczne

Poniżej przedstawiono różne rodzaje przekaźników:



Element o adresie I1 jest typu „styk normalnie otwarty” - przewodzi sygnał wtedy, gdy wartość logiczna przypisanej zmiennej jest 1. Element o adresie I2 jest typu „styk normalnie zamknięty” - przewodzi sygnał, gdy przypisana dla niego zmienna ma wartość logiczną 0. Przekaźnik Q1 i Q2 działają w ten sposób, że ich styki są zwierane w momencie dotarcia sygnału. Q3 również zadziała w momencie dotarcia doń sygnału, ale stan ten będzie trwał nawet po odcięciu tego sygnału - jest to przekaźnik z pamięcią. Ustawiony stan w tym przekaźniku będzie trwał nawet po wyłączeniu zasilania sterownika. W przypadku styków takich jak styki o adresie %Q3, %M15 jako atrybuty zmiennej można zadeklarować opcję „retentive” (robimy to w tablicy deklaracji zmiennych). Stan przekaźników z aktywną opcją „retentive” będzie pamiętany nawet po wyłączeniu zasilania sterownika i zostanie odtworzony po ponownym załączeniu zasilania sterownika. M25 to przekaźnik uaktywniany zboczem narastającym sygnału (zmianą wartości logicznej z 0 na 1). Styki tego przekaźnika są zwierane na czas jednego cyklu sterownika.

Nie jest możliwe użycie przekaźnika `-(S)-`, `-(R)-`, `-( )-`, `-(/)-`, itp. jako przekaźnika wprowadzającego sygnał do szczebla, jak również przekaźnika `--] [--` , `--]/[--` do wyprowadzania sygnału ze szczebla.

Przykład 2. Liczniki i przekaźniki czasowe – blok TMR

Program sygnalizujący, że sygnał aktywny na wejściu I1 trwał nieprzerwanie przynajmniej 10 sekund:

Program zrealizowano w oparciu o przekaźnik czasowy bez pamięci. Wejściu I1 przypisany jest styk otwarty - czyli przewodzący sygnał wtedy, gdy wartość logiczna zmiennej I1 jest 1. Załączenie I1 spowoduje uaktywnienie bloku funkcyjnego TMR. Wtedy nastąpi zliczanie czasu, a jego wartość bieżąca przechowywana będzie w rejestrze R1. Właśnie ta wartość wyświetlana jest podczas pracy programatora ONLINE / RUN. Oprócz wartości bieżącej przekaźnika są także inne informacje o nim; przechowywane są w dwóch następnych rejestrach (w naszym przypadku w R2 i w R3). Dlatego dla każdego bloku funkcyjnego TMR należy zarezerwować trzy kolejne rejestry. Jako wartość zadaną ustawiono 100. Jest to wartość czasu podana w dziesiątych częściach sekundy, po osiągnięciu, której na wyjściu przekaźnika pojawi się sygnał logiczny 1. Każdorazowy zanik sygnału na wejściu I1 powoduje wyzerowanie wartości czasu dotychczas zliczonego. Jeżeli jednak sygnał trwa przynajmniej tyle czasu ile wynosi PV, to przekaźnik Q1 zostaje załączony. Licznik zlicza nadal, aż do momentu, w którym zaniknie sygnał I1, po czym jest znów zerowany. Istnieje możliwość zmiany podstawy czasu z dziesiątych części sekundy na setne części.

Przykład 3. Liczniki i przekaźniki czasowe – blok ONDTR

Program sygnalizujący, że sygnał aktywny na wejściu I1 trwał przynajmniej 10 sekund:

Wykorzystano przekaźnik czasowy z pamięcią. Praca jego jest podobna do pracy przekaźnika bez pamięci, a różnica polega na tym, że zlicza on czas gdy dopływa do niego sygnał, i zatrzymuje naliczoną wartość gdy sygnał przestaje dopływać. Dla wyzerowania naliczonej wartości potrzebne jest więc dodatkowe wejście - rolę tą pełni wejście R (*Reset*), sterowane w naszym przypadku przekaźnikiem I2. Na ten licznik trzeba zarezerwować trzy kolejne rejestry.

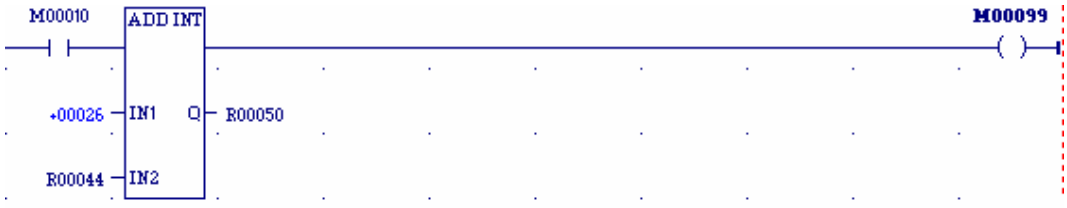
Przykład 4. Liczniki i przekaźniki czasowe – blok UPCTR

Program sygnalizujący, że do wejścia I1 dotarło przynajmniej 10 impulsów:

Realizacja oparta jest o licznik zliczający w górę UPCTR. Służy on do zliczania impulsów od 0 do wartości zadanej (u nas wartość zadana wynosi 10). Każda zmiana sygnału I1 z 0 na 1 powoduje zwiększenie wartości bieżącej o 1. Podanie sygnału I2 powoduje wyzerowanie licznika. Po zrównaniu się z wartością zadaną przesyłany jest sygnał do przekaźnika M1. Na ten licznik trzeba zarezerwować trzy kolejne rejestry.

Przykład 5. Funkcje matematyczne – blok ADD

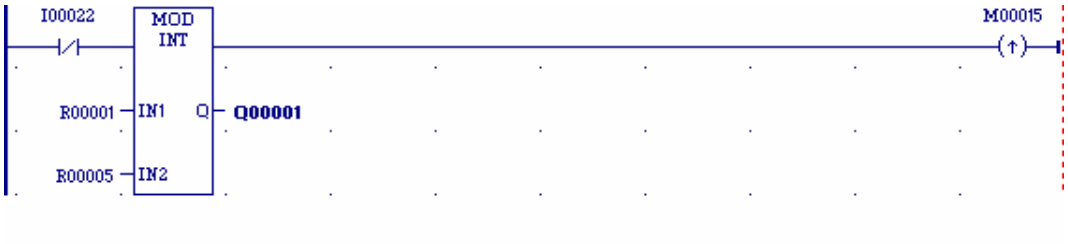
Dodawanie liczby 26 do liczby w rejestrze R44:



Do dodawania liczb wykorzystano blok ADD_INT. Służy on do dodawania liczb całkowitych ze znakiem (16-bitowe). Pierwszy operand to stała równa 26, drugi - liczba w rejestrze R44. Wynik operacji przesyłany jest do rejestru R50. Może się zdarzyć, ze wynik przekracza dopuszczalny zakres wartości; wtedy parametr wyjściowy przyjmuje największą wartość, a sygnał potwierdzenia operacji (sygnał Ok) nie jest przesyłany. Dlatego dobrze jest sprawdzać poprawność wykonywanej operacji, na przykład wpisując bit poprawności wykonania funkcji do wyznaczonego w tym celu rejestru (w przykładzie jest to M99).

Przykład 6. Funkcje matematyczne – blok MOD

Sprawdzanie czy liczba w rejestrze R1 jest wielokrotnością liczby w rejestrze R5:



Do sprawdzenia tego czy liczba w rejestrze R1 jest wielokrotnością liczby w rejestrze R5 sprawdzono czy liczba w R1 dzieli się bez reszty przez liczbę w R5. Posłużono się blokiem funkcyjnym MOD_INT, który wykonuje dzielenie dwóch liczb o typie INT, a wynikiem działania jest reszta z dzielenia. Znak wyniku jest zawsze taki sam jak znak pierwszego parametru wejściowego (u nas liczby w R1). Gdy do bloku dopływa sygnał, czyli I22 jest wyłączone, wykonywane jest dzielenie i wynik jest obliczany w następujący sposób:

$$Q = I1 - (I1 \text{ DIV } I2) * I2$$

(gdzie wynikiem dzielenia DIV jest liczba całkowita).

Wynik równy 0 świadczy o tym że liczba w R1 jest wielokrotnością liczby w R5. (W przykładzie wynik odczytywany jest bezpośrednio na wyjściach, począwszy od Q1).

Przykład 7. Funkcje matematyczne – blok SQRT

Obliczenie pierwiastka kwadratowego z liczby znajdującej się w rejestrze R1:

Do obliczania pierwiastka kwadratowego służy blok SQRT. Przyrostek INT oznacza, że pierwiastek będzie liczony z liczby o pojedynczej precyzji (16 bitów). Gdy do bloku dociera sygnał zezwolenia na pracę (poprzez M1), parametr Q przyjmuje wartość równą części całkowitej pierwiastka z liczby w R1. Sygnał wyjściowy jest przesyłany, gdy wynik operacji nie przekracza dopuszczalnego zakresu wartości oraz gdy pierwiastkowana liczba nie jest ujemna. Wynik pierwiastkowania zostanie umieszczony w szesnastu bitach, począwszy od rejestru M9.

Przykład 8. Relacje matematyczne – blok EQ

Sprawdzenie, czy liczba w rejestrze R1 to liczba 16:

Blok EQ umożliwia porównywanie dwóch liczb. Jeśli parametry wejściowe spełniają relację równości, przesyłany jest sygnał potwierdzenia Q (w przykładzie trafia on do komórki M20).

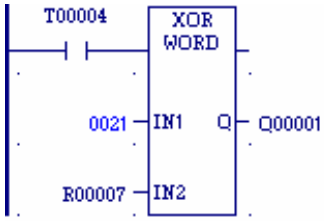
Przykład 9. Operacje bitowe – blok AND

Operacja AND na dwóch słowach bitowych:

Zmienna tymczasowa T2 zezwala na wykonanie operacji koniunkcji dwóch słów bitowych. Funkcja zostanie wykonana na 16 bitach, o adresach początkowych: R1 (pierwsze słowo) i R10 (drugie słowo). Poprawne wykonanie sygnalizowane jest na wyjściu Ok (sygnał z Ok trafia do T3), a wynik umieszczany jest w rejestrze R15. Blok AND może być wykorzystywany do zerowania wybranych rejestrów - gdy jednym z operandów jest liczba zero.

Przykład 10. Operacje bitowe – blok XOR

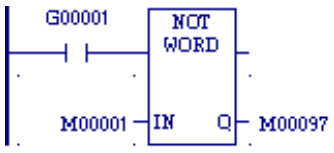
Sprawdzenie czy liczba w rejestrze R7 ma pięć najmniej znaczących bitów postaci: 10101 i które bity odbiegają od tego wzorca:



Liczba dwójkowa 10101 to liczba 21 w systemie dziesiętnym. Dlatego porównywanie odbywać się będzie z liczbą 21. Porównywanie zrealizowano stosując blok XOR. Na każdych dwóch bitach liczb: 21 i liczby w rejestrze R7 wykonywana jest operacja różnicy symetrycznej. Jeżeli którykolwiek bit liczby w rejestrze R7 odbiega od wzorca 10101, to ten bit w wyniku będzie ustawiony na 1. Jeżeli natomiast jest zgodność tych dwóch liczb, to wynikiem operacji będzie zero. Dla wykrycia ewentualnego błędu w wyniku operacji można kontrolować wyjście Ok, jednak przykład pokazuje, że z punktu widzenia poprawności programu niewykorzystanie wyjścia Ok nie jest błędem.

Przykład 11. Operacje bitowe – blok NOT

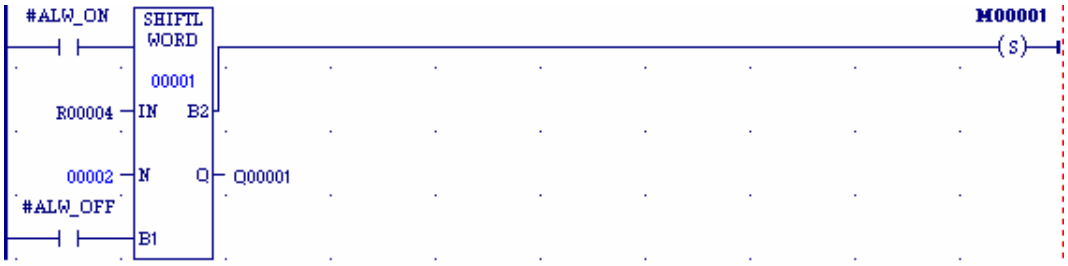
Negacja logiczna słowa bitowego o adresie początkowym M1:



Jeżeli komórka G1 zezwala na wykonanie operacji, to następuje zmiana stanu logicznego każdego bitu na przeciwny. Wynik operacji ulokowany zostanie w szesnastu bitach od adresu M97. Jest możliwość uzyskania potwierdzenia wykonania operacji negacji - sygnał z wyjścia Ok.

Przykład 12. Operacje bitowe – blok SHL

Przesunięcie słowa bitowego w lewo o dwa bity:



Funkcja SHL może być wykorzystywana do przesunięcia wszystkich bitów jednego słowa bitowego lub kilku słów bitowych w lewo, o wyszczególnioną liczbę miejsc. Liczba ta może być zarówno stałą jak i zmienną. Oprócz sygnału zezwalającego na wykonanie operacji (w naszym przypadku wejście zezwalające załączone jest zawsze, bo sygnał pochodzi z przekaźnika ALW_ON, czyli zawsze załączonego) należy podać do bloku adres pierwszego słowa ciągu słów, którego bity mają zostać przesunięte (u nas R4), liczbę bitów, o które ma być przesunięte dane słowo (u nas o 2 bity), oraz wartość bitów które mają zostać wstawione w puste miejsca słowa powstałe po przesunięciu (w naszym przykładzie są to zera). Funkcja zwraca wartość ostatniego bitu, który wyszedł poza zakres słowa po dokonaniu przesunięcia. Należy także wyszczególnić adres pierwszego słowa ciągu słów otrzymanego po przesunięciu bitów słowa (u nas: Q1). Zachodzi także możliwość przesuwania więcej niż jednego słowa bitowego - przez zmianę parametru LEN.

Przykład 13. Operacje bitowe – blok ROR

Przesunięcie słowa bitowego w prawo w obiegu zamkniętym:

W tym przykładzie liczba ulokowana w adresie R4 podlega cyklicznemu przesuwaniu, co sekundę, o dwa bity w prawo. Przesunięcie odbywa się w ten sposób, że najmniej znaczące bity słowa (tzn. z prawej strony słowa) zostają wpisane w puste miejsca po stronie lewej. Wynik operacji przesłany jest na wyjścia, począwszy od adresu Q1. W przykładzie nie korzystano z wyjścia potwierdzającego wykonanie operacji. Ilość przesuwanych słów można zmienić zmieniając parametr LEN.

Przykład 14. Operacje bitowe – blok BIT TEST

Sprawdzenie czy trzeci bit słowa o adresie początkowym w R4 jest jedynką logiczną:

Po uaktywnieniu wejścia I1 Funkcja BTST sprawdza wyszczególniony bit (bit trzeci) podanego słowa bitowego (adres początkowy w R4). Wartość tego bitu jest przesyłana bezpośrednio na wyjście Q bloku funkcyjnego (a stamtąd na wyjście Q3 sterownika). Zachodzi możliwość zmiany liczby słów bitowych ciągu słów, z którego wybierany jest bit do przetestowania (maksymalnie 256).

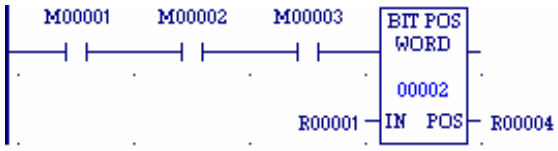
Przykład 15. Operacje bitowe – blok BIT SET

Ustawienie wartości bitu słowa bitowego o adresie początkowym M1 na 1, gdzie numer bitu do ustawienia wyszczególniony jest w R3:

Aby ustawić dany bit słowa bitowego na 1 stosujemy funkcję BSET. Blok posiada wejście zezwalające na wykonanie operacji oraz wyjście potwierdzające wykonanie tejże operacji. Podanie numeru bitu słowa bitowego do ustawienia w 1 jako adresu początkowego umożliwia dynamiczne zadawanie lokacji tego bitu. W razie potrzeby można zmienić liczbę słów bitowych ciągu słów LEN, z którego wybierany jest bit, którego wartość ma zostać ustawiona (maksymalnie 256).

Przykład 16. Operacje bitowe – blok BIT POS

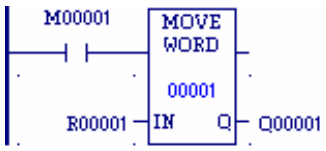
Znaleźć pierwsze wystąpienie jedynki w dwóch słowach bitowych o adresie początkowym w R1:



Do realizacji tego zadania posłużono się funkcją BPOS. Jeżeli zostaną załączone przekaźniki M1, M2 i M3, to zostanie przeprowadzona operacja przeszukiwania dwóch słów bitowych o adresie początkowym R1. Wynikiem operacji jest pozycja pierwszego niezerowego bitu w przeszukiwanych słowach bitowych lub zero, jeżeli w przeszukiwanych słowach występują same zera (wynik operacji umieszczany jest w R4). Ilość przeszukiwanych słów może oczywiście ulec zmianie, gdy zmienimy parametr LEN. Można też obserwować sygnał wyjściowy Ok, pojawiający się po dopłynięciu do bloku funkcyjnego sygnału wejściowego.

Przykład 17. Operacje na danych – blok MOVE

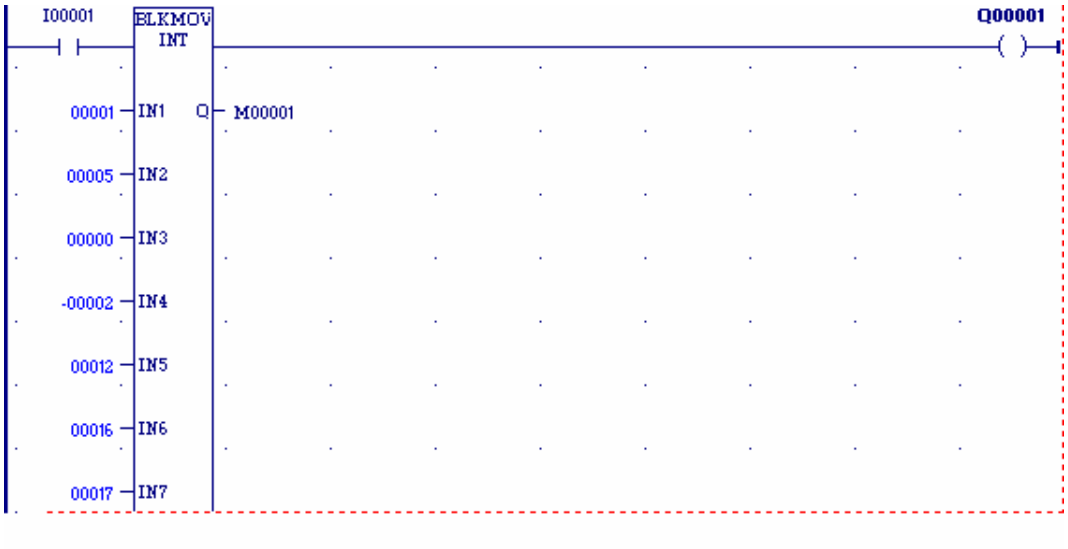
Skopiowanie słowa bitowego z rejestru R1 na wyjścia, od adresu Q1:



Do przemieszczania danych jako pojedynczych bitów służy funkcja MOVE. Ponieważ dane są przesyłane jako bity, nowy adres nie musi odpowiadać temu samemu typowi danych co adres oryginalny. Należy pamiętać, że skopiowanie jednego słowa bitowego na wyjście, począwszy od Q1, spowoduje, że wynik zajmie szesnaście kolejnych lokacji. Gdyby zadać długość LEN równą 2, to wynik zajmie 32 kolejne lokacje, licząc od Q1. Po dopłynięciu sygnału do bloku funkcyjnego pojawia się na wyjściu Ok sygnał potwierdzenia, który może być wykorzystywany lub nie.

Przykład 18. Operacje na danych – blok BLKMOV

Skopiowanie grupy siedmiu stałych wartości do obszaru pamięci rozpoczynającego się od adresu M1:



Stałe: 1, 5, 0, 2, 12, 16, 17 zostaną skopiowane do pamięci od adresu M1, gdy do bloku funkcyjnego BLKMV dotrze sygnał z przekaźnika I1. Każda stała typu INT jest zapisana na szesnastu bitach, więc w sumie zostaną zajęte komórki od M1 do M112. Wykonanie operacji sygnalizowane jest na wyjściu Q1.

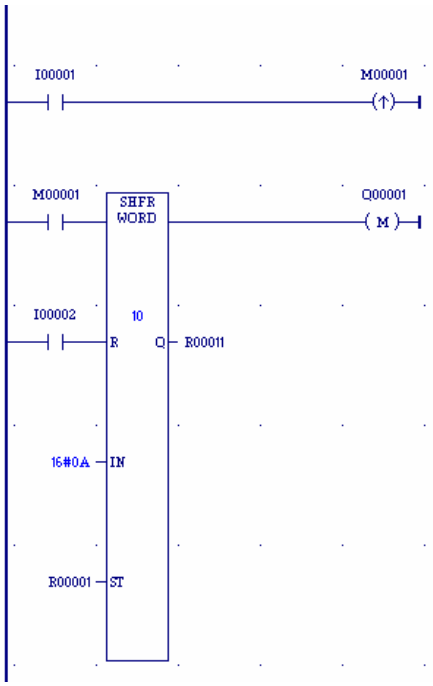
Przykład 19. Operacje na danych – blok BLK CLR

Zerowanie trzech kolejnych słów w pamięci sterownika, poczynając od adresu M1:

Aby wyzerować określoną liczbę kolejno następujących po sobie słów w pamięci sterownika, najlepiej posłużyć się funkcją BLKCLR. Umożliwia ona skasowanie określonego miejsca w pamięci poprzez podanie adresu początkowego słowa i ilości słów do skasowania. Blok zadziała wtedy gdy zostanie załączony przekaźnik I1, a potwierdzenie wykonania operacji zostanie przesłane na wyjście Q1.

Przykład 20. Operacje na danych – blok SHFR

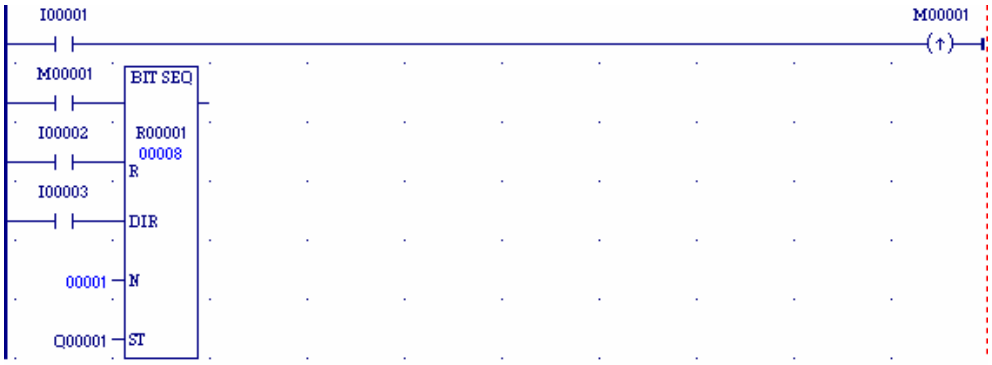
Program tworzący pamięć kolejkową o pojemności 10 liczb (obszar pamięci zawierający 10 liczb, wpisanie kolejnej liczby powoduje przesunięcie pozostałych o jedną pozycję w dół, a ostatnia liczba wysyłana jest pod wskazany adres):



Do budowy pamięci kolejkowej (inaczej pamięci FIFO: *First Input First Output*) służy blok funkcyjny SHFR. Poprzez parametr LEN zadajemy długość tej pamięci, parametr IN zawiera adres początkowy pamięci kolejkowej, a parametr Q - adres, pod który wysyłane są elementy „wypchnięte” z pamięci. Przesuwanie elementów w pamięci odbywa się, gdy do bloku funkcyjnego dopływa sygnał i dokonywane jest tyle razy, ile sterownik wykona cykli. Dlatego jeżeli chcemy przesuwać tylko jeden raz przy jednokrotnym załączeniu I1, to musimy zamienić sygnał z przekaźnika I1 na pojedynczy impuls o czasie trwania równym jednemu cyklowi sterownika - do tego celu wykorzystano przekaźnik M1. Wartość wpisywana do pamięci jest stała i wynosi A w systemie heksadecymalnym, czyli 10 w systemie dziesiętnym. Początek pamięci kolejkowej usytuowany jest w rejestrze R1, a elementy „wypychane” z pamięci trafiają do rejestru R20. Blok posiada wyjście potwierdzające wykonanie operacji, przykładowo podłączone do przekaźnika Q1.

Przykład 21. Operacje na danych – blok BIT SEQ

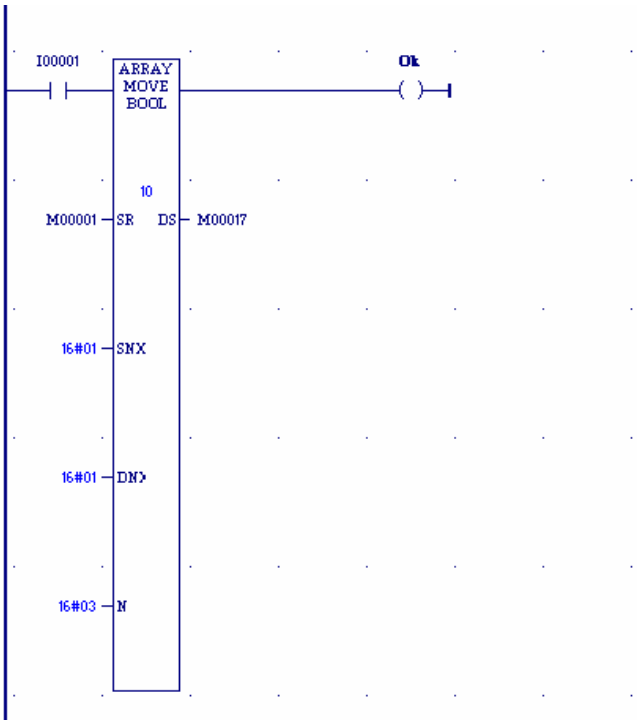
Przesuwanie jedynki od wyjścia Q1 do wyjścia Q8:



Zadanie łatwo jest zrealizować na bazie bloku BIT_SEQ. Powoduje on przesuwanie bitu w górę lub w dół o jedną pozycję. Blok posiada wejście zezwalające na pracę; na wejście to bezpieczniej jest wprowadzić impuls o czasie trwania jednego cyklu - stąd wcześniejszy przekaźnik M1. Sygnał doprowadzony na wejście R powoduje ustawienie warunków początkowych, tzn. ustawienie jedynki tylko w miejscu wyspecyfikowanym na wejściu STEP (ponieważ na wejściu tym jest stała równa 1, to po załączeniu I3 jedynka zostanie ustawiona na pierwszym bicie licząc od adresu początkowego). Dla prawidłowej pracy bloku pozostaje jeszcze tylko podanie adresu pierwszego bitu, na którym działa funkcja BIT_SEQ. Aby umożliwić łatwą obserwację działania układu, jako adres ten podano Q1. Należy również podać parametr LEN, aby wiadomo było do jakiego momentu ma się odbywać przesuwanie bitu. Jeżeli przesuwanie ma się odbywać od Q1 do Q8, czyli o osiem pozycji, należy wpisać LEN=8. Zachodzi możliwość kontrolowania czy operacja została wykonana pomyślnie (wyjście Ok), z możliwości tej jednak w zadaniu nie skorzystano.

Przykład 22. Operacje tablicowe – blok ARRAY MOVE

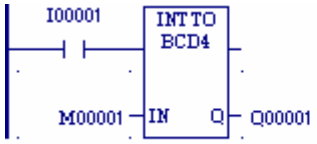
Kopiowanie tablicy 10-elementowej od M1 do M17, z możliwością zadania indeksacji elementów:



Przekaźnik I1 daje zezwolenie na wykonanie operacji. SR określa adres początkowy źródłowej tablicy danych (w przykładzie wynosi on M1), DS - adres początkowy docelowej tablicy danych (równy M17). LEN to liczba elementów, z których składa się tablica źródłowa, jak również docelowa (10 elementów). SNX oraz DNX to indeksy pierwszego z kopiowanych elementów tablicy źródłowej oraz docelowej, natomiast N oznacza liczbę elementów, które mają zostać skopiowane (kopiujemy 3 elementy). Wykonanie operacji sygnalizowane jest na wyjściu Ok.

Przykład 23. Funkcje konwersji – blok INT

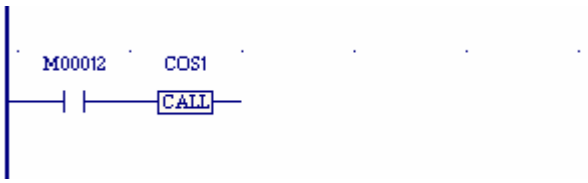
Zamiana liczby całkowitej ze znakiem (typ INT) na liczbę w kodzie BCD. Liczba zapisana jest w szesnastu komórkach o adresie początkowym M1:



Wykorzystano blok funkcyjny dokonujący konwersji znaków typu INT na BCD. Zezwolenie przekaźnika I1 powoduje przeliczenie liczby całkowitej ze znakiem na liczbę BCD i przesłanie jej bezpośrednio na wyjścia, gdzie adresem początkowym liczby po konwersji jest Q1. W przykładzie nie skorzystano z sygnału potwierdzenia wykonania konwersji (sygnał ten pojawia się po dopłynięciu do bloku funkcyjnego sygnału i dokonaniu konwersji bez przekroczenia dozwolonego zakresu wartości).

Przykład 24. Funkcje sterujące – blok CALL

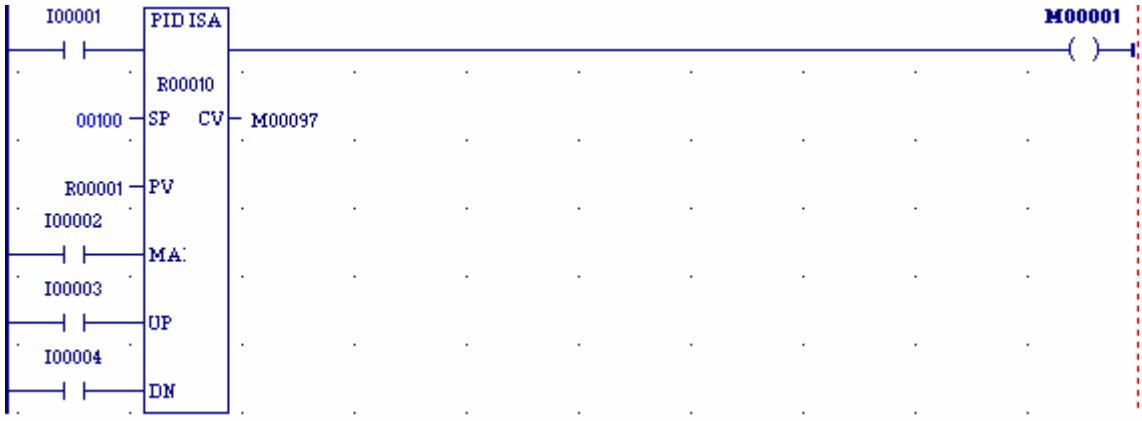
Przywołanie procedury o nazwie „COS1" w danym miejscu programu sterującego:



W momencie, gdy przekaźnik M12 zostanie załączony, nastąpi natychmiastowe przywołanie procedury o nazwie „COS1" i jej kompletne wykonanie, a następnie powrót do punktu następującego bezpośrednio po bloku CALL. Edycja bloku CALL może odbyć się przez ustawienie kursora na tym bloku i naciśnięcie klawisza F10 (Zoom). W bloku deklaracji powinna znaleźć się definicja procedury „COS1" .

Przykład 25. Funkcje sterujące – blok PID

Podłączenie regulatora PID z możliwością ręcznego zadawania parametrów:



Przekaźnik I1 zezwala na pracę regulatora. Wartość SP zawiera wartość zadaną wielkości regulowanej (punkt pracy regulatora), PV jest wielkością regulowaną. Gdy przekaźnik I1 jest wyłączony, to regulator pracuje w trybie automatycznym. Jeżeli natomiast chcemy z jakichś powodów zadać ręcznie parametry wyjściowe, to należy przejść w tryb pracy „manual" - przekaźnik I2 załączony. Teraz mamy możliwość zwiększania nastaw wyjściowych regulatora - I3 lub zmniejszania - I4. Pamiętać trzeba, że regulator PID zajmuje 40 kolejnych rejestrów, więc nie powinniśmy ich używać przez inne bloki funkcyjne (z wyjątkiem niektórych wartości, które może zmienić użytkownik poprzez przesłanie pożądanych wartości do odpowiednich rejestrów sterownika przy użyciu innych bloków funkcyjnych; informacje o tym, których rejestrów uwaga ta dotyczy znajdują się np. w podręczniku programisty „Sterowniki programowalne serii 90-20 i 90-30"). Blok funkcyjny PID wysyła sygnał potwierdzający zrealizowanie algorytmu bez przeszkód. Sygnał wypracowany przez PID w podanym przykładzie przesyłany jest do pamięci od lokacji M97. Aby regulator PID zaczął działać należy zadać mu przynajmniej podstawowe parametry, tj. współczynnik proporcjonalności różny od zera oraz zakres wartości sygnału wyjściowego większy od zera. Można tego dokonać edytując blok funkcyjny PID (ustawić kursor na bloku funkcyjnym i nacisnąć F10).

Wskazówki do programowania sterowników

Wskazówka 1

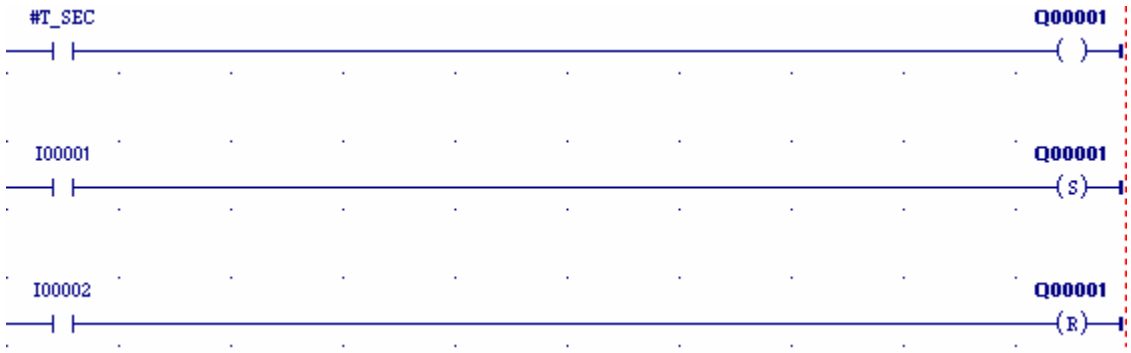
W przypadku korzystania z przekaźników czasowych jak i liczników pamiętać należy o tym, że:

1. Odstęp w ich adresowaniu powinien być nie mniejszy jak 3 rejestry.
2. Niepodanie wartości zadanej PV spowoduje, że wyjście tego bloku funkcyjnego będzie cały czas aktywne.
3. Każdorazowy zanik sygnału zezwalającego *Enable* spowoduje wyzerowanie TMR.

Dla próby proponujemy sprawdzić zasadę pierwszą - ulokować jeden przekaźnik czasowy np. w rejestrze R2, a drugi przekaźnik czasowy np. w rejestrze R3 i sprawdzić, jaki to ma wpływ na pracę tego typu bloków funkcyjnych.

Wskazówka 2

Z dużą rozwagą należy podchodzić do sytuacji, gdy stosujemy różne typy zmiennych dla tej samej komórki rejestru:



W zaprezentowanym przykładzie można jeszcze kontrolować przebieg wykonywania programu. Niestety, w praktyce spotkać się można z o wiele bardziej rozbudowanymi strukturami, zawierającymi instrukcje skoku czy też podprogramy. Wtedy doprowadzenie do konfliktu typów zmiennych powoduje, że przestajemy kontrolować przebieg programu i nie jesteśmy w stanie przewidzieć stanu, w jakim znajdzie się sterownik.

Zaleca się ponadto, aby w programie znalazł się tylko jeden szczebel bezpośrednio sterujący danym wyjściem. Oprogramowanie narzędziowe CIMPLICITY ME posiada odpowiednie narzędzia służące do wyeliminowania podwójnego użycia tego samego przekaźnika.

Wskazówka 3

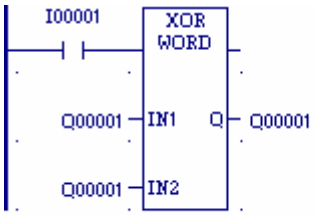
Gdy korzystamy z przekaźników z pamięcią, (aktywną opcją „retentive"). pamiętać trzeba o możliwości zadania warunków początkowych, koniecznych np. przy restarcie programu. Można to zrealizować np. używając do tego celu dodatkowego wejścia, które będzie zerowało określone obszary pamięci sterownika. Pamiętać należy także o zachowaniu informacji np. o miejscu w programie lub o zmiennych, gdyby projektowany system musiał być odporny na zaniki napięcia zasilającego sterownik.

Wskazówka 4

Do wyzerowania bitu pamięci w sterowniku może służyć przekaźnik:

a do wyzerowania słowa bitowego blok XOR, AND, BLK CLR, itp.

Przykładowo podano sposób wyzerowania słowa bitowego zaczynającego się w Q1:



Wskazówka 5

Stosując bloki MOVE możemy dokonywać przemieszczenia bitu, liczby lub słowa bitowego. Przemieszczenia mogą się odbywać pomiędzy wejściami, wyjściami i rejestrami, z uwzględnieniem typów zmiennych. Przesyłając wartości z określonych rejestrów do pamięci bitowej pamiętać trzeba o tym, że jeden rejestr zawiera 16 bitów - więc np. przesłanie trzech rejestrów spowoduje zwiększenie indeksacji adresów obszaru bitowego o 48 (a nie o 3).

Tematy zadań

Zadanie 1.1 Transkoder

Należy napisać program, który będzie zamieniał liczbę załączonych wejść na numer załączonego wyjścia (np. gdy załączymy dowolne dwa wejścia, to fakt ten powinien zostać zasygnalizowany załączeniem wyjścia drugiego - Q2, gdy załączymy dowolne trzy wejścia - powinno to być sygnalizowane załączeniem wyjścia trzeciego - Q3, itp.). Należy ograniczyć się do trzech pierwszych wejść i trzech wyjść.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJ_1	Wejście pierwsze
%I0002	WEJ_2	Wejście drugie
%I0003	WEJ_3	Wejście trzecie
%Q0001	WYJ_1	Wyjście pierwsze
%Q0002	WYJ_2	Wyjście drugie
%Q0003	WYJ_3	Wyjście trzecie

Powodzenia!

Zadanie 1.2 Transkoder strobowany

Zmodyfikować poprzedni program tak, aby stan wyjść zostawał zapamiętywany w momencie podania impulsu „WPIS".

Proponowana tablica deklaracji zmiennych - jak poprzednio.

Ref Address	Name	Description
%I0001	WEJ_1	Wejście pierwsze
%I0002	WEJ_2	Wejście drugie
%I0003	WEJ_3	Wejście trzecie
%I0004	WEJ_WPIS	Wejście wpisujące
%Q0001	WYJ_1	Wyjście pierwsze
%Q0002	WYJ_2	Wyjście drugie
%Q0003	WYJ_3	Wyjście trzecie

Powodzenia!

Zadanie 2.1 Licznik modulo 3

Napisać program na licznik impulsów przychodzących do wejścia I1. Licznik ma liczyć do trzech (podanie czterech impulsów powoduje powrót do stanu wyjściowego). Wejście I4 powinno zerować licznik.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJSCIE	Wejście zliczające
%I0004	RESET	Wejście zerujące

Powodzenia!

Zadanie 2.2 Dodawanie i mnożenie

Zmodyfikować przykład z ostatniego zadania w następujący sposób:

Dodać drugi licznik do trzech, zliczający impulsy z wejścia I2. Wartości zliczone przez liczniki powinny być następnie

a. dodawane

b. mnożone.

W zależności od stanu wejścia I3 na wyjściach Q1...Q4 możemy obserwować wynik odpowiedniej operacji matematycznej w postaci dwójkowej. Jeżeli I3=0, to na Q1Q4 powinien być wynik dodawania, a gdy I3=1, to na wyjściach Q1...Q4 powinien być wynik mnożenia.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJ_1	Wejście zliczające nr 1
%I0002	WEJ_2	Wejście zliczające nr 2
%I0003	TRYB	Przełącznik tego, co ma być na wy
%I0004	RESET	Wejście zerujące

Powodzenia!

Zadanie 2.3 Relacje pomiędzy liczbami

Do programu z poprzedniego zadania dodać układ wykrywania relacji pomiędzy liczbami znajdującymi się w rejestrach R2 i R6. Należy wykryć, czy liczba w pierwszym rejestrze jest większa od liczby w rejestrze drugim. Fakt ten ma być sygnalizowany przełączaniem wyjścia Q8 z częstotliwością 1 Hz.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJ_1	Wejście zliczające nr 1
%I0002	WEJ_2	Wejście zliczające nr 2
%I0003	TRYB	Przełącznik tego, co ma być na wy
%I0004	RESET	Wejście zerujące
%Q0008	SYGNAL	Sygnalizacja zadanej relacji

Powodzenia!

Zadanie 3 Generator fali prostokątnej

Napisać program na układ generujący przebieg prostokątny z możliwością zadawania okresu przebiegu jak i współczynnika wypełnienia. Układ powinien samoczynnie wystartować po włączeniu zasilania oraz powinien mieć wejście synchronizujące STROB (inicjujące pracę od zadanych warunków początkowych). Generator powinien mieć wyjście proste i wyjście zanegowane.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	STROB	Restart liczników
%Q0001	Q	Wyjście proste
%Q0002	NOTQ	Wyjście zanegowane

Powodzenia!

Zadanie 4 Liczniki kaskadowe

Napisać program na układ do zliczania impulsów i przetrzymywania ich liczby w rejestrach. Układ ma zliczać impulsy pochodzące z wejścia I2. Zliczanie powinno się odbywać w następujący sposób: licznik nr 1 ma liczyć od 0001 do 0004, a następnie wysłać impuls do licznika nr 2. Licznik nr 2 ma również liczyć od 0001 do 0004. Aby zapoczątkować zliczanie, trzeba wysłać impuls STARTu do I1. Wtedy powinno nastąpić ustawienie rejestrów R2 i R6 w stan 0001 oraz załączenie Q1. Następnie podajemy na I2 impulsy zliczane. Jednoczesny stan liczników wynoszący 0004 ma zostać wykryty i zasygnalizowany przez wyłączenie Q1 i załączenie Q2; jednocześnie powinno nastąpić wtedy zablokowanie liczników.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	START	Start
%I0002	WEJSCIE	Wejście zliczające
%Q0001	ZLICZAM	Zliczanie
%Q0002	KONIEC	Koniec zliczania

Powodzenia!

Zadanie 5 Sterowanie drzwi w tramwaju

Zaprogramować PLC do pełnienia funkcji układu sterującego otwieraniem drzwi w tramwaju. Każde wejście do tramwaju posiada przycisk żądania otwarcia drzwi. Naciśnięcie go jest pamiętane do momentu wydania zezwolenia przez motorniczego na otwarcie drzwi. On również decyduje o zamknięciu drzwi w danym wagonie. Ma również możliwość otwierania drzwi w danym wagonie (jeden przycisk otwiera wtedy wszystkie drzwi w wagonie). Wydanie zezwolenia na otwarcie drzwi lub otwarcie ich w dowolnym wagonie powinno być sygnalizowane, np. na wyjściu Q8. Gdy motorniczy otworzy drzwi w którymkolwiek wagonie, to naciśnięcie zezwolenia na otwarcie drzwi nie powinno być przyjęte. Przyjąć, że tramwaj ma dwa wagony, a wagon troje drzwi. Wszystkie włączniki - monostabilne.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	ZADA_1	Żądanie otwarcia drzwi 1
%I0002	ZADA_2	Żądanie otwarcia drzwi 2
%I0003	ZADA_3	Żądanie otwarcia drzwi 3
%I0004	ZADA_4	Żądanie otwarcia drzwi 4
%I0005	ZADA_5	Żądanie otwarcia drzwi 5
%I0006	ZADA_6	Żądanie otwarcia drzwi 6
%I0007	ZEZWOLE	Zezwolenie otwarcia drzwi
%I0008	ZANKN_1	Zamknij wagon pierwszy
%I0009	ZAMKN_2	Zamknij wagon drugi
%I00010	OTW_PIR	Otwórz wagon pierwszy
%I00011	OTW_DRU	Otwórz wagon drugi
%Q0001	DRZWI_1	Otwarcie drzwi 1
%Q0002	DRZWI_2	Otwarcie drzwi 2
%Q0003	DRZWI_3	Otwarcie drzwi 3
%Q0004	DRZWI_4	Otwarcie drzwi 4
%Q0005	DRZWI_5	Otwarcie drzwi 5
%Q0006	DRZWI_6	Otwarcie drzwi 6
%Q0008	SYGNAL	Sygnalizacja otwarcia drzwi

Powodzenia!

Zadanie 6.1 Sterowanie windą dwu poziomową

Zaprojektować sterowanie windą 2-poziomową, w której wyróżniamy następujące sygnały sterujące:

I1 - żądanie jazdy w dół,

I2 - żądanie jazdy w górę,

Sygnałami wyjściowymi są:

Q5 - włączenie silnika do jazdy w dół,

Q6 - włączenie silnika do jazdy w górę.

Żądanie jazdy może być przyjęte dopiero po zakończeniu poprzedniego cyklu.

Cykl składa się z:

1. włączenia silnika do jazdy w odpowiednim kierunku (na czas 5 sekund)
2. czasu oczekiwania po dojechaniu do odpowiedniego poziomu (3 sekundy).

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	W_DOL	Żądanie jazdy w dół
%I0002	W_GORE	Żądanie jazdy w górę
%Q0005	SIL_DOL	Silnik w dół
%Q0006	SIL_GOR	Silnik w górę

Powodzenia!

Zadanie 6.2 Sterowanie windą dwu poziomową z czujnikami

Zaprojektować sterowanie windą 2-poziomową posiadającą możliwość włączenia silnika do jazdy szybkiej i jazdy wolnej w dół lub w górę. Najpierw winda włącza silnik do jazdy szybkiej, natomiast przy dojeżdżaniu do danego poziomu wyłącza go i załącza silnik do jazdy wolnej. Winda posiada czujniki:

- czujnik dojeżdżania windy w dół,
- czujnik dojeżdżania windy w gore,
- czujnik dojechania windy w dół,
- czujnik dojechania windy w gore.

W zależności od tego, w którym kierunku winda jedzie, fakt ten powinien być sygnalizowany odpowiednią kontrolką w kabinie.

Zasada pracy windy:

Żądanie jazdy przyjmowane jest dopiero po zakończeniu jazdy. Generuje ono jeden cykl pracy windy. Po przyjęciu żądania jest ono bezpośrednio realizowane. Operator windy ma mieć możliwość natychmiastowego zatrzymania windy w dowolnym miejscu. Winda nie powinna przyjąć żądania jazdy na poziom, na którym aktualnie się znajduje.

Cykl składa się z:

1. włączenia silnika do jazdy szybkiej w odpowiednim kierunku,
2. wyłączenia silnika szybkiego i włączenia wolnego (po przyjęciu sygnału z czujnika dojeżdżania do odpowiedniego poziomu),
3. wyłączenia silnika jazdy wolnej (po otrzymaniu sygnału o dojechaniu do danego poziomu) i wyłączenia kontrolki w kabinie,
4. czasu oczekiwania (3 sekundy).

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	W_DOL	Żądanie jazdy w dół
%I0002	W_GORE	Żądanie jazdy w górę
%I0004	STOP	Stop
%I0005	CZU_DOL	Czujnik dojeżdżania w dół
%I0006	CZU_GOR	Czujnik dojeżdżania w górę
%I0007	DOL	Winda dojechała w dół
%I0008	GORA	Winda dojechała w górę
%Q0001	KON_DOL	Kontrolka jazdy w dół
%Q0002	KON_GOR	Kontrolka jazdy w górę
%Q0003	WOL_DOL	Jazda wolna w dół
%Q0004	WOL_GOR	Jazda wolna w górę
%Q0005	SZY_DOL	Jazda szybka w dół
%Q0006	SIL_SZY	Jazda szybka w górę

Powodzenia!

Zadanie 7 Linia napełniania kartonów

Zaprojektować sterowanie linią do napełniania kartonów zadaną ilością płynu. Linia zbudowana jest w oparciu o taśmociąg, wyposażony w czujniki I1, I2. Operator posiada przyciski do sterowania: I3 i I4 (monostabilne).

I1 – przesunięto o jeden karton,

I2 – karton jest obecny pod zaworem,

I3 – START,

I4 – STOP,

Q1 – otwarcie zaworu do napełniania,

Q2 – włączenie silnika do przesuwu taśmociągu.

Po przesunięciu o jeden karton powinno nastąpić zatrzymanie taśmociągu na czas 4 s. Gdy w danym miejscu znajduje się karton, powinno nastąpić jego napełnianie przez czas 3 s. Po zatrzymaniu linii przyciskiem STOP lub po zaniku napięcia zasilającego powinno nastąpić zachowanie aktualnego stanu pracy linii w pamięci sterownika, tak aby po ponownym jej uruchomieniu nie nastąpiło zarówno przelanie kartonu jak i też jego ominięcie.

Ref Address	Name	Description
%I0001	KON_PRZ	Przesunięto jeden w dół
%I0002	JEST	Jest karton
%I0003	PRACA	Praca start
%I0004	STOP	Praca stop
%Q0001	OTW_ZAW	Otwarcie zaworu
%Q0002	PRZESUW	Przesuw taśmy
%Q0007	K_STOP	Kontrolka: STOP
%Q0008	K_START	Kontrolka: STRAT
%M0001	CZEKAJ	Oczekiwanie po przesuwie
%M0002	ZAWOR	Otwórz zawór
%M00010	START	Cykl trwa
%M00011	ZAKONCZ	Zakończ cykl
%M00013	ZBO_STA	Zbocze start
%M00015	ZBO_STO	Zbocze stop
%M00016	KONIEC	Koniec napełniania
%M00017	ZERUJ	Koniec cyklu
%M00019	ODCZYT	Odtwórz stan pracy
%M00020	ZAPIS	Zachowaj stan pracy
%M00021	PRZ_ZBO	Zbocze przesunięcia

Powodzenia!

Zadanie 8 Sygnalizacja świetlna

Zaprojektować sterowanie sygnalizacją świetlną na przejściu dla pieszych. Należy założyć, że układ będzie generował siedem stanów odpowiadających następującym zdarzeniom:

1. pojazdy mają światło zielone, piesi - czerwone,
2. pojazdy - żółte, piesi - czerwone,
3. pojazdy - czerwone, piesi czerwone,
4. pojazdy - czerwone, piesi - zielone,
5. pojazdy - czerwone,
6. pojazdy - czerwone i żółte, piesi - czerwone,
7. pojazdy - żółte.

Każdemu z tych zdarzeń przypisać należy flagę (przykładowo flaga 1 będzie powodowała zaświecenie światła zielonego dla pojazdów i światła czerwonego dla pieszych). Wyróżniamy także dwa tryby pracy układu:

a. tryb „pulsowanie”,

b. tryb „sterowanie”.

Po włączeniu zasilania do sterownika powinien on przejść w tryb „pulsowanie” - pulsowanie światła żółtego. Pulsowanie powinno się odbywać z okresem równym 2 s. W momencie załączenia wejścia I2 sterownik wchodzi w tryb „sterowanie”. Następuje zaświecenie światła zielonego dla pojazdów i światła czerwonego dla pieszych (flaga 1). Układ oczekuje na naciśnięcie przycisku żądania zielonego światła dla pieszych (wejście I1). Gdy to żądanie wystąpi, układ sprawdza czy światło zielone dla pojazdów świeciło się przynajmniej przez 30 sekund. Jeżeli tak było, to układ inicjuje cykl zapalenia światła zielonego dla pieszych. Jeżeli natomiast światło zielone dla pojazdów świeciło się przez czas krótszy niż 30 sekund, to układ pamięta o zgłoszonym żądaniu, lecz z jego realizacją czeka aż upłyną wspomniane 30 sekundy. Zgłoszenie żądania pieszych jest sygnalizowane zaświeceniem kontrolki na słupie (Q6). Układ nie powinien jednak przyjąć żądania pieszych w przypadku, gdy jest on właśnie w trakcie realizacji takiego żądania (z poprzedniego cyklu).

Wykonanie cyklu jest następujące:

- Zaświecenie światła żółtego dla pojazdów.
- Zgaszenie światła żółtego dla pojazdów i zaświecenie światła czerwonego dla pojazdów (po upływie 5 sekund od ostatniej zmiany).
- Zaświecenie światła zielonego dla pieszych (po upływie 3 sekund od poprzedniej zmiany).
- Mruganie światła zielonego dla pieszych (po upływie 15 sekund od ostatniej zmiany).
- Zgaszenie światła zielonego dla pieszych i zaświecenie światła czerwonego dla pieszych (po upływie 6 sekund od ostatnie zmiany).
- Zaświecenie światła żółtego (po upływie 3 sekund od ostatniej zmiany).
- Zgaszenie światła czerwonego i żółtego dla pojazdów i zaświecenie zielonego (po 3 sekundach od ostatniej zmiany).
- Po zakończeniu cyklu należy zgasić kontrolkę zgłoszenia żądania na słupie.
- Przyjąć, że wszystkie włączniki są monostabilne (po puszczeniu wracają samoczynnie do pozycji wyjściowej).

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	ZAD PIE	Żądanie pieszych
%I0002	STEROW	Przejście w tryb sterowanie
%I0003	PULSUJ	Powrót do trybu pulsowanie
%Q0001	POJ CZE	Światło czerwone dla pojazdów
%Q0002	POJ ZIE	Światło zielone dal pojazdów
%Q0003	PIE CZE	Światło czerwone dla pieszych
%Q0005	PIE ZIE	Światło zielone dla pieszych
%Q0006	SLUP	Kontrolka na słupie
%M0001	FLAGA1	Pojazdy: ziel, piesi: czerw
%M0002	FLAGA2	Pojazdy: żółte, piesi: czerw
%M0003	FLAGA3	Pojazdy: czerw, piesi: czerw
%M0004	FLAGA4	Pojazdy: czerw, piesi: ziel
%M0005	FLAGA5	Pojazdy: czerw
%M0006	FLAGA6	Pojazdy: czerw, żółte, piesi: czerw
%M0006	FLAGA7	Pojazdy: żółte

Powodzenia!

Zadanie 9 Regulator PID

Napisać program do sterowania grzałką tak, aby temperatura w pomieszczeniu wynosiła tyle, ile zadano w rejestrze R2. Wartość tej temperatury może być zmieniana przez operatora w granicach od 0 do 50 jednostek. Należy przyjąć, że różnica temperatury zadanej i regulowanej o 1 stopień powinna wywołać wartość sterującą równą 1 jednostce (współczynnik wzmocnienia P=1). Wartość sterująca powinna zawierać się w granicach 0 ... 32000 jednostek. Należy także uwzględnić zalecenie, aby grzałka nie otrzymywała zbyt gwałtownych uderzeń prądowych (ograniczyć minimalny czas reakcji regulatora PID na skokową zmianę wartości zadanej do maksymalnie 100% w ciągu 10 sekund). Przyjąć że wartość zadana znajduje się w rejestrze R2, wartość regulowana - w R6, wartość sterująca - w R10.

Można również uwzględnić współczynnik całkujący równy 0.063 rep/s i zaobserwować zachowanie regulatora.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	PRACA	Regulator wyłączony
%I0002	MANUAL	Sterowanie ręczne
%I0003	GORA	Regulacja w górę
%I0004	DOL	Regulacja w dół
%I0007	ZWIEKSZ	Zwiększ wartość zadaną
%I0008	ZMINEJS	Zmniejsz wartość zadaną
%Q0001	DZIALA	Regulator pracuje
%R0002	ZADANA	Wartość zadana
%R0006	REGUL	Wartość wielkości regulowanej
%R0010	STERUJ	Wartość sterująca
%R0018	REJ_PID	Rejestr bloku PID

Powodzenia!

Zadania o ziększonym stopniu zaawansowania

Zadanie 10 Odczyt daty i czasu z zegara kalendarzowego w sterowniku

Ćwiczenie to można przeprowadzić korzystając ze sterownika posiadającego zegar czasu rzeczywistego, np.: IC693UDR005, IC693UAL006, IC693CPU331 i jednostek wyższych.

Napisać program dokonujący cyklicznego odczytu daty i czasu (rok, miesiąc, dzień, godzina, minuta, sekunda, dzień tygodnia) co 1 sekundę. Dla operacji odczytu zegara są do dyspozycji rejestry od R1 do R6.

Proponowana tablica zmiennych:

Ref Address	Name	Description
%T0001	POTWIE	Odczyt zegara OK.
%R0001	TRYB	0=odczyt, 1= zapis
%R0002	ZAW_1	Zawsze 1

Powodzenia!

Zadanie 11 Sterownie silnikami krokowymi

Ćwiczenie niniejsze można przeprowadzić korzystając ze sterownika Micro-90 posiadającego przynajmniej jedno wyjście tranzystorowe, np. IC693UDR005, IC693UAL006 itp.

Napisać program sterujący silnikiem krokowym w następujący sposób: po doprowadzeniu do wejścia I1 sygnału logicznego „1” ma zostać wygenerowanych 50 impulsów o częstotliwości 20Hz.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJ1	Sygnał startu od operatora
%Q0494	START	Wygeneruj impuls
%T001	ZBOCZE	Zmienna pomocnicza
%AQ123	CZESTOT	Częstotliwość impulsów
%AQ124	ILOSC	Ilość impulsów

Powodzenia!

Zadanie 12 Komunikacja w protokole SNP

Napisać program, przy pomocy, którego możliwe będzie nawiązanie komunikacji w protokole SNP pomiędzy sterownikiem nadrzędnym (master) - sterownik Micro 23- lub 28-punktowy, a sterownikiem podrzędnym (slave) - sterownik Micro 14-punktowy. Zadaniem sterownika nadrzędnego jest, po nawiązaniu komunikacji, odczytanie wartości 8 wyjść ze sterownika podrzędnego oraz zapisanie 6 wejść w tym sterowniku. Program należy napisać w sterowniku nadrzędnym.

Proponowana tablica zmiennych:

Ref Address	Name	Description
%R0301	STATUS	Status komunikacji
%R0302	OPOZN	Opóźnienie komunikacji
%R0305	DANE1	Pierwsze słowo bloku danych
%R0312	DANE8	Ósme słowo bloku danych
%R0325	LICZNIK	Wartość licznika
%M0801	RESET	Reset licznika
%T0201	START	Załączenie timera
%T0202	KOM_1SK	Nawiązanie komunikacji w 1 skanie programu
%T0203	BLAD	Błąd komunikacji
%T0204	KOMUNIK	Nawiązanie komunikacji
%T0205	ZM_POM	Zmienna pomocnicza
%T0206	PO_AW	Nawiązanie komunikacji po awarii

Powodzenia!

Zadanie 13 Komunikacja w protokole SNP-X

Napisać program, przy pomocy, którego możliwe będzie nawiązanie komunikacji w protokole SNP-X pomiędzy sterownikiem nadrzędnym (master) - sterownik serii 90-30, a dwoma sterownikami podrzędnymi (slave) - sterowniki Micro 14-punktowe. Zadaniem sterownika nadrzędnego jest, po nawiązaniu komunikacji, odczytanie wartości rejestru pierwszego z jednego i drugiego sterownika podrzędnego i umieszczenie wartości tych rejestrów, odpowiednio w rejestrze 100 i 120. Program należy napisać w sterowniku nadrzędnym.

Proponowana tablica zmiennych:

Ref Address	Name	Description
%R0001	STATUS	Status komunikacji
%R0002	OPOZN	Opóźnienie komunikacji
%R0005	DANE1	Pierwsze słowo bloku danych
%R0012	DANE8	Ósme słowo bloku danych
%R0019	DANE15	Piętnaste słowo bloku danych
%R0026	DANE22	Dwudzieste drugie słowo bloku danych
%R0050	LICZNIK	Wartość licznika
%M0001	RESET	Reset licznika
%T0001	START	Załączenie timera
%T0002	KOM_1SK	Nawiązanie komunikacji w 1 skanie programu
%T0003	BLAD	Błąd komunikacji
%T0004	KOMUNIK	Nawiązanie komunikacji
%T0005	ZM_POM1	Zmienna pomocnicza 1
%T0006	ZM_POM2	Zmienna pomocnicza 2

Powodzenia!

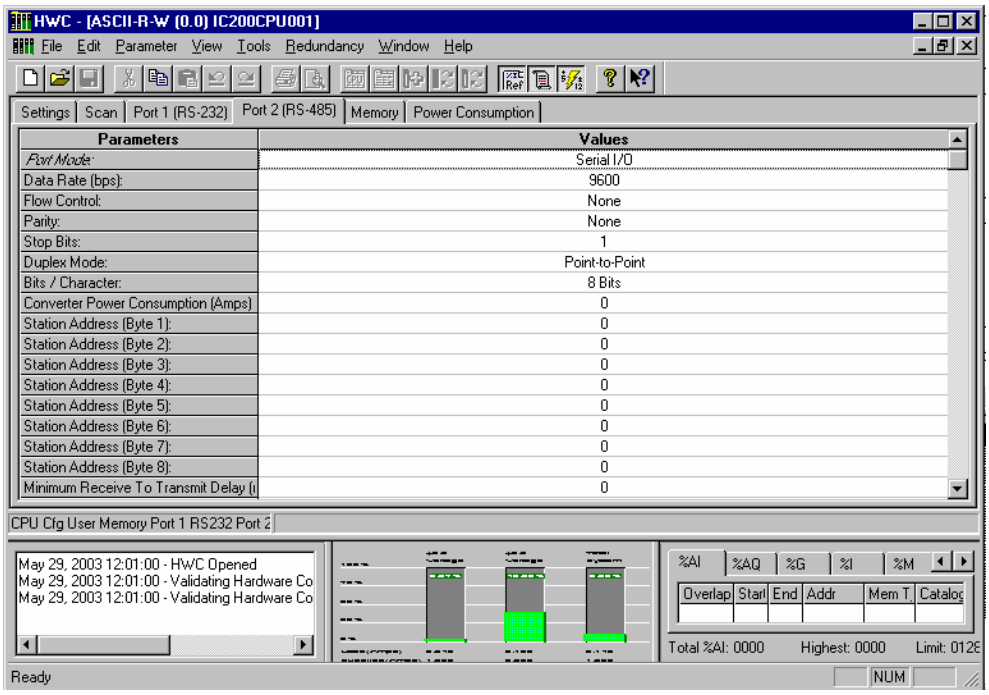
Zadanie 14 Przesyłanie danych przez port szeregowy

Napisać program realizujący cykliczne, co 1 sekundę, wysyłanie poprzez port szeregowy w formacie ASCII ciągu znaków: ABCD. Program ma wyłącznie wysyłać znaki ASCII, nie należy implementować procedur do odczytu znaków ASCII z portu szeregowego.

Proponowana tablica zmiennych:

Ref Address	Name	Description
%R00002	IloscSlowDoWyslania	Ilość słów do wysłania
%R00005	NrPortu	Numer portu (1=19, 2=20)
%R00050	Tmr1	Timer 1
%R00053	Tmr2	Timer 2
%R00056	Tmr3	Timer 3
%R00059	Tmr4	Timer 4
%R00100	StatusFunkcjiComreq	Status komunikacji
%R00101	ComInit	Operacja COMREQ Init Port
%R00115	ComCancel	Operacja COMREQ cancel
%R00122	ComCancel1	Operacja COMREQ cancel - cd
%R00123	ComPortStatus	Operacja COMREQ Get Port Status
%R00129	ComPortStatus1	Operacja COMREQ Get Port Status – cd
%R00143	ComWrite	Operacja COMREQ Write Bytes
%R00150	ComWrite1	Operacja COMREQ Write Bytes - cd
%M00001	ComFault	COMREQ Fault Marker – sygnał błędu wywołania funkcji - COMREQ
%T00001	Sekwencja	Sekwencja kroków (typ zmiennej - word)
%T00001	Krok1	Krok1
%T00002	Krok2	Krok2
%T00003	Krok3	Krok3
%T00004	Krok4	Krok4
%T00005	Krok5	Krok5
%T00006	Krok6	Krok6
%T00007	Krok7	Krok7
%T00008	Krok8	Krok8
%T00009	Krok9	Krok9
%T00010	Krok10	Krok10
%T00011	Krok11	Krok11
%T00012	Krok12	Krok12
%T00013	Krok13	Krok13
%T00014	Krok14	Krok14
%T00015	Krok15	Krok15
%T00016	Krok16	Krok16
%I00001	Wejscia	Wejścia lokalne
%I00065	SlowoStatusowePortu	Port Status Word – słowo statusowe portu
%I00076	WS	Write Success – sukces wysyłania znaków
%I00079	RS	Read Success – sukces odczytu
%G00001	Impuls_1min	Impuls co 1 minutę

Przykładowa konfiguracja portu do wysyłania znaków ASCII:



Powodzenia!

Informacje pomocnicze do zadań

Zadanie 1.1 Transkoder

Proponowana tablica deklaracji zmiennych nie ulega zmianie:

Ref Address	Name	Description
%I0001	WEJ_1	Wejście pierwsze
%I0002	WEJ_2	Wejście drugie
%I0003	WEJ_3	Wejście trzecie
%Q0001	WYJ_1	Wyjście pierwsze
%Q0002	WYJ_2	Wyjście drugie
%Q0003	WYJ_3	Wyjście trzecie

Nie zachodzi potrzeba stosowania dodatkowych zmiennych jak i bloków funkcyjnych.

Dla rozwiązania zadania można napisać tabelę prawdy, czyli wszystkie kombinacje sygnałów wejściowych i odpowiadające im sygnały wyjściowe.

Zadanie 1.2 Transkoder strobowany

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJ_1	Wejście pierwsze
%I0002	WEJ_2	Wejście drugie
%I0003	WEJ_3	Wejście trzecie
%I0004	WE_WPIS	Wyjście wpisujące
%Q0001	WYJ_1	Wyjście pierwsze
%Q0002	WYJ_2	Wyjście drugie
%Q0003	WYJ_3	Wejście trzecie
%M0001	FLAGA1	Do załączenia Q1
%M0002	FLAGA2	Do załączenia Q2
%M0003	FLAGA3	Do załączenia Q3
%M0009	IMPULS	Impuls zerujący
%M0010	WPIS	Dokonanie wpisu

Nie zachodzi potrzeba stosowania bloków funkcyjnych.

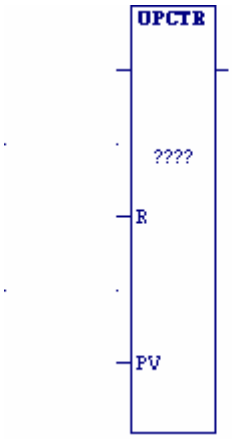
Zadanie 2.1 Licznik modulo 3

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJSCIE	Wejście zliczające
%I0004	RESET	Wejście zerujące
%M0001	POWROT	Impuls zerujący

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:



E – enable: umożliwia pracę bloku funkcyjnego

R – reset: stan wysoki tego wejścia zeruje licznik

PV – preset value: wartość zadana

Q – wyjście: jest aktywne, gdy aktualna wartość rejestru roboczego jest równa PV

Zadanie 2.2 Dodawanie i mnożenie

Proponowana pełna tablica deklaracji zmiennych:

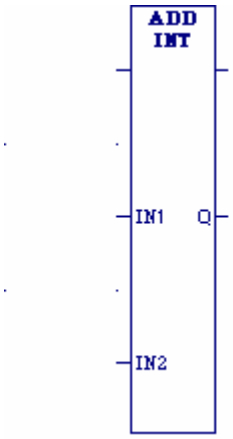
Ref Address	Name	Description
%I0001	WEJ_1	Wejście zliczające nr 1
%I0002	WEJ_2	Wejście zliczające nr 2
%I0003	TRYB	Przełącznik tego, co ma być na wyjściu
%I0004	RESET	Wejście zerujące
%M0001	POWROT1	Impuls zerujące licznik nr 1
%M0002	POWROT2	Impuls zerujący licznik nr 2

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:

Opis bloku podano w zadaniu 21

2. Dodawanie dwóch liczb:



E – enable: umożliwia pracę bloku funkcyjnego

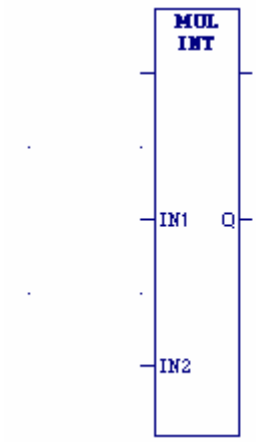
I1 – parametr wejściowy pierwszy

I2 – parametr wejściowy drugi

Q – wyjście: jest aktywne, gdy aktualna wartość rejestru roboczego jest równa PV

OK – potwierdzenie poprawnego wykonania działania

3. Mnożenie dwóch liczb:



E – enable: umożliwia pracę bloku funkcyjnego

I1 – parametr wejściowy pierwszy

I2 – parametr wejściowy drugi

Q – wyjście: jest aktywne, gdy aktualna wartość rejestru roboczego jest równa PV

OK - potwierdzenie poprawnego wykonania działania

Zadanie 2.3 Relacje pomiędzy liczbami

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJ_1	Wejście zliczające nr 1
%I0002	WEJ_2	Wejście zliczające nr 2
%I0003	TRYB	Przełącznik tego, co ma być na wyjściu
%I0004	RESET	Wejście zerujące
%Q0008	SYGNAL	Sygnalizacja zadanej relacji
%M0001	POWROT1	Impuls zerujący licznik nr 1
%M0002	POWROT2	Impuls zerujące licznik nr 2
%M0005	MNIEJ	Liczba w rej. R2 jest < od liczby w rej. R6

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:

Opis bloku - w zadaniu 2.1

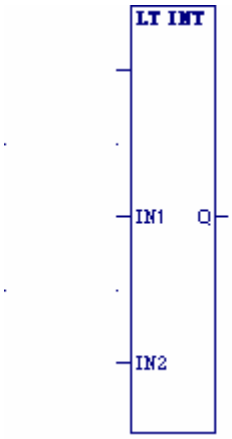
2. Dodawanie dwóch liczb:

Opis bloku - jak w zadaniu 2.2

3. Mnożenie dwóch liczb:

Opis bloku - jak w zadaniu 2.2

4. Relacja matematyczna mniejszości:



E – enable: umożliwia pracę bloku funkcyjnego

I1 – parametr wejściowy pierwszy

I2 – parametr wejściowy drugi

Q – sygnał wyjściowy pojawiający się gdy parametry I1 i I2 spełniają relację

Zadanie 3 Generator fali prostokątnej

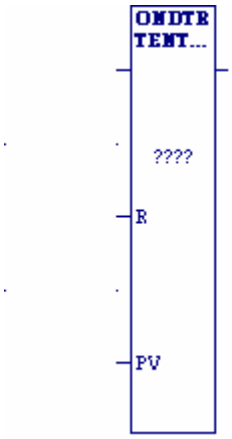
Okres przebiegu jak i jego wypełnienie można regulować zmieniając wartość PV zadaną (*Preset Value*).

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	STROB	Restart liczników
%Q0001	Q	Wyjście proste
%Q0002	NOTQ	Wyjście zanegowane
%M0001	ZERUJ1	Zerowanie licznika 1
%M0002	ZATRZY1	Zatrzymaj licznik 1
%M0003	ZATRZY2	Zatrzymaj licznik 2
%M0004	ZRERUJ2	Zerowanie licznika 2
%M0009	PIK	Zamiana sygnału na impuls
%M0010	RESET	Reset liczników

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:



E – enable: umożliwia pracę bloku funkcyjnego

R – reset: wejście zerujące

PV – preset value: wartość zadana

Q – sygnał wyjściowy informujący o zrównaniu lub przekroczeniu wartości zadanej przez stan rejestrów

Zadanie 4 Liczniki kaskadowe

Proponowana pełna tablica deklaracji zmiennych:

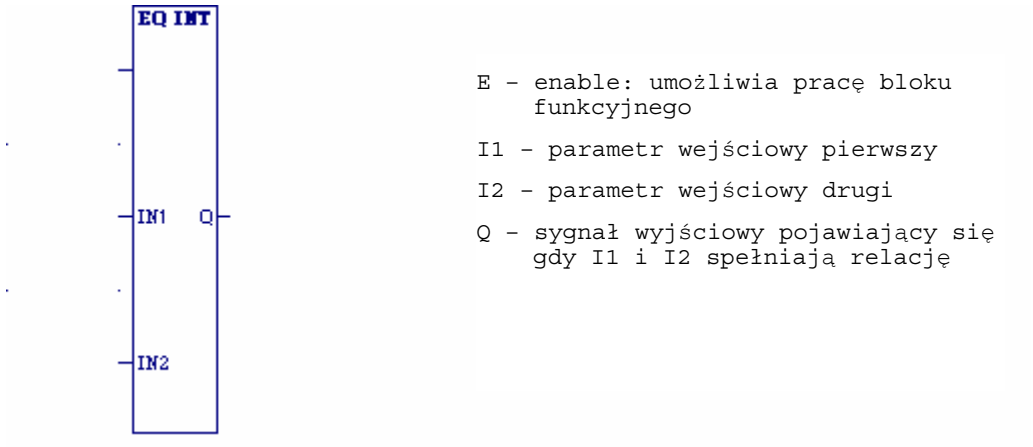
Ref Address	Name	Description
%I0001	START	Start
%I0002	WEJSCIE	Wejście zliczające
%Q0001	ZLICZAM	Zliczanie
%Q0002	KONIEC	Koniec zliczania
%M0001	KASOW1	Kasowanie licznika 1
%M0003	KASOW2	Kasowanie licznika 2
%M0004	WEJ1	Wejście licznika 1
%M0005	WEJ2	Wejście licznika 2
%M0006	R6=0000	Wykrywanie czy R6=0000
%M0010	R2=0004	Wykrywanie czy R2=0004
%M0011	R6=0004	Wykrywanie czy R6=0004
%M0040	RESET	Reset

Proponowane do wykorzystania bloki funkcyjne:

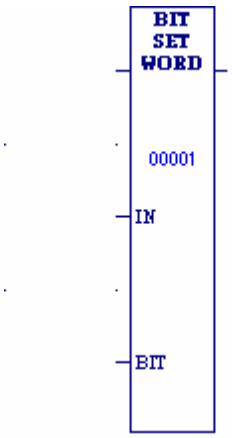
1. Licznik zliczający w górę:

Opis w zadaniu 2.1

2. Relacja matematyczna równości:



3. Ustawianie wartości danego bitu ciągu bitowego na 1:



E – enable: umożliwia pracę bloku funkcyjnego

IN – adres pierwszego słowa ciągu słów

BIT – numer bitu słowa IN, którego wartość ma zostać sprawdzona

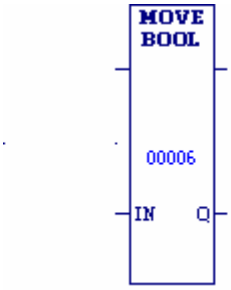
Zadanie 5 Sterowanie drzwiami tramwaju

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	ZADA_1	Żądanie otwarcia drzwi 1
%I0002	ZADA_2	Żądanie otwarcia drzwi 2
%I0003	ZADA_3	Żądanie otwarcia drzwi 3
%I0004	ZADA_4	Żądanie otwarcia drzwi 4
%I0005	ZADA_5	Żądanie otwarcia drzwi 5
%I0006	ZADA_6	Żądanie otwarcia drzwi 6
%I0007	ZEZWOLE	Zezwolenie otwarcia drzwi
%I0008	ZAMKN_1	Zamknij wagon pierwszy
%I0009	ZAMKN_2	Zamknij wagon drugi
%I0010	OTW_PIE	Otwórz wagon pierwszy
%I0011	OTW_DRU	Otwórz wagon drugi
%Q0001	DRZI_1	Otwarcie drzwi 1
%Q0002	DRZI_2	Otwarcie drzwi 2
%Q0003	DRZI_3	Otwarcie drzwi 3
%Q0004	DRZI_4	Otwarcie drzwi 4
%Q0005	DRZI_5	Otwarcie drzwi 5
%Q0006	DRZI_6	Otwarcie drzwi 6
%Q0008	SYGNAL	Sygnalizacja otwarcia drzwi
%M0001	ZAP_1	Zapamiętanie żądania 1
%M0002	ZAP_2	Zapamiętanie żądania 2
%M0003	ZAP_3	Zapamiętanie żądania 3
%M0004	ZAP_4	Zapamiętanie żądania 4
%M0005	ZAP_5	Zapamiętanie żądania 5
%M0006	ZAP_6	Zapamiętanie żądania 6
%M0008	OTW_2	Sygnalizacja otwarcia wagonu 2
%M0009	OTW_1	Sygnalizacja otwarcia wagonu 1
%M0010	ZEZ_OTW	Otworzyć drzwi

Proponowane do wykorzystania bloki funkcyjne:

1. Blok przesłania bitów:



E - enable: umożliwia pracę bloku funkcyjnego

IN - wartość lub adres do przeniesienia

Q- wyjście

OK - potwierdzenie wykonania operacji

Zadanie 6.1 Sterowanie windą dwu poziomową

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	W_DOL	Żądanie jazdy w dół
%I0002	W_GORE	Żądanie jazdy w górę
%Q0005	SIL_GOL	Silnik w dół
%Q0006	SIL_GOR	Silnik do góry
%M0001	LICZ_R2	Liczy R2
%M0002	LICZ_R6	Liczy R6
%M0003	KAS_R6	Kasowanie R6
%M0009	KAS_R2	Kasowanie R2
%M0031	LICZY10	Liczy R10
%M0032	LICZY14	Liczy R14
%M0033	KAS_R14	Kasowanie R14
%M0039	KAS_R10	Kasowanie R10
%M0100	BLOKADA	Winda właśnie jedzie

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:

Opis w zadaniu 3

Zadanie 6.2 Sterowanie windą dwu poziomową z czujnikami

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	W_DOL	Żądanie jazdy w dół
%I0002	W_GORE	Żądanie jazdy w górę
%I0004	STOP	Stop
%I0005	CZU_DOL	Czujnik dojeżdżania w dół
%I0006	CZU_GOR	Czujnik dojeżdżania w górę
%I0007	DOL	Winda dojechała w dół
%I0008	GORA	Winda dojechała do góry
%Q0001	KON_DOL	Kontrolka jazdy w dół
%Q0002	KON_GOR	Kontrolka jazdy w górę
%Q0003	WOL_DOL	Jazda wolna w dół
%Q0004	WOL_GOR	Jazda wolna w górę
%Q0005	SZY_DOL	Jazda szybka w dół
%Q0006	SIL_SZY	Jazda szybka w górę
%M0001	LICZY_2	Liczy R2
%M0002	KASUJ_2	Kasowanie R2
%M0003	LICZY_6	Liczy R6
%M0004	KAS_R6	Kasowanie R6
%M1000	BLOKADA	Blokada żądania

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:

Opis w zadaniu 3

Zadanie 7 Linia napełniania kartonów z zabezpieczeniami

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	KON_PRZ	Przesunięto o jeden w przód
%I0002	JEST	Jest karton
%I0003	PRACA	Praca start
%I0004	STOP	Praca stop
%Q0001	OTW_ZAW	Otwarcie zaworu
%Q0002	PRZESUW	Przesuw taśmy
%Q0007	K_STOP	Kontrolka: STOP
%Q0008	K_START	Kontrolka: START
%M0001	CZEKAJ	Oczekiwanie po przesuwie
%M0002	ZAWOR	Otwórz zawór
%M0010	START	Cykl trwa
%M0011	ZAKONCZ	Zakończ cykl
%M0013	ZBO_STA	Zbocze start
%M0015	ZBO_STO	Zbocze stop
%M0016	KONIEC	Koniec napełniania
%M0017	ZERUJ	Koniec cyklu
%M0019	ODCZYT	Odtwórz stan pracy
%M0020	ZAPIS	Zachowaj stan pracy
%M0021	PRZ_ZOB	Zbocze przesunięcia

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:

Opis w zadaniu 3

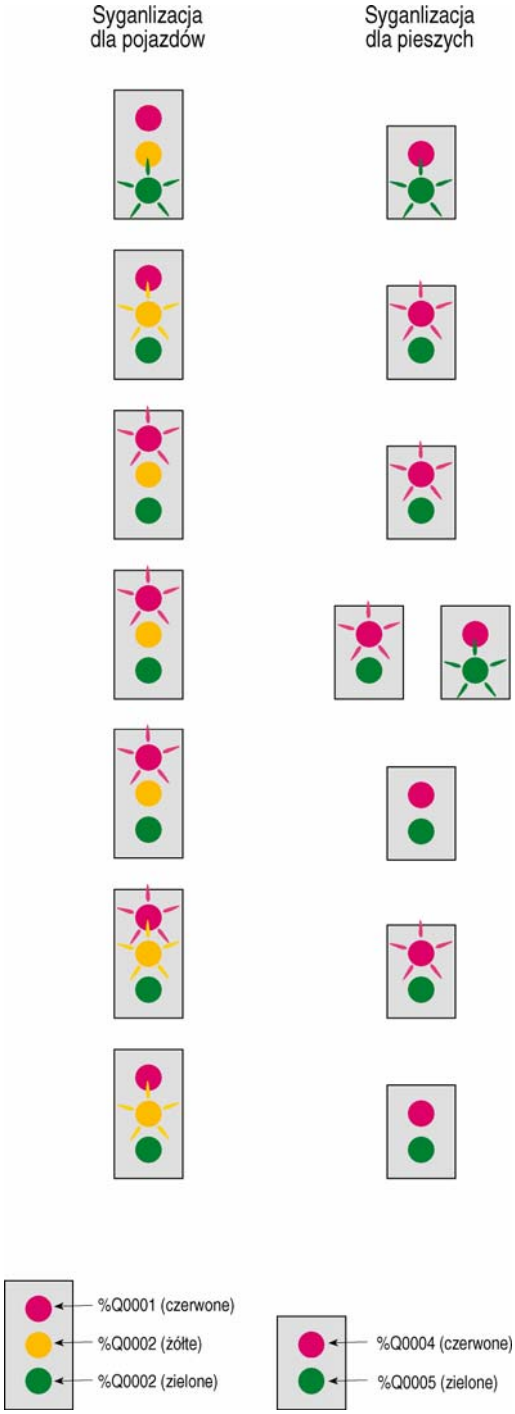
2. Blok przesłania bitów:

Opis w zadaniu 5

3. Wywołanie podprogramu:

E – enable: zezwolenie na wykonanie poleceń

Zadanie 8 Sygnalizacja świetlna



Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	ZAD_PIE	Żadnych pieszych
%I0002	STEROW	Przejście do trybu sterowania
%I0003	PULSUJ	Powrót do trybu pulsowania
%Q0001	POJ_CZE	Światło czerwone dla pojazdów
%Q0002	POJ_ZOL	Światło żółte dla pojazdów
%Q0003	POJ_ZIE	Światło zielone dla pojazdów
%Q0004	PIE_CZE	Światło czerwone dla pieszych
%Q0005	PIE_ZIE	Światło zielone dla pieszych
%Q0006	SLUP	Kontrolka na słupie
%M0001	FLAGA1	Pojazdy: ziel, piesi: czerw
%M0002	FLAGA2	Pojazdy: żółte, piesi: czerw
%M0003	FLAGA3	Pojazdy: czerw, piesi: czerw
%M0004	FLAGA4	Pojazdy: czerw, piesi: ziel
%M0005	FLAGA5	Pojazdy: czerw
%M0006	FLAGA6	Pojazdy: czerw, żółte, piesi: czerwone
%M0007	FLAGA7	Pojazdy: żółte
%M0100	LICZY1	Działa licznik nr 1
%M0101	KASUJ1	Kasowanie licznika nr 1
%M0102	LICZY2	Działa licznik nr 2
%M0103	KASUJ2	Kasowanie licznika nr 2
%M0999	ZAPAL	Zapal kontrolkę na słupie
%M1000	TRYB	Tryb pracy (serow./pulsow.)
%M1001	ZGLOSZ	Piesi zgłosili żądanie
%M1006	LICZ_10	Liczy %R0010
%M1007	KAS_10	Kasowanie %R0010
%M0008	LICZ_14	Liczy %R0014
%M0009	KAS_14	Kasowanie %R0014
%M1010	LICZ_18	Liczy %R0018
%M1011	KAS_18	Kasowanie %R0018
%M1012	MIGANIA	Miganie światła zielonego
%M1013	LICZ_22	Liczy %R0022
%M1014	KAS_22	Kasowanie %R0022
%M1015	LICZ_26	Liczy %R0026
%M1016	KAS_26	Kasowanie %R0026
%M1017	LICZ_30	Liczy %I0030
%M0018	DALEJ	Ciąg dalszy do %I0001+1
%M0019	KAS_30	Kasowanie %R0030
%M1020	KASUJ	Kasowanie rejestrów
%M1021	KAS_R34	Kasowanie %R0034
%M1022	LICZ_34	Liczy %R0034
%M1023	START30	Rozpoczęcie cyklu 30 sekundowego
%M1024	BLOKADA	Blokada 30 sekundowa

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:

Opis w zadaniu 3

Zadanie 9 Regulator PID

Proponowana pełna tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	PRACA	Regulator włączony
%I0002	MANUAL	Sterowanie ręczne
%I0003	GORA	Regulacja w górę
%I0004	DOL	Regulacja w dół
%I0007	ZWIEKSZ	Zwiększ wartość zadaną
%I0008	ZMINEJS	Zmniejsz wartość zadaną
%Q0001	DZIALA	Regulator pracuje
%M0001	STOP_50	Koniec zwiększania R2
%M0002	ZBO_DOL	Zbocze sygnału wpisującego w R2
%M0003	ZBO_GOR	Zbocze sygnału wpisującego w R14
%M0004	NIE_ZMN	Nie zmniejszaj więcej
%R0002	ZADANA	Wartość zadana
%R0006	REGUL	Wartość wielkości regulowanej
%R0010	STERUJ	Wartość sterująca
%R0014	POMOC	Rejestr pomocniczy
%R0018	REJ_PID	Rejestr PID

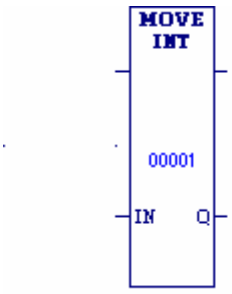
Uwaga: aby zrealizować zadanie w praktyce, wielkości regulowanej powinien zostać przypisany rejestr wejścia analogowego (%AI), przez które realizowany jest pomiar wielkości regulowanej (temperatury), a wartości sterującej rejestr wyjścia analogowego (%AQ), służącego do sterowania mocą grzałki. Do realizacji zadania konieczny jest sterownik 90-30.

Proponowane do wykorzystania bloki funkcyjne:

1. Licznik zliczający w górę:

Opis w zadaniu 3

2. Blok przemieszczania liczb całkowitych:



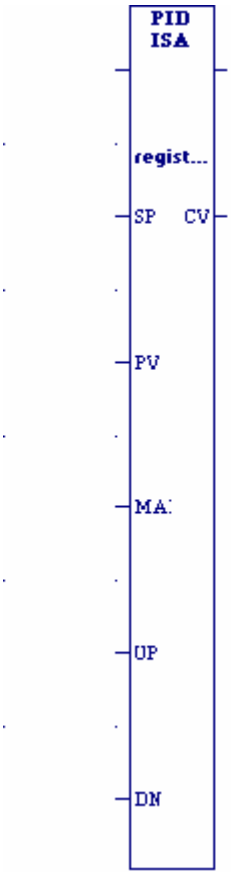
E – enable: zezwolenie na wykonanie operacji

IN – wartość stała lub adres zmiennej, której wartość ma być przemieszczona

Ok – sygnał potwierdzenia wykonania operacji

Q – miejsce, do którego ma się odbyć kopiowanie

3. Regulator PID:



E – enable: zezwolenie na wykonanie operacji

SP – punkt pracy regulatora

PV – wielkość regulowana

MAN – wejście przełączania w ręczny tryb pracy

UP – zwiększenie sygnału sterującego (tylko podczas ręcznego trybu pracy)

DN – zmniejszenie sygnału sterującego (tylko podczas ręcznego trybu pracy)

Register – adres pierwszego z 40 rejestrów, w których przechowywane są parametry regulatora

Ok – sygnał potwierdzenia zrealizowania algorytmu bez przeszkód

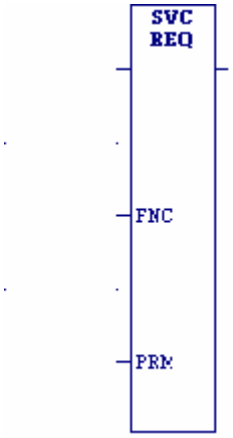
CV – wartość sygnału sterującego

Zadanie 10 Odczyt daty i czasu z zegara kalendarzowego w sterowniku

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%T0001	POTWIE	Odczyt zegara OK
%R0001	TRYB	0 = odczyt, 1 = zapis
%R0002	ZAW_1	Zawsze 1

Należy użyć blok funkcyjny SVCREQ i wpisać numer funkcji 7:



E – enable: zezwolenie na wykonanie operacji

Ok – potwierdzenie poprawnie wykonanej operacji przez blok funkcyjny

FNC – na wejściu tym określamy numer funkcji realizowanej przez blok SVCREQ. Dla operacji odczytu lub zapisu zegara w sterowniku należy wpisać 7

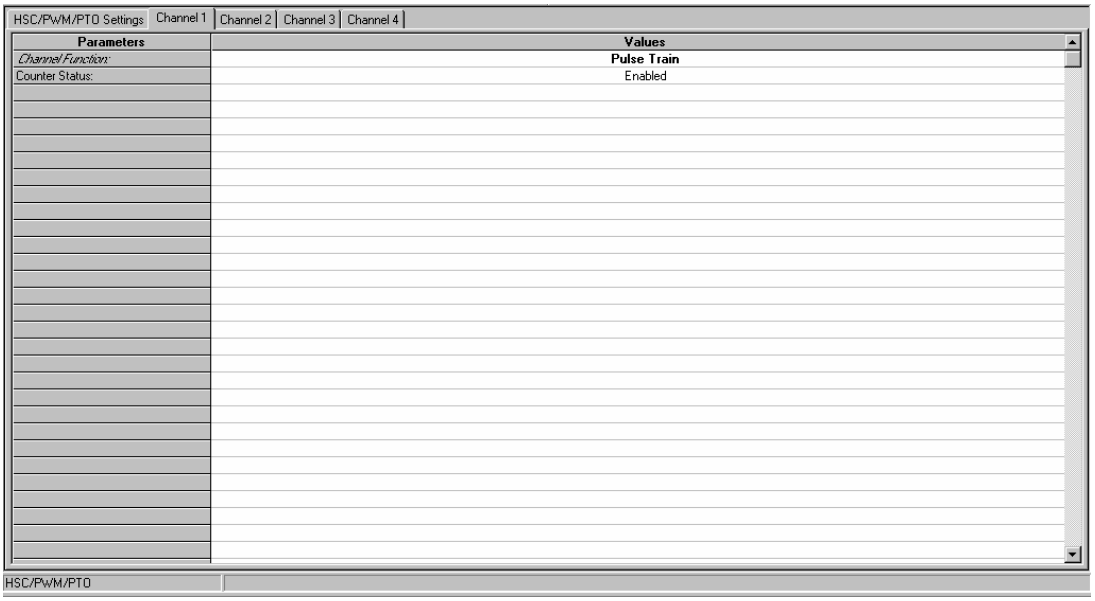
PRM – zawiera adres początkowy bloku parametrów funkcji określonej przez FNC

Funkcja SVCREQ o numerze 7 może dokonać odczytu lub zmiany ustawienia zegara w sterowniku podtrzymującego aktualny czas i datę. Funkcja SVCREQ nr 7 może operować na 6 rejestrach. Wartość 0 w pierwszym rejestrze wykorzystywanym przez tą funkcję oznacza, że dokonywany będzie odczyt bieżącego czasu i daty, wartość 1 - oznacza ustawienie nowego czasu i daty. Wpisanie wartości 1 do drugiego rejestru wykorzystywanego przez funkcję SVCREQ oznacza że dane będą w formacie BCD. Przy odczycie zegara rejestry od 1 do 6 będą zawierały następujące informacje:

	bajt wyższy	bajt niższy
rejestr pierwszy	0	
rejestr drugi	1	
rejestr trzeci	miesiąc	rok
rejestr czwarty	godzina	dzień
rejestr piąty	sekunda	minuta
rejestr szósty	dzień tygodnia	niewykorzystany (zera)

Zadanie 11 Sterownie silnikami krokowymi

Zakładamy, że sterowanie silnikiem krokowym odbywać się będzie za pomocą wyjścia Q1 (wyjście tranzystorowe). Aby uaktywnić pracę tego wyjścia jako generatora fali prostokątnej należy odpowiednio skonfigurować sterownik, np.:



Następnie konieczne jest załadowanie wartości do rejestrów AQ123 (częstotliwość z zakresu 15...5000Hz) oraz AQ124 (ilość impulsów z zakresu 0...65535) oraz uruchomienie generatora fali prostokątnej wyjściemQ494.

Proponowana tablica deklaracji zmiennych:

Ref Address	Name	Description
%I0001	WEJ1	Sygnał startu od operatora
%Q0494	START	Wygeneruj impuls
%T0001	ZBOCZE	Zmienna pomocnicza
%AQ123	CZESTOT	Częstotliwość impulsów
%AQ124	ILOSC	Ilość impulsów

Zadanie 12 Komunikacja w protokole SNP

- w programie należy wykorzystać funkcję COMREQ
- w celu odczytania wyjść należy użyć komend: Attach oraz Read System Memory
- w celu zapisania wejść należy użyć komend: Attach oraz Write System Memory
- program należy napisać w ten sposób aby odczyt i zapis następowały cyklicznie po sobie. Warunkiem rozpoczęcia kolejnego cyklu jest pojawienie się wartości 1 w rejestrze statusowym komunikacji
- program należy napisać w ten sposób, aby po wyłączeniu i ponownym włączeniu zasilania, komunikacja pomiędzy sterownikami została nawiązana automatycznie

Proponowana tablica zmiennych:

Ref Address	Name	Description
%R0301	STATUS	Status komunikacji
%R0302	OPOZN	Opóźnienie komunikacji
%R0305	DANE1	Pierwsze słowo bloku danych
%R0312	DANE8	Ósme słowo bloku danych
%R0325	LICZNIK	Wartość licznika
%M0801	RESET	Reset licznika
%T0201	START	Załączenie timera
%T0202	KOM_1SK	Nawiązanie komunikacji w 1 skanie programu
%T0203	BLAD	Błąd komunikacji
%T0204	KOMUNIK	Nawiązanie komunikacji
%T0205	ZM_POM	Zmienna pomocnicza
%T0206	PO_AW	Nawiązanie komunikacji po awarii

Proponowane do wykorzystania bloki funkcyjne:

1. Blok przemieszczenia liczb całkowitych:

Opis w zadaniu 9

2. Przekaźnik czasowy bez pamięci:

Opis w przykładzie 2

3. Relacja matematyczna równości:

Opis w zadaniu 4

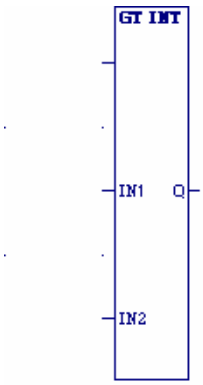
4. Blok przemieszczenia siedmiu stałych wartości:

Opis w przykładzie 18

5. Licznik zliczający w górę:

Opis w zadaniu 2.1

6. Relacja matematyczna większości



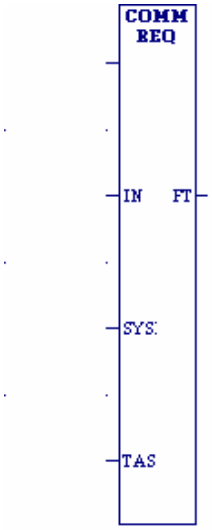
E – enable: zezwolenie na wykonanie operacji

I1 – parametr wejściowy pierwszy

I2 – parametr wejściowy drugi

Q – sygnał wyjściowy, pojawiający się, gdy I1 i I2 spełniają relację

1. Funkcja umożliwiająca nawiązanie komunikacji



E – enable: zezwolenie na wykonanie operacji

IN – adres pierwszego słowa bloku danych

SYS – numer kasety i numer gniazda, w którym jest zainstalowany moduł, z którym ma zostać nawiązana komunikacja

TAS – numer portu: 19 (0013H)-port pierwszy
20 (0014H)-port drugi

FT – sygnał wyjściowy, pojawiający się, gdy nawiązanie komunikacji się nie powiedzie

Opis parametrów bloku danych BLKMOV dla funkcji COMREQ

Attach: 07200 (1C20)

Słowo	Definicja	Wartość
adres startowy	Długość bloku danych	7
adres startowy+1	Flaga WAIT/NOWAIT	0=NOWAIT
adres startowy+2	Typ zmiennej* słowa statusowego	np. 8=%R
adres startowy+3	Adres słowa statusowego minus 1	np. 99
adres startowy+4	nie używane	0
adres startowy+5	nie używane	0
adres startowy+6	Numer komendy	7200
adres startowy+7	1 i 2 bajt ID slve'a	wartość hex
adres startowy+8	3 i 4 bajt ID slave'a	wartość hex
adres startowy+9	5 i 6 bajt ID slave'a	wartość hex
adres startowy+10	7 i 8 bajt ID slave'a	0
adres startowy+11	Typ zmiennej, w której przechowywane są informacje zwrotne od urządzenia slave	np. 8=%R
adres startowy+12	Adres, pod którym przechowywane są informacje od urządzenia slave w pamięci urządzenia master	np. 170

Read System Memory: 07202 (1C22)

Słowo	Definicja	Wartość
adres startowy	Długość bloku danych	6
adres startowy+1	Flaga WAIT/NOWAIT	0=NOWAIT
adres startowy+2	Typ zmiennej* słowa statusowego	np. 8=%R
adres startowy+3	Adres słowa statusowego minus 1	np. 300
adres startowy+4	nie używane	0
adres startowy+5	nie używane	0
adres startowy+6	Numer komendy	7202
adres startowy+7	Typ zmiennej* odczytywanej z urządzenia slave	np. 70=%I
adres startowy+8	Adres zmiennej odczytywanej z urządzenia slave	np. 1
adres startowy+9	Ilość zmiennych odczytywanych z urządzenia slave	np. 8
adres startowy+10	Typ zmiennej*, pod jakim ma być umieszczona odczytana zmienna z urządzenia slave	np. 70=%I
adres startowy+11	Adres, pod jakim ma być umieszczona zmienna odczytana z urządzenia slave	np. 17

Write System Memory: 07203 (1C23)

Słowo	Definicja	Wartość
adres startowy	Długość bloku danych	6
adres startowy+1	Flaga WAIT/NOWAIT	0=NOWAIT
adres startowy+2	Typ zmiennej* słowa statusowego	np. 8=%R
adres startowy+3	Adres słowa statusowego minus 1	np. 300
adres startowy+4	nie używane	0
adres startowy+5	nie używane	0
adres startowy+6	Numer komendy	7203
adres startowy+7	Typ zmiennej*, pod jakim ma być umieszczona zmienna w urządzeniu slave	np. 72=%Q
adres startowy+8	Adres, pod jakim ma być umieszczona zmienna w urządzeniu slave	np. 1
adres startowy+9	Ilość zmiennych zapisywanych do urządzenia slave	np. 6
adres startowy+10	Typ zmiennej* zapisywanej do urządzenia slave	np. 72=%Q
adres startowy+11	Adres zmiennej zapisywanej do urządzenia slave	np. 100

*Oznaczenie typów zmiennych:

Wartość	Opis
70	%I
72	%Q
74	%T
76	%M
8	%R
10	%AI
12	%AQ

Szczegółowe informacje na temat komunikacji, przy użyciu protokołu SNP można znaleźć w książce "**Series 90 PLC Serial Communications**" (GFK-0582).

Zadanie 13 Komunikacja w protokole SNP-X

- w programie należy wykorzystać funkcję COMREQ
- w celu odczytania rejestrów należy użyć komendy X-Read
- program należy napisać w ten sposób aby odczyt i zapis następowały cyklicznie po sobie. Warunkiem rozpoczęcia kolejnego cyklu jest pojawienie się wartości 1 w rejestrze statusowym komunikacji
- sterowniki podrzędne posiadają numery identyfikacyjne: 111111 (12593H) i 222222 (12850H)
- moduł komunikacyjny sterownika nadrzędnego umieszczony jest w 3 slocie kasety podstawowej

Proponowana tablica zmiennych:

Ref Address	Name	Description
%R0001	STATUS	Status komunikacji
%R0002	OPOZN	Opóźnienie komunikacji
%R0005	DANE1	Pierwsze słowo bloku danych
%R0012	DANE8	Ósme słowo bloku danych
%R0019	DANE15	Piętnaste słowo bloku danych
%R0026	DANE22	Dwudzieste drugie słowo bloku danych
%R0050	LICZNIK	Wartość licznika
%M0001	RESET	Reset licznika
%T0001	START	Załączenie timera
%T0002	KOM_1SK	Nawiązanie komunikacji w 1 skanie programu
%T0003	BLAD	Błąd komunikacji
%T0004	KOMUNIK	Nawiązanie komunikacji
%T0005	ZM_POM1	Zmienna pomocnicza 1
%T0006	ZM_POM2	Zmienna pomocnicza 2

Proponowane do zadania bloki funkcyjne:

1. Blok przemieszczenia liczb całkowitych:

Opis w zadaniu 9

2. Przekaźnik czasowy bez pamięci:

Opis w przykładzie 2

3. Relacja matematyczna równości:

Opis w zadaniu 4

4. Blok przemieszczenia siedmiu stałych wartości:

Opis w przykładzie 18

5. Licznik zliczający w górę:

Opis w zadaniu 2.1

6. Funkcja umożliwiająca nawiązanie komunikacji

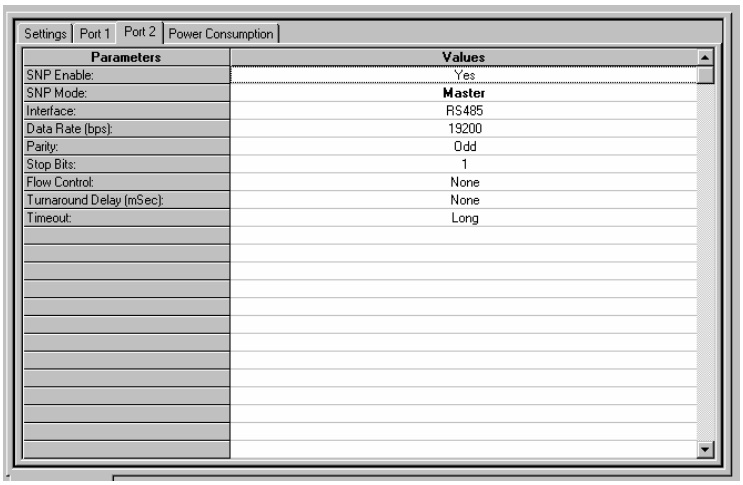
Opis w zadaniu 12

Opis parametrów komendy X-Read 07101 (1BBD)

Słowo	Definicja	Wartość
adres startowy	Długość bloku danych	17
adres startowy+1	Flaga WAIT/NOWAIT	0=NOWAIT
adres startowy+2	Typ zmiennej* słowa statusowego	np. 8=%R
adres startowy+3	Adres słowa statusowego minus 1	np. 0
adres startowy+4	nie używane	0
adres startowy+5	nie używane	0
adres startowy+6	Numer komendy	7101
adres startowy+7	1 i 2 bajt ID slve'a	12593 / 12850
adres startowy+8	3 i 4 bajt ID slave'a	12593 / 12850
adres startowy+9	5 i 6 bajt ID slave'a	12593 / 12850
adres startowy+10	7 i 8 bajt ID slave'a	0
adres startowy+11	Typ komunikacji	0=single-session, 1=multi-session
adres startowy+12	Typ zmiennej* odczytywanej z urządzenia slave	np. 8=%R
adres startowy+13	Adres zmiennej odczytywanej z urządzenia slave	(w programie - 1)
adres startowy+14	Ilość zmiennych odczytywanych z urządzenia slave	(w programie -1)
adres startowy+15	Typ zmiennej*, pod jakim ma być umieszczona zmienna odczytana z urządzenia slave	np. 8=%R
adres startowy+16	Adres, pod jakim ma być umieszczona zmienna odczytana z urządzenia slave	(w programie - 100 i 120 - odpowiednio dla 1 i 2 slave'a)
adres startowy+17	Typ zmiennej*, pod jakim ma być umieszczone w urządzeniu master słowo statusowe z urządzenia slave	np. 8=%R
adres startowy+18	Adres, pod jakim słowo statusowe z urządzenia slave ma być umieszczone w rządzeniu master	(w programie - 0)
adres startowy+19	Response Timeout	0
adres startowy+20	Broadcast Delay	0
adres startowy+21	Modem Turnaround Time	0
adres startowy+22	Transmission Delay	0

* Typy zmiennych i parametry funkcji COMREQ – patrz: zadanie 12.

Przykładowa konfiguracja urządzenia Master i Slave



Szczegółowe informacje na temat komunikacji, przy użyciu protokołu SNP-X można znaleźć w książce “Series 90 PLC Serial Communications” (GFK-0582).

Zadanie 14 Przesyłanie danych przez port szeregowy

W zadaniu należy skorzystać z bloku funkcyjnego COMREQ. Blok ten należy odpowiednio skonfigurować (por. np z dokumentacją gfk-1645), podając m.in. kody heksadecymalne wysyłanych znaków:

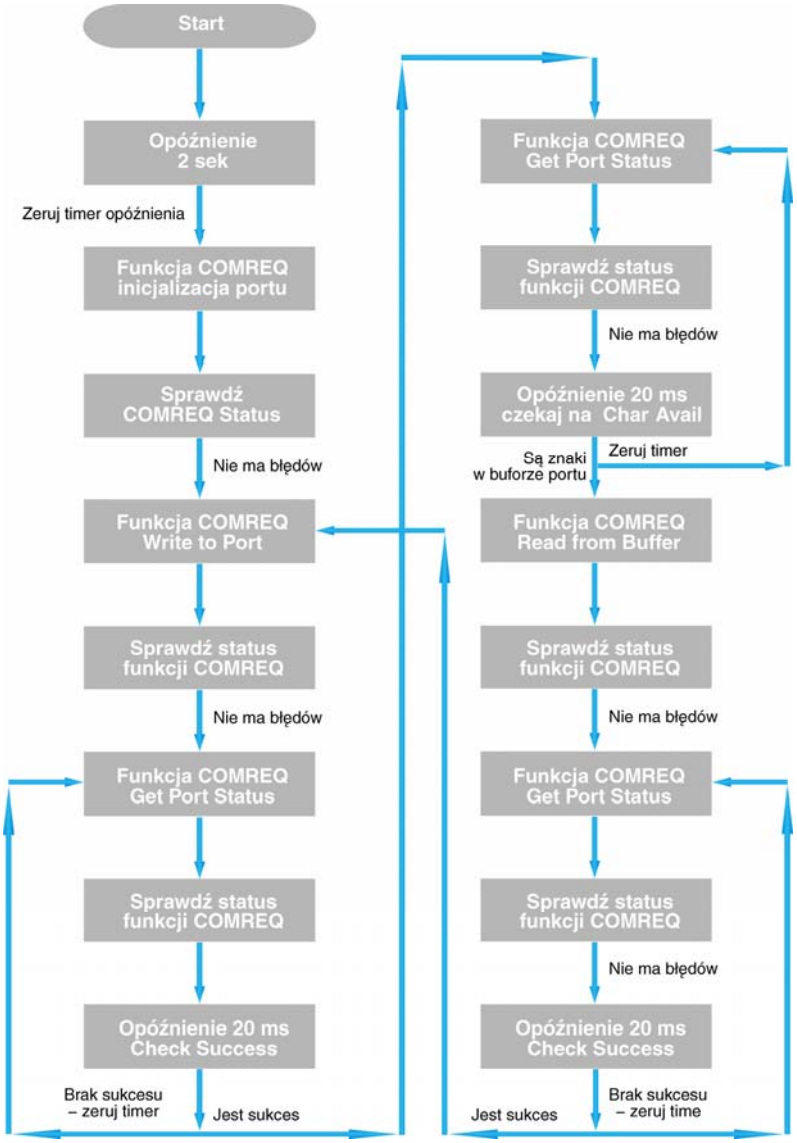
A – 41H

B – 42H

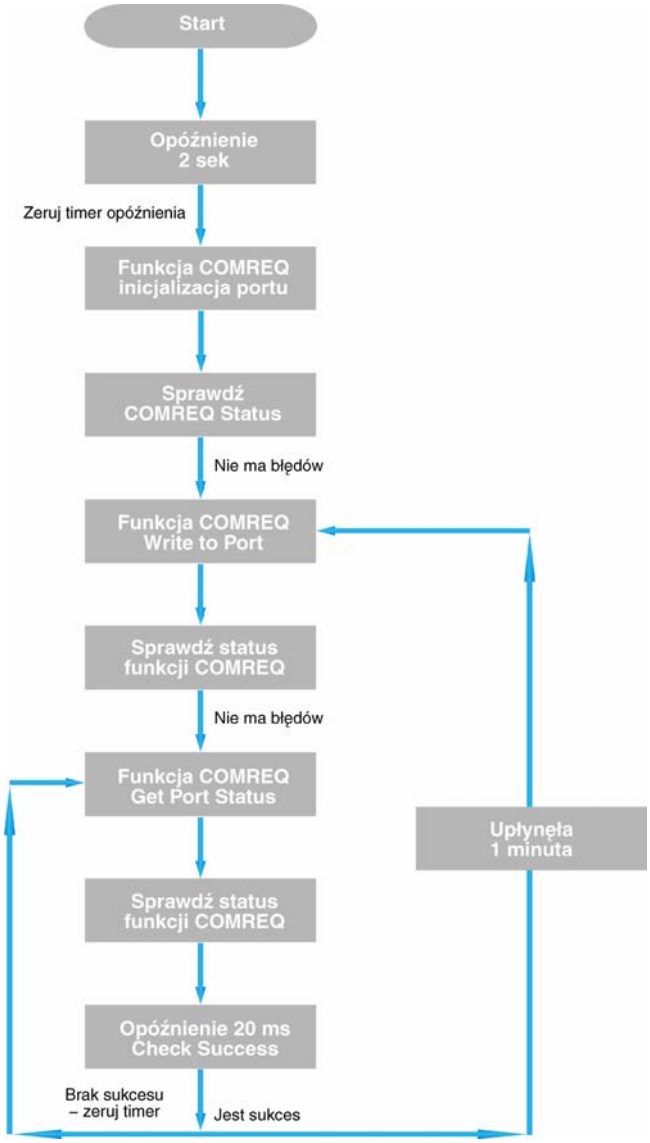
C – 43H

D – 44H

Ogólnie schemat blokowy wysyłania/czytania znaków ASCII poprzez port szeregowy wygląda następująco:


Start
Opóźnienie 2 sek
Zeruj timer opóźnienia
Funkcja COMREQ inicjalizacja portu
Sprawdź COMREQ Status
Nie ma błędów
Funkcja COMREQ Write to Port
Sprawdź status funkcji COMREQ
Nie ma błędów
Funkcja COMREQ Get Port Status
Sprawdź status funkcji COMREQ
Opóźnienie 20 ms Check Success
Brak sukcesu – zeruj timer
Jest sukces
Funkcja COMREQ Get Port Status
Sprawdź status funkcji COMREQ
Nie ma błędów
Opóźnienie 20 ms czekaj na Char Avail
Są znaki w buforze portu
Zeruj timer
Funkcja COMREQ Read from Buffer
Sprawdź status funkcji COMREQ
Nie ma błędów
Funkcja COMREQ Get Port Status
Sprawdź status funkcji COMREQ
Nie ma błędów
Opóźnienie 20 ms Check Success
Jest sukces
Brak sukcesu – zeruj time

Dla realizacji niniejszego zadania schemat ten jednak został nieco uproszczony (pomijamy fazy czyszczenia znaków z portu szeregowego):



Sterownik realizuje następujące zadania:

- opóźnienie 2 sekundowe, zalecane po załączeniu zasilania sterownika,
- inicjalizacja portu za pomocą funkcji COMREQ,
- sprawdzenie statusu realizacji zadania przez blok COMREQ,
- sprawdzenie statusu portu szeregowego za pomocą funkcji COMREQ,
- sprawdzenie statusu realizacji zadania przez blok COMREQ,
- odmierzanie czasu 1 minuty, po którym nastąpi ponowne wysłanie ciągu znaków ASCII.

Jak widać ze schematu blokowego, każdorazowe wywołanie bloku COMREQ pociąga za sobą konieczność sprawdzania statusu realizacji zadania przez blok COMREQ.

Blok funkcyjny CPMREQ jest blokiem uniwersalnym, realizuje on taką funkcję (definiowaną przez numer komendy), jaka zostanie podana w bloku rejestrów przyporządkowanych do wywoływanego bloku COMREQ. Poniżej znajduje się skrótowy opis wybranych funkcji dla bloku COMREQ, dokładniejszy opis znajduje się w dokumentacji np. GFK-1645, GFK-1503, GFK-0582D.

Funkcja do inicjalizowania portu (4300)

Funkcja ta powoduje wysłanie komendy zerowania do określonego portu. Powoduje ona również przerwanie wszystkich wykonywanych w danym momencie funkcji COMREQ oraz zeruje wewnętrzny bufor wejściowy.

Przykład bloku danych funkcji do inicjalizowania portu.

	Wartość (dziesiętna)	Wartość (heksadecymalna)	Znaczenie
Adres	0001	0001	Długość bloku danych
adres + 1	0000	0000	Tryb NOWAIT
adres + 2	0008	0008	Typ pamięci słowa statusu (%R)
adres + 3	0000	0000	Adres słowa statusu minus 1 (%R0001)
adres + 4	0000	0000	Nie wykorzystywane
adres + 5	0000	0000	Nie wykorzystywane
adres + 6	4300	10CC	Polecenie inicjalizacji portu

Uwaga:Słowa statusowe poleceń COMREQ przerwanych na skutek wykonania tego polecenia nie są aktualizowane.

Ostrzeżenie: Jeżeli polecenie to zostanie wysłane w czasie, gdy polecenie COMREQ do zapisu bajtów (4401) wysyła ciąg przez port szeregowy, transmisja jest wstrzymywana. Miejsce, w którym nastąpiło przerwanie wysyłania ciągu znaków jest nieokreślone. Dodatkowo, ostatni znak odbierany przez urządzenie komunikujące się z jednostką centralną jest również nieokreślony

Funkcja do odczytu statusu portu (4303)

Funkcja ta zwraca bieżący status portu. Wykrywane są następujące zdarzenia:

1. Poprzednio zainicjowano żądanie odczytu, żądana liczba znaków została odczytana lub upłynął maksymalny czas oczekiwania.
2. Poprzednio zainicjowano żądanie zapisu i przesłano odpowiednią liczbę znaków lub nastąpiło przeterminowanie.

Status zwrócony przez funkcję informuje o zaistniałym zdarzeniu (lub zdarzeniach). Jednocześnie może zaistnieć więcej niż jedno zdarzenie, jeżeli poprzednio zainicjowano zarówno polecenie do odczytu, jak i do zapisu.

Przykład bloku danych funkcji do odczytu statusu portu

	Wartość (dziesiętna)	Wartość (heksadecymalna)	Znaczenie
adres	0003	0003	Długość bloku danych
adres + 1	0000	0000	Tryb NOWAIT
adres + 2	0008	0008	Typ pamięci słowa statusu (%R)
adres + 3	0000	0000	Adres słowa statusu minus 1 (%R0001)
adres + 4	0000	0000	Nie wykorzystywane
adres + 5	0000	0000	Nie wykorzystywane
adres + 6	4303	10CF	Funkcja do odczytu statusu portu
adres + 7	0076	004C	Typ pamięci statusu portu (%M)
adres + 8	0101	0065	Przesunięcie w pamięci statusu portu (%M101)

Status portu

Status portu składa się ze słowa statusu oraz liczby znaków w buforze wejściowym, które nie zostały odczytane przez program sterujący (znaki odebrane, które można odczytać).

słowo 1	Słowo statusu portu (opis zamieszczono poniżej)
słowo 2	Znaki umieszczone w buforze wejściowym

Dostępne są następujące słowa statusu portu:

Bit	Nazwa	Definicja	Znaczenie	
15	RI	Trwanie odczytu	1	Dotyczy odczytu bajtów lub odczytu ciągu znaków
			0	Przeterminowanie poprzedniego polecenia odczytu bajtów lub odczytu ciągów, anulowanie lub zakończenie
14	RS	Pomyślne zakończenie odczytu	1	Pomyślne zakończenie odczytu bajtów lub odczytu ciągów
			0	Dotyczy odczytu nowych bajtów lub odczytu ciągów znaków
13	RT	Przeterminowanie	1	Wystąpienie przeterminowania w czasie odczytu bajtów lub odczytu ciągów
			0	Dotyczy odczytu nowych bajtów lub odczytu ciągów znaków
12	WI	Trwanie zapisu	1	Dotyczy zapisu nowych bajtów
			0	Poprzednie wywoływane polecenie zapisu bajtów przeterminowane, anulowane lub zakończone
11	WS	Pomyślne zakończenie operacji zapisu	1	Zakończenie poprzednio wywołanego polecenia zapisu bajtów.
			0	Dotyczy zapisu nowych bajtów
10	WT	Przeterminowanie operacji zapisu	1	Wystąpienie przeterminowania w czasie zapisu bajtów.
			0	Dotyczy zapisu nowych bajtów
9	CA	Dostępne znaki	1	Znaki nieodczytane z bufora.
			0	Brak w buforze nieodczytanych znaków.
8	OF	Błąd przepełnienia	1	Wystąpienie przepełnienia wewnętrznego bufora lub portu szeregowego.
			0	Dotyczy odczytu statusu portu
7	FE	Błąd ramki	1	Wystąpienie błędu ramki w porcie szeregowym
			0	Dotyczy odczytu statusu portu
6	PE	Błąd parzystości	1	Wystąpienie błędu parzystości w porcie szeregowym
			0	Dotyczy odczytu statusu portu
5	CT	Aktywny sygnał CTS	1	Aktywna linia CTS portu szeregowego lub port szeregowy nie posiada linii CTS
			0	Linia CTS portu szeregowego nie aktywna
4 - 0	U	nie wykorzystywany, powinien być równy 0		

Funkcja do zapisu bajtów (4401)

Funkcja ta umożliwia przesłanie jednego lub więcej bajtów do urządzenia zewnętrznego za pomocą określonego portu szeregowego. Znak (znaki) do przesłania muszą znajdować się w pamięci słów (obszar typu %R). Nie powinny one być zmieniane do momentu zakończenia wykonywania funkcji.

Pojedyncze wywołanie tej funkcji pozwala na przesłanie do 250 znaków. Operacja ta jest kończona dopiero w momencie wysłania wszystkich znaków lub w przypadku wystąpienia przeterminowania (przykładowo, w przypadku sprzętowego sterowania przepływem, jeżeli urządzenie zewnętrzne w ogóle nie zezwala na transmisję).

Przykład bloku danych funkcji do zapisu bajtów

	Wartość (dziesiętna)	Wartość (heksadecymalna)	Znaczenie
adres	0006	0006	Długość bloku danych (wraz z wysyłanymi znakami)
adres + 1	0000	0000	Tryb NOWAIT
adres + 2	0008	0008	Typ pamięci słowa statusu (%R)
adres + 3	0000	0000	Adres słowa statusu minus 1 (%R0001)
adres + 4	0000	0000	Nie wykorzystywane
adres + 5	0000	0000	Nie wykorzystywane
adres + 6	4401	1131	Funkcja do zapisu bajtów
adres + 7	0030	001E	Przeterminowanie transmisji (30 sekund). Porównać z uwagą zamieszczoną poniżej.
adres + 8	0005	0005	Liczba bajtów do zapisu
adres + 9	25960	6568	‘h’ (68h), ‘e’ (65h)
adres + 10	27756	6C6C	‘l’ (6Ch), ‘l’ (6Ch)
adres + 11	0111	006F	‘o’ (6Fh)

Pomimo wykorzystywania w niniejszym przykładzie drukowalnych znaków ASCII, nie ma żadnych ograniczeń, jeżeli chodzi o znaki, które można przesyłać.

Uwaga: W przypadku wprowadzenia przeterminowania o wartości równej zero, przeterminowanie będzie równe czasowi potrzebnemu na wysłanie danych plus 4 sekundy.

Ostrzeżenie: Jeżeli w czasie wysyłania ciągu bajtów przez tę funkcję wywołana zostanie funkcja COMREQ do inicjalizowania portu (4300) albo funkcja przerywania aktywnego polecenia (4399) wszystkich poleceń lub poleceń zapisu, spowoduje to przerwanie transmisji. Miejsce, w którym nastąpiło przerwanie wysyłania ciągu jest nieokreślone. Dodatkowo, ostatni znak odbierany przez urządzenie komunikujące się z jednostką centralną jest również nieokreślony.

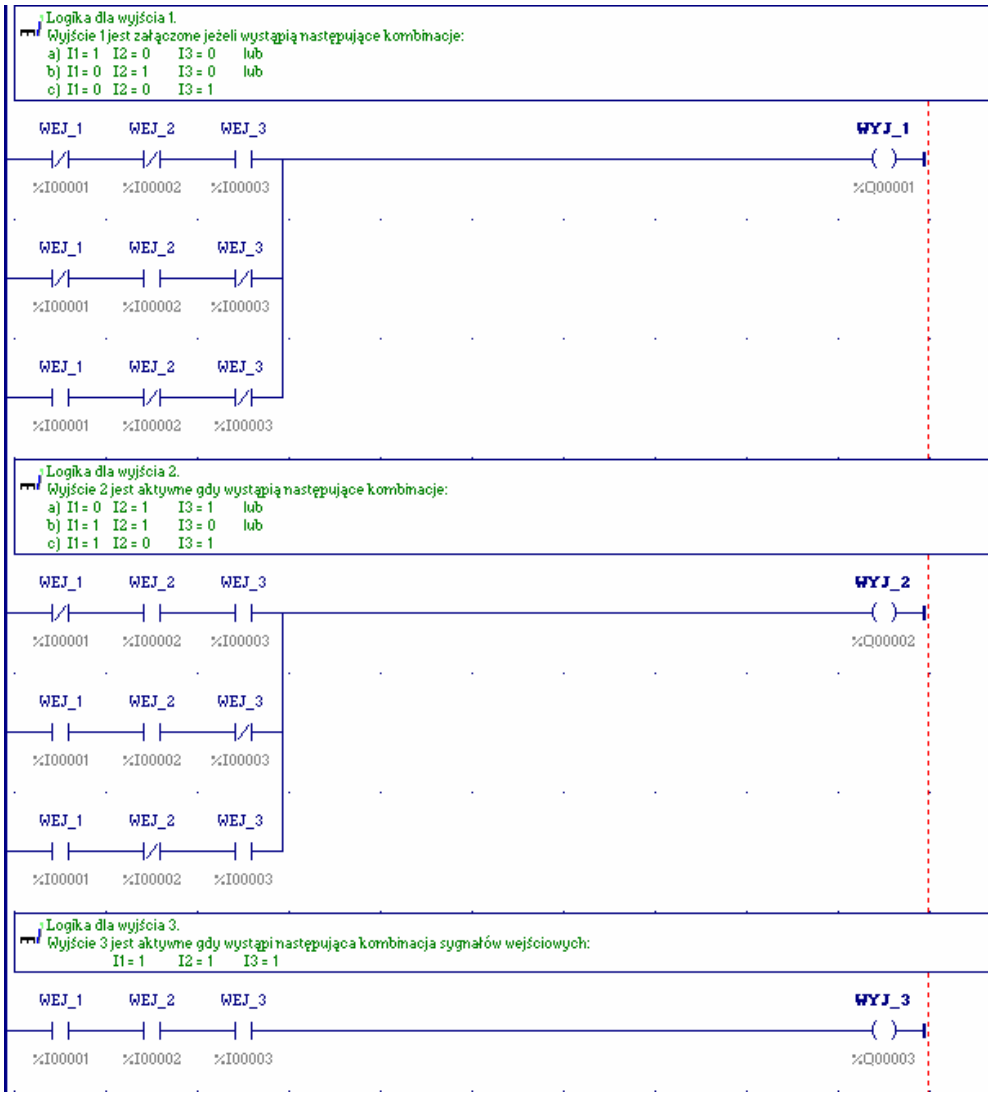
W niniejszym przykładzie skonfigurowano 2 słowa (4 bajty = 4 znaki) danych do wysłania, adres początku grupy rejestrów w których podajemy kody heksadecymalne znaków do wysłania: %R152.

Należy pamiętać o skonfigurowaniu odpowiedniego portu do pracy w protokole Serial I/O (w sterownikach VersaMax Micro 14-pt jest to port 1, w sterownikach 23- i 28-pt jest to port 2, w sterownikach serii VersaMax oraz w CPU363 może to być zarówno port 1 jak i port 2).

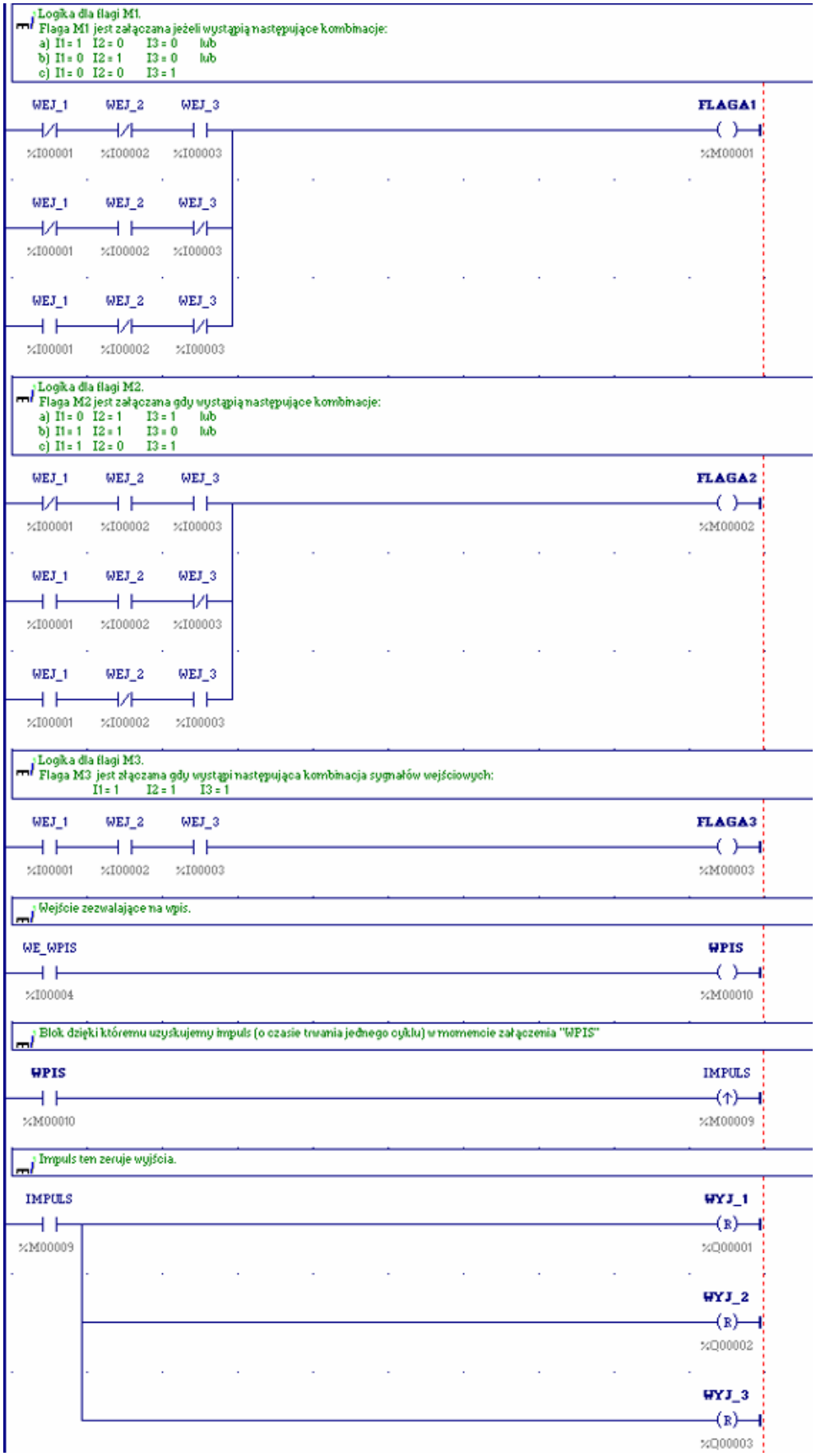
Przykłady rozwiązań

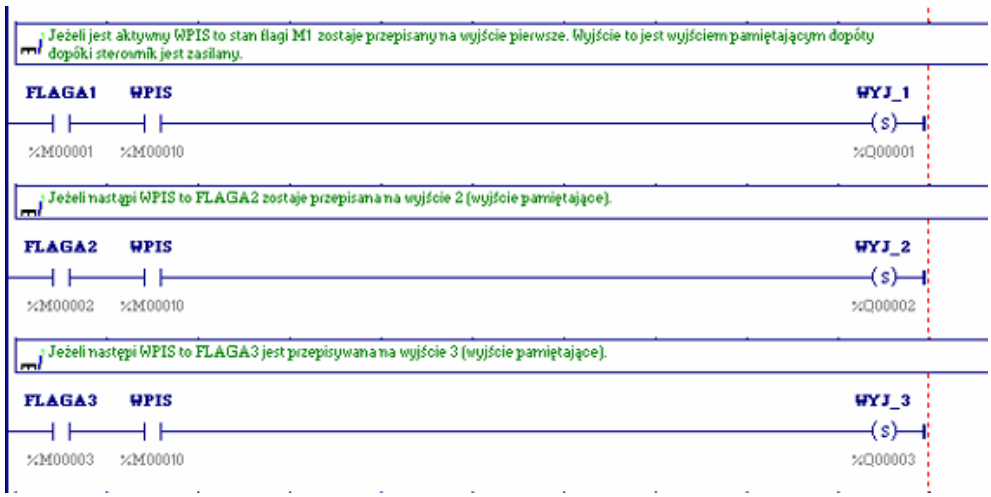
(Uwaga: komentarze odnoszą się do szczebli bezpośrednio po nich następujących.)

Zadanie 1.1 Transkoder



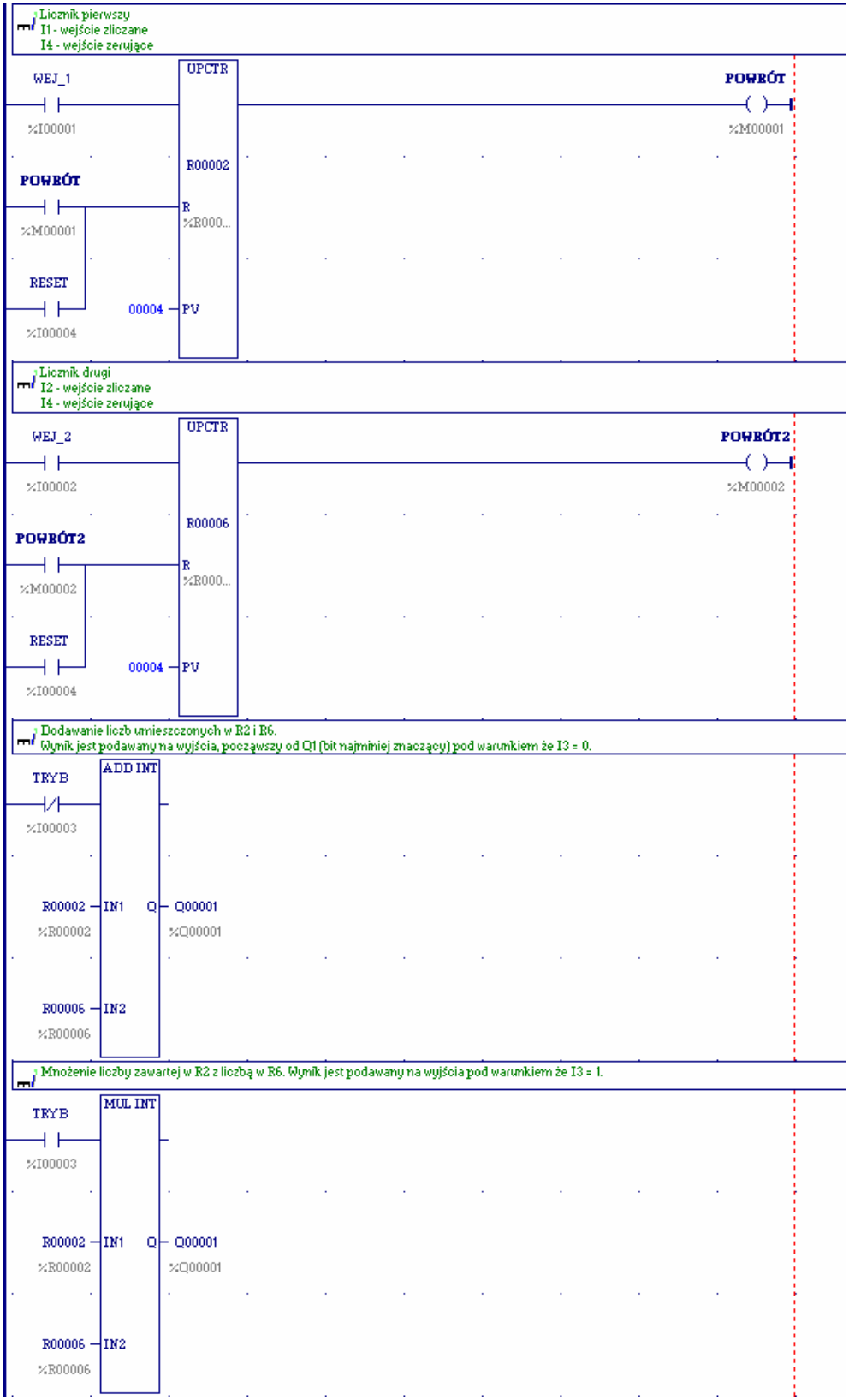
Zadanie 1.2 Transkoder strobowany



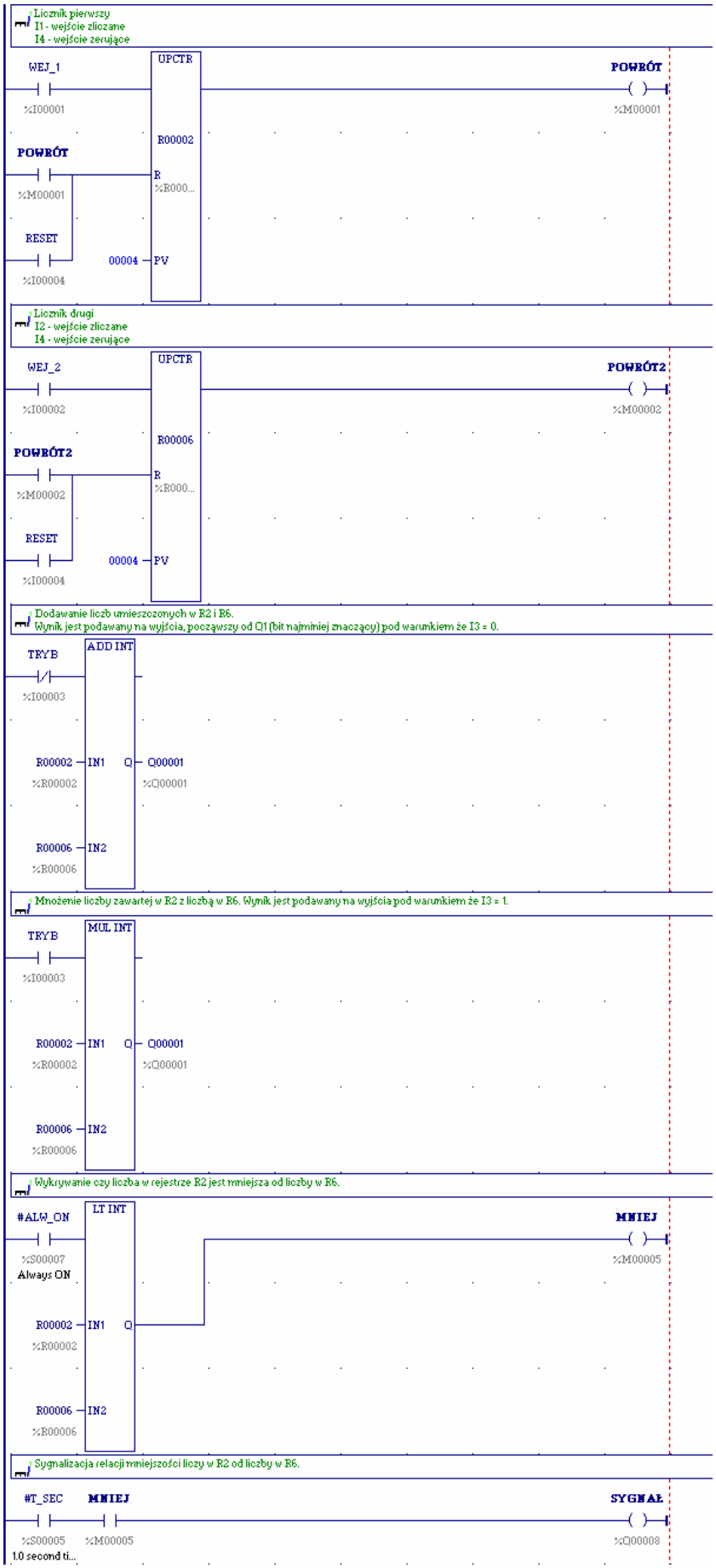


Zadanie 2.1 Licznik modulo 3

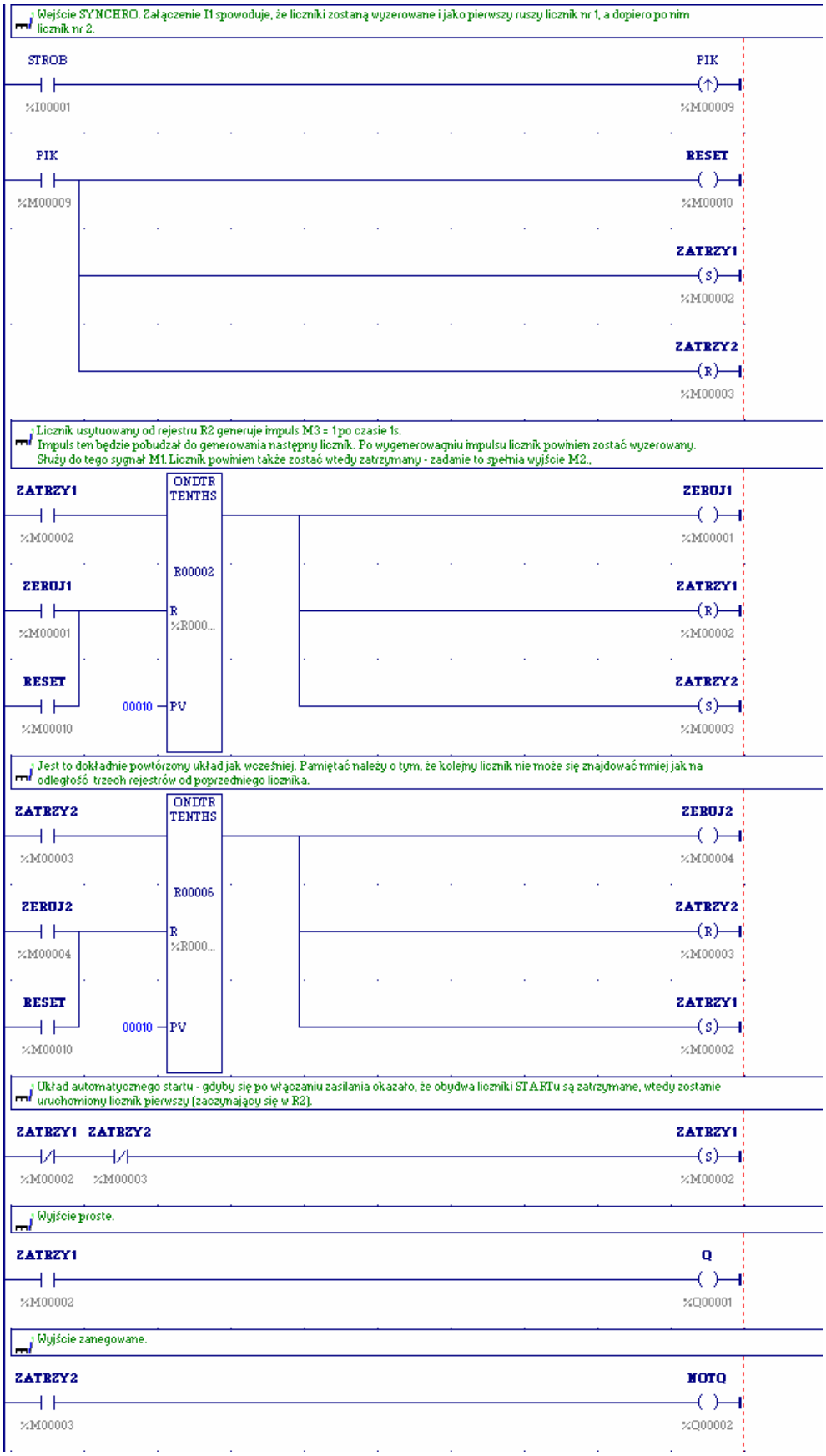
Zadanie 2.2 Dodawanie i mnożenie



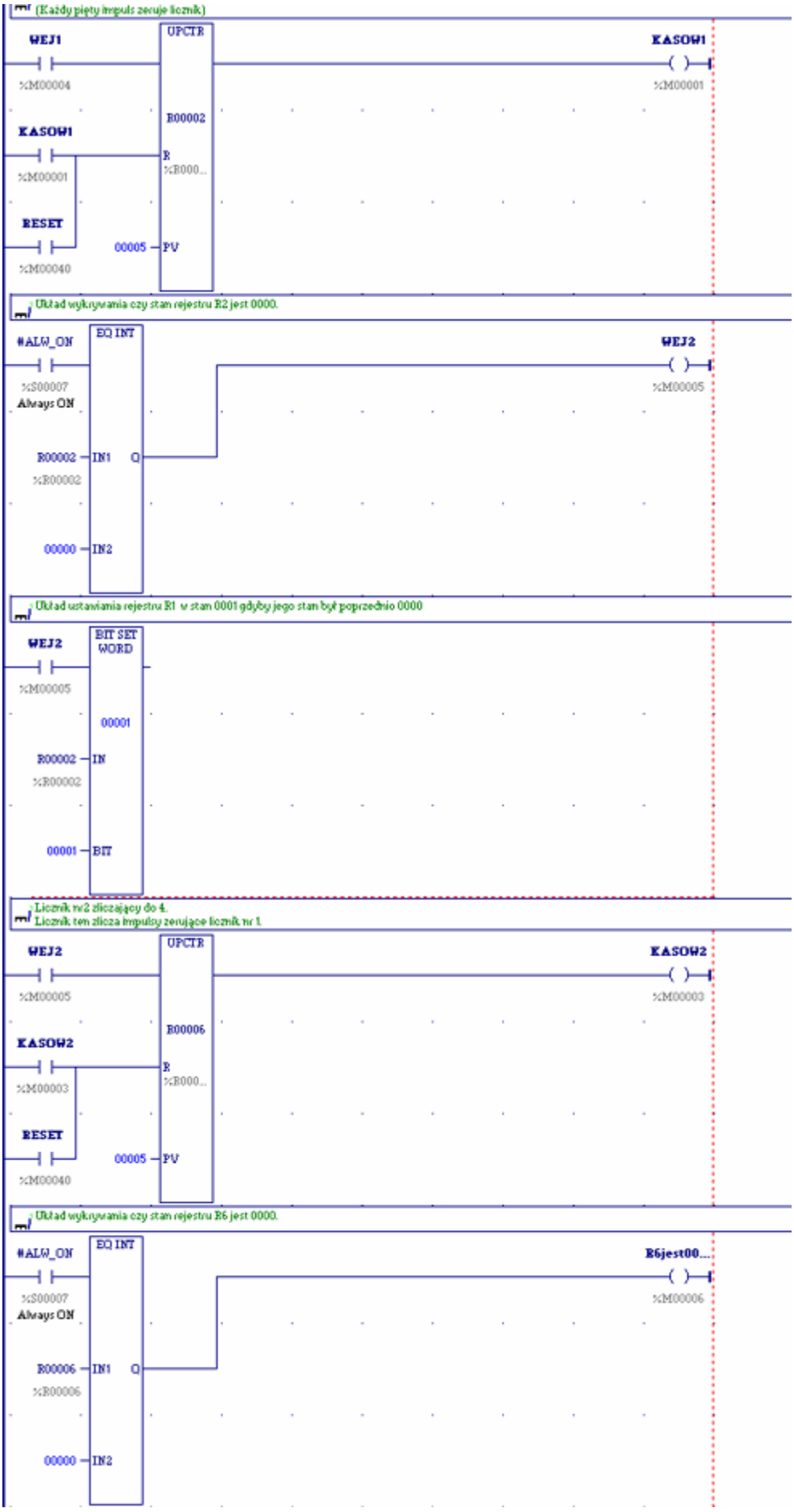
Zadanie 2.3 Relacje pomiędzy liczbami

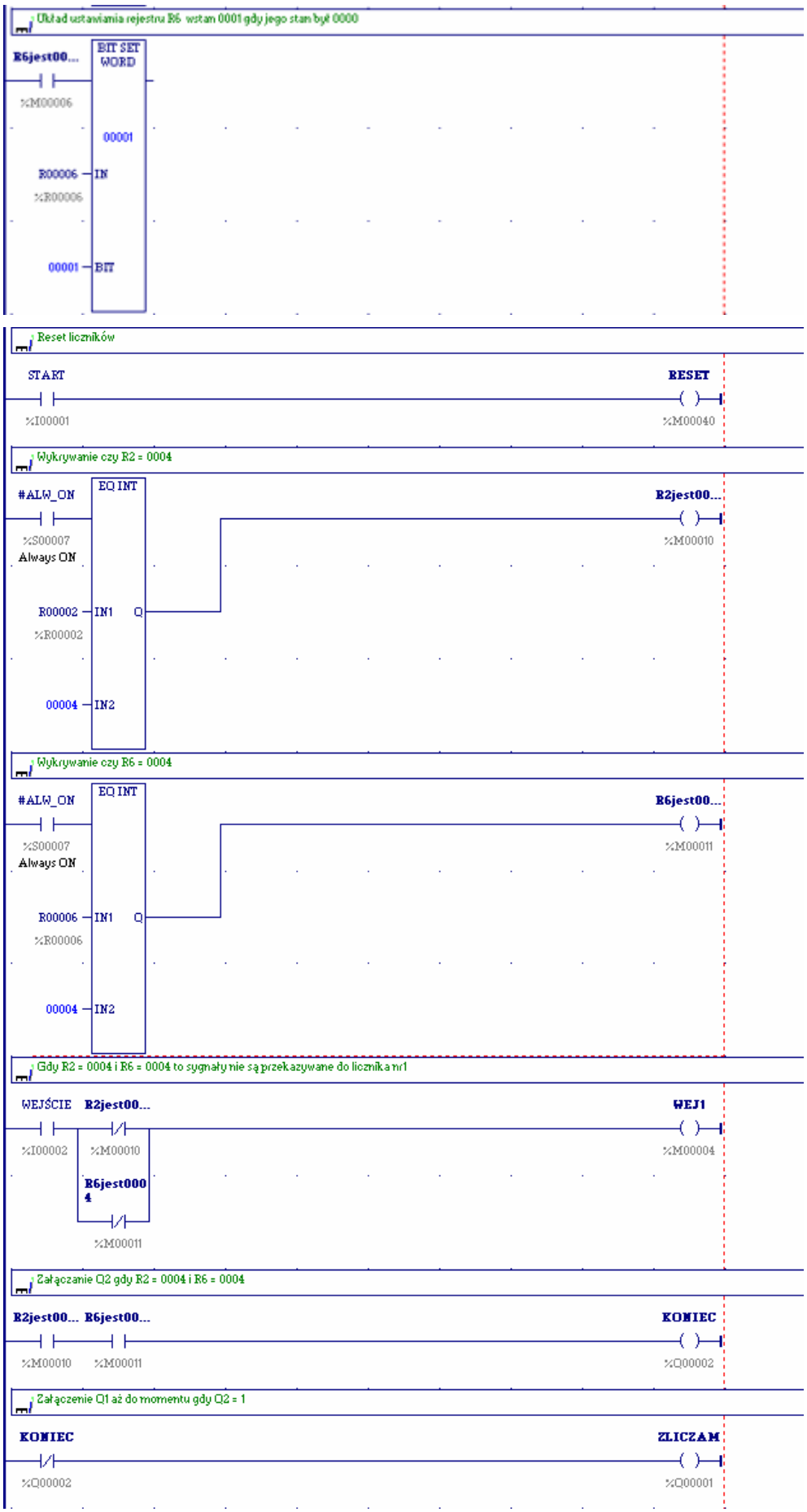


Zadanie 3 Generator fali prostokątnej

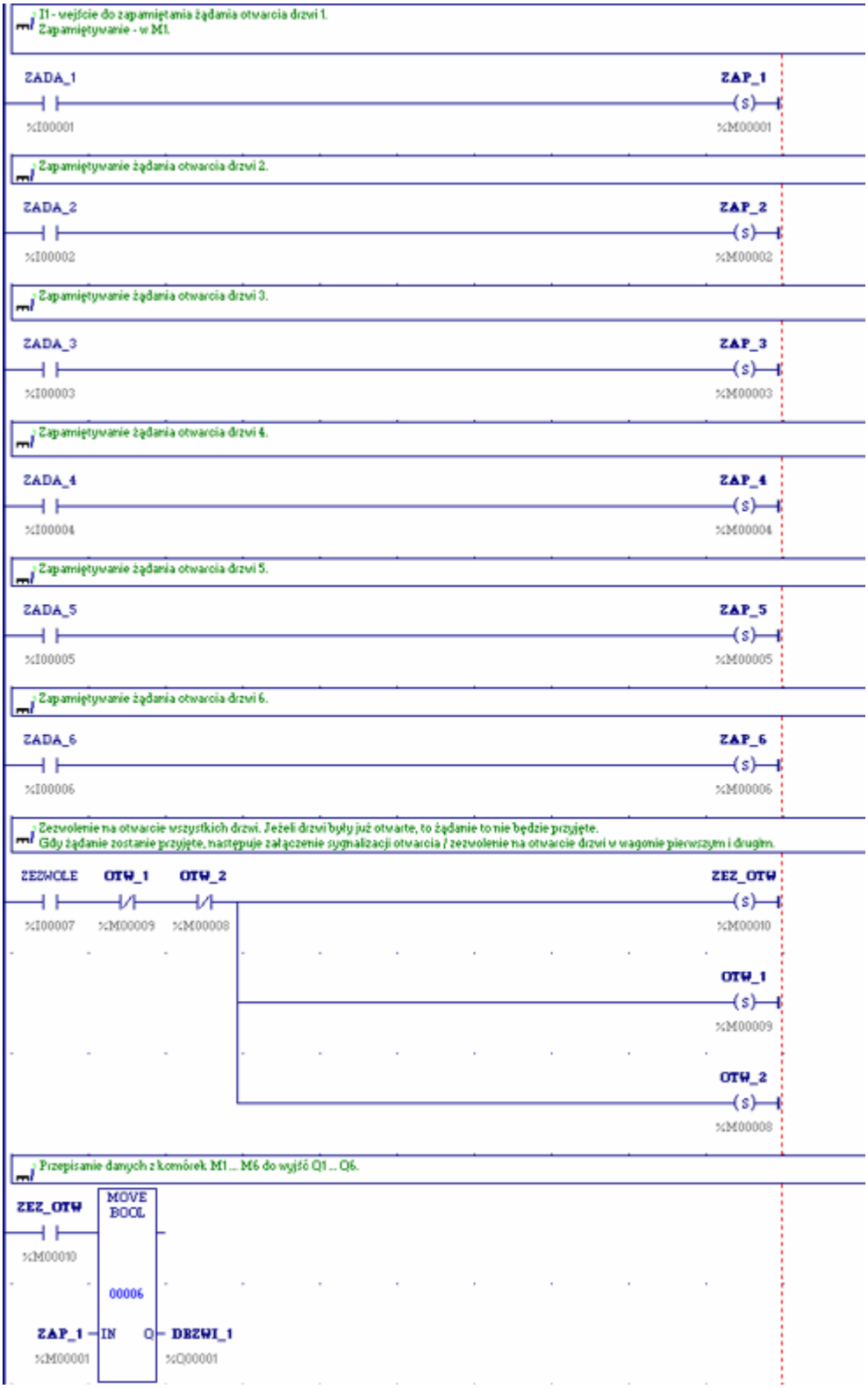


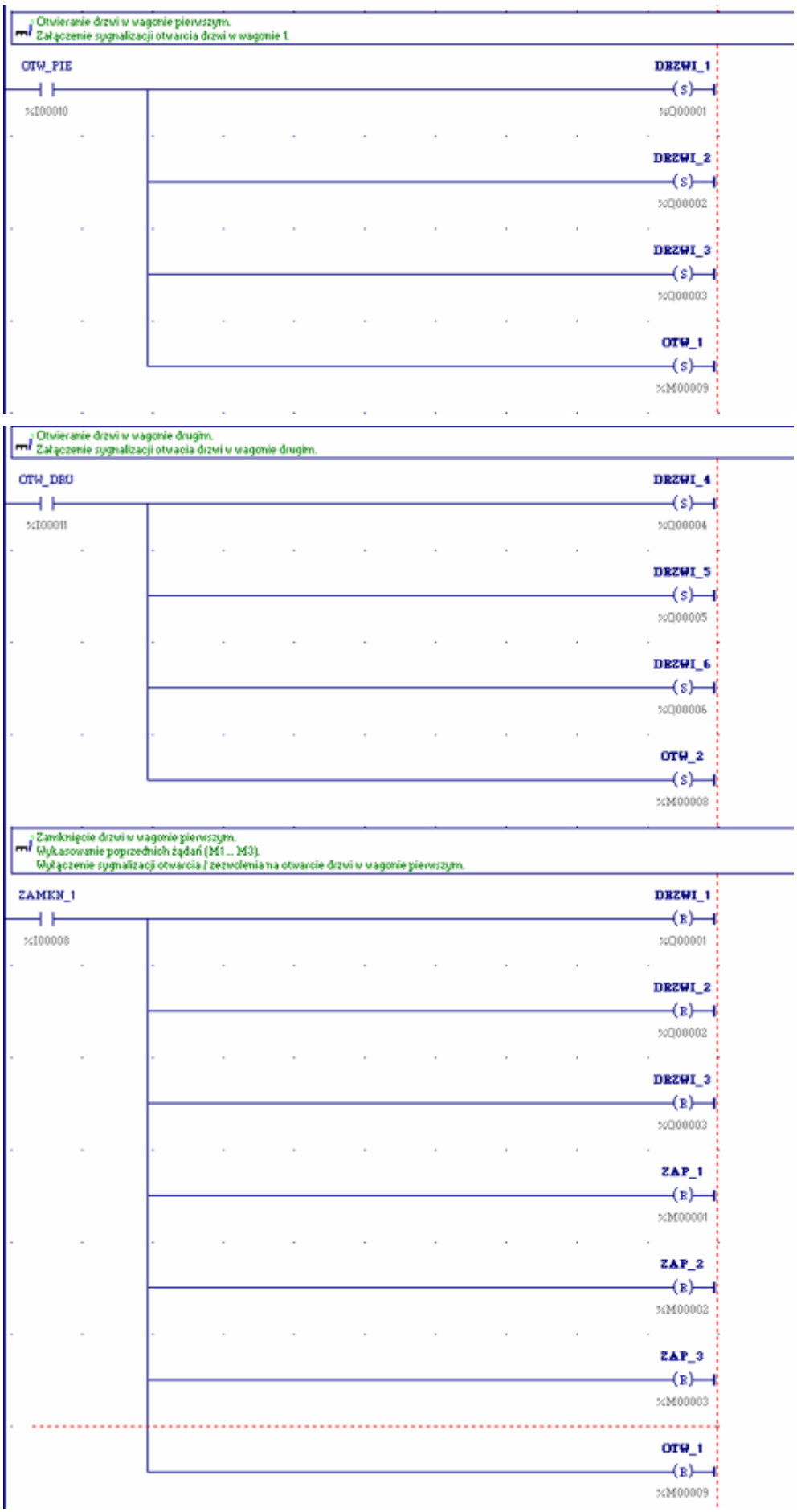
Zadanie 4 Liczniki kaskadowe

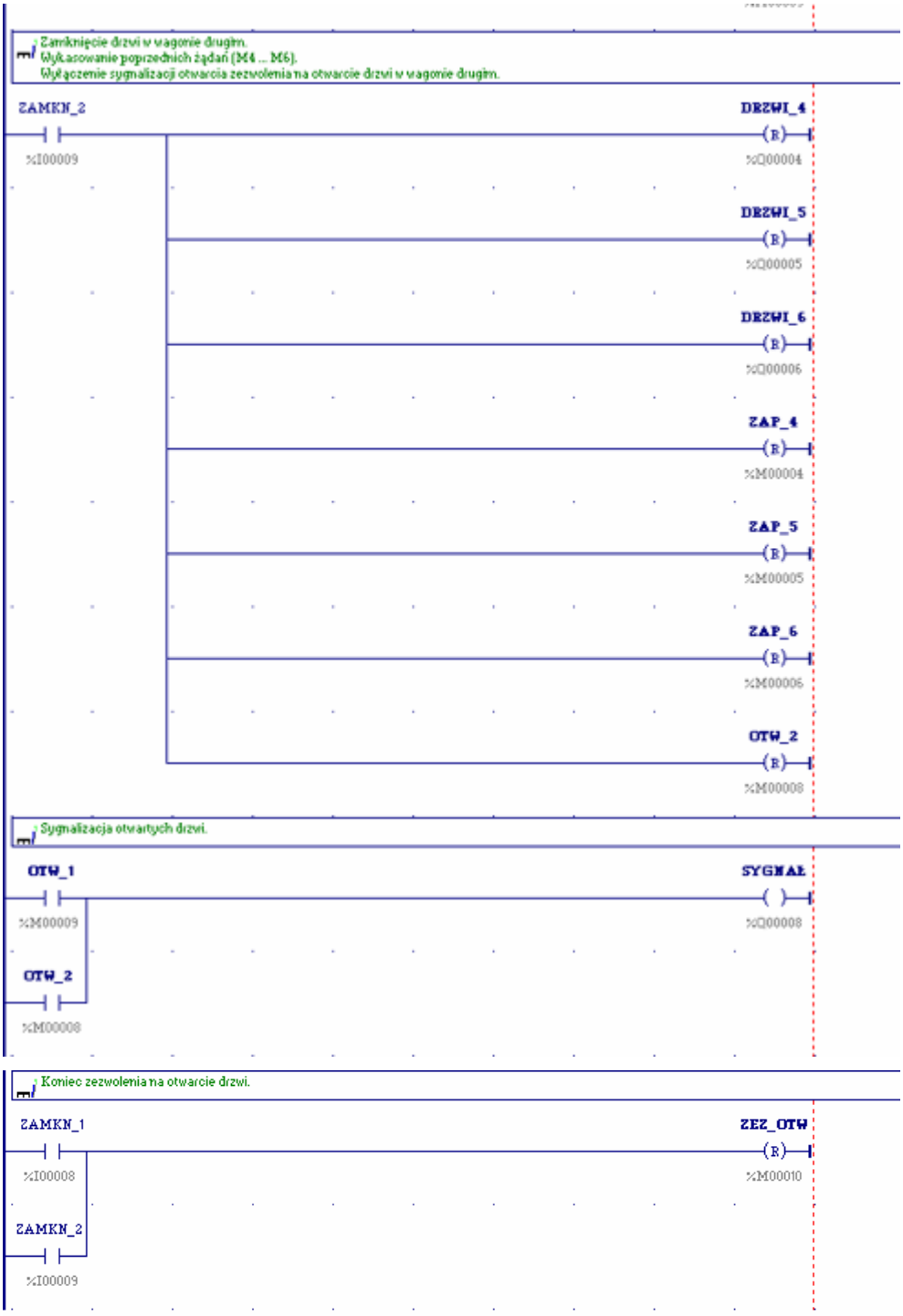



Układ ustawiania rejestru R6 wstan 0001 gdy jego stan był 0000
R6jest00...
%M00006
BIT SET WORD
00001
R00006
IN
%R00006
00001
BIT
Reset liczników
START
%I00001
RESET
%M00040
Wykrywanie czy R2 = 0004
#ALW_ON
%S00007
Always ON
EQ INT
R00002
%R00002
00004
IN1
IN2
Q
R2jest00...
%M00010
Wykrywanie czy R6 = 0004
#ALW_ON
%S00007
Always ON
EQ INT
R00006
%R00006
00004
IN1
IN2
Q
R6jest00...
%M00011
Gdy R2 = 0004 i R6 = 0004 to sygnały nie są przekazywane do licznika nr1
WEJŚCIE
%I00002
R2jest00...
%M00010
R6jest0004
%M00011
WEJ1
%M00004
Załączanie Q2 gdy R2 = 0004 i R6 = 0004
R2jest00...
%M00010
R6jest00...
%M00011
KONIEC
%Q00002
Załączenie Q1 aż do momentu gdy Q2 = 1
KONIEC
%Q00002
ZLICZAM
%Q00001

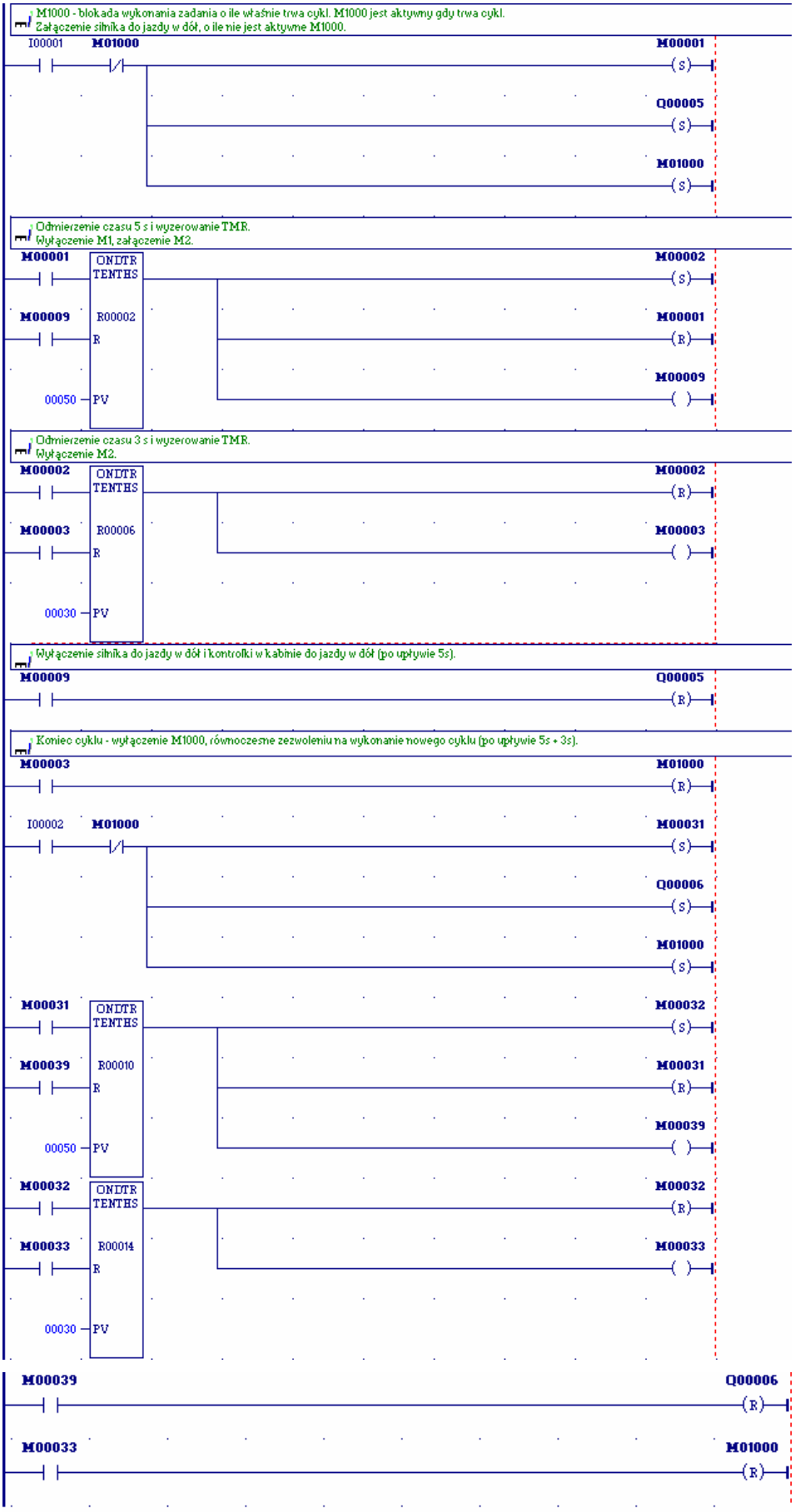
Zadanie 5 Sterownie drzwiami w tramwaju



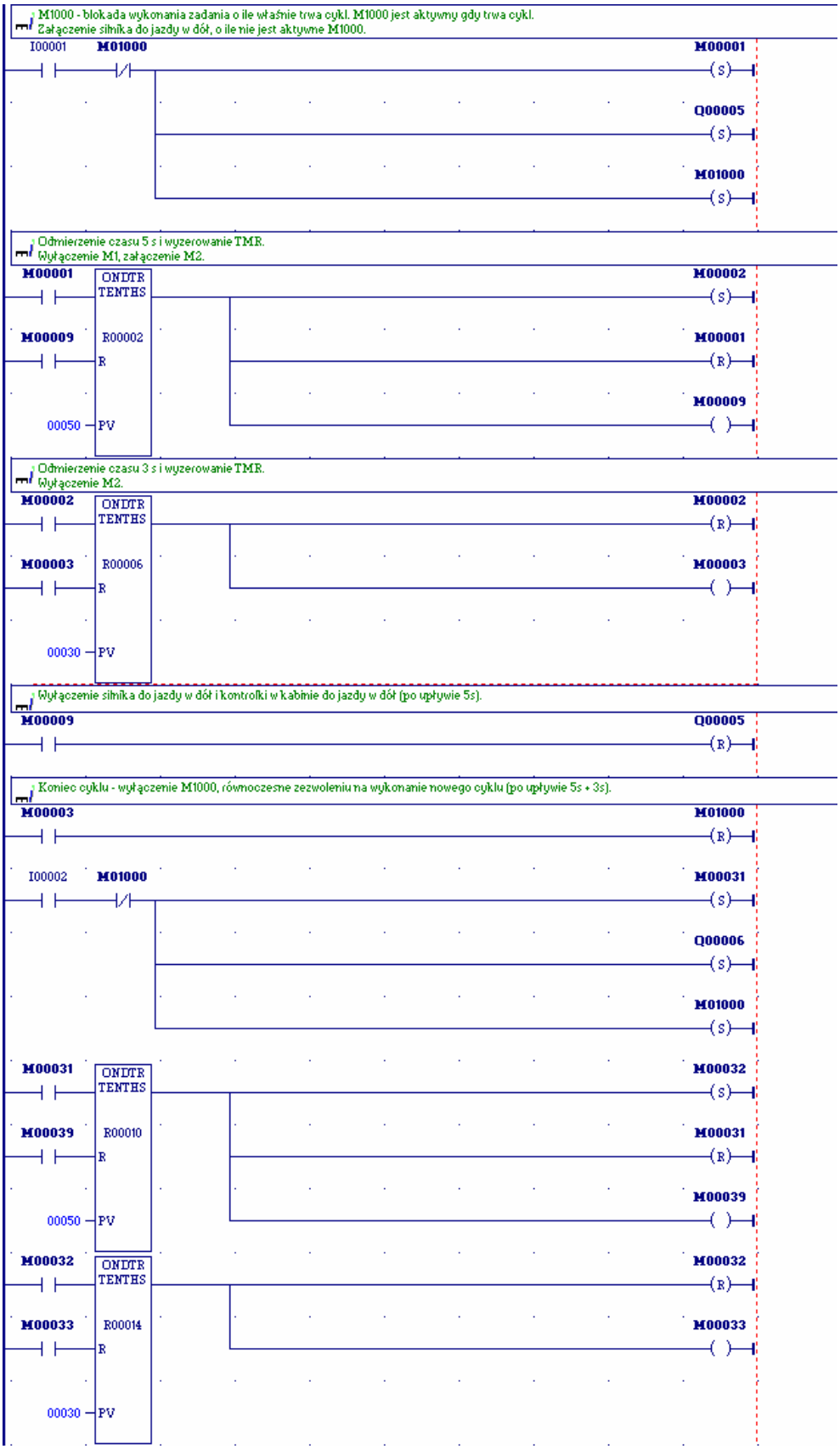

Otwieranie drzwi w wagonie pierwszym.
Załączenie sygnalizacji otwarcia drzwi w wagonie 1.
OTW_PIE
%I00010
DRZWI_1
(S)
%Q00001
DRZWI_2
(S)
%Q00002
DRZWI_3
(S)
%Q00003
OTW_1
(S)
%M00009
Otwieranie drzwi w wagonie drugim.
Załączenie sygnalizacji otwarcia drzwi w wagonie drugim.
OTW_DRU
%I00011
DRZWI_4
(S)
%Q00004
DRZWI_5
(S)
%Q00005
DRZWI_6
(S)
%Q00006
OTW_2
(S)
%M00008
Zamknięcie drzwi w wagonie pierwszym.
Wykasowanie poprzednich żądań (M1 .. M3).
Wyłączenie sygnalizacji otwarcia / zezwolenia na otwarcie drzwi w wagonie pierwszym.
ZAMKN_1
%I00008
DRZWI_1
(R)
%Q00001
DRZWI_2
(R)
%Q00002
DRZWI_3
(R)
%Q00003
ZAP_1
(R)
%M00001
ZAP_2
(R)
%M00002
ZAP_3
(R)
%M00003
OTW_1
(R)
%M00009

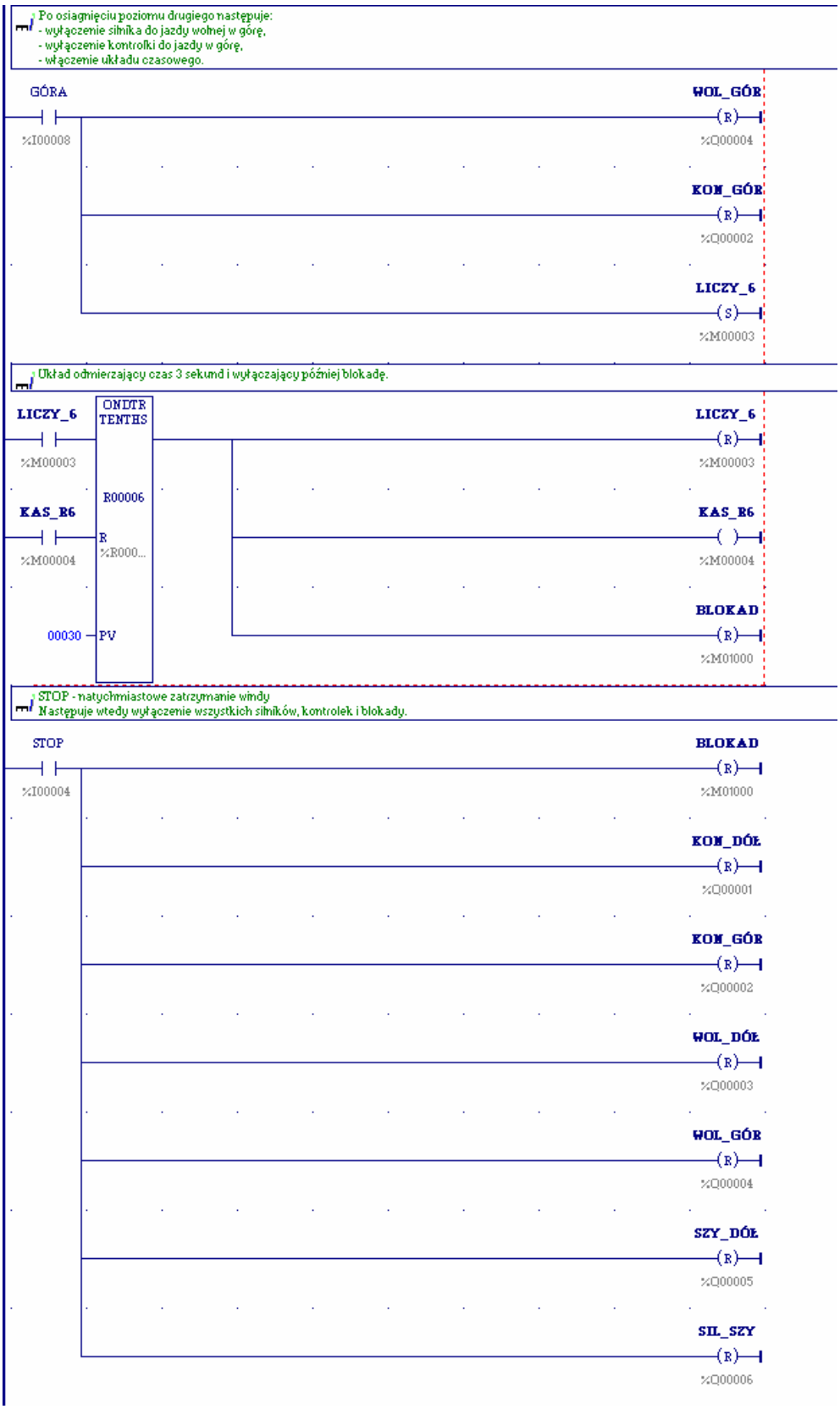

Zamknięcie drzwi w wagonie drugim.
Wykasowanie poprzednich żądań (M4 ... M6).
Wyłączenie sygnalizacji otwarcia zezwolenia na otwarcie drzwi w wagonie drugim.
ZAMKN_2
%I00009
DRZWI_4
%Q00004
DRZWI_5
%Q00005
DRZWI_6
%Q00006
ZAP_4
%M00004
ZAP_5
%M00005
ZAP_6
%M00006
OTW_2
%M00008
Sygnalizacja otwartych drzwi.
OTW_1
%M00009
OTW_2
%M00008
SYGNAŁ
%Q00008
Koniec zezwolenia na otwarcie drzwi.
ZAMKN_1
%I00008
ZAMKN_2
%I00009
ZEZ_OTW
%M00010

Zadanie 6.1 Sterowanie windą dwu poziomową

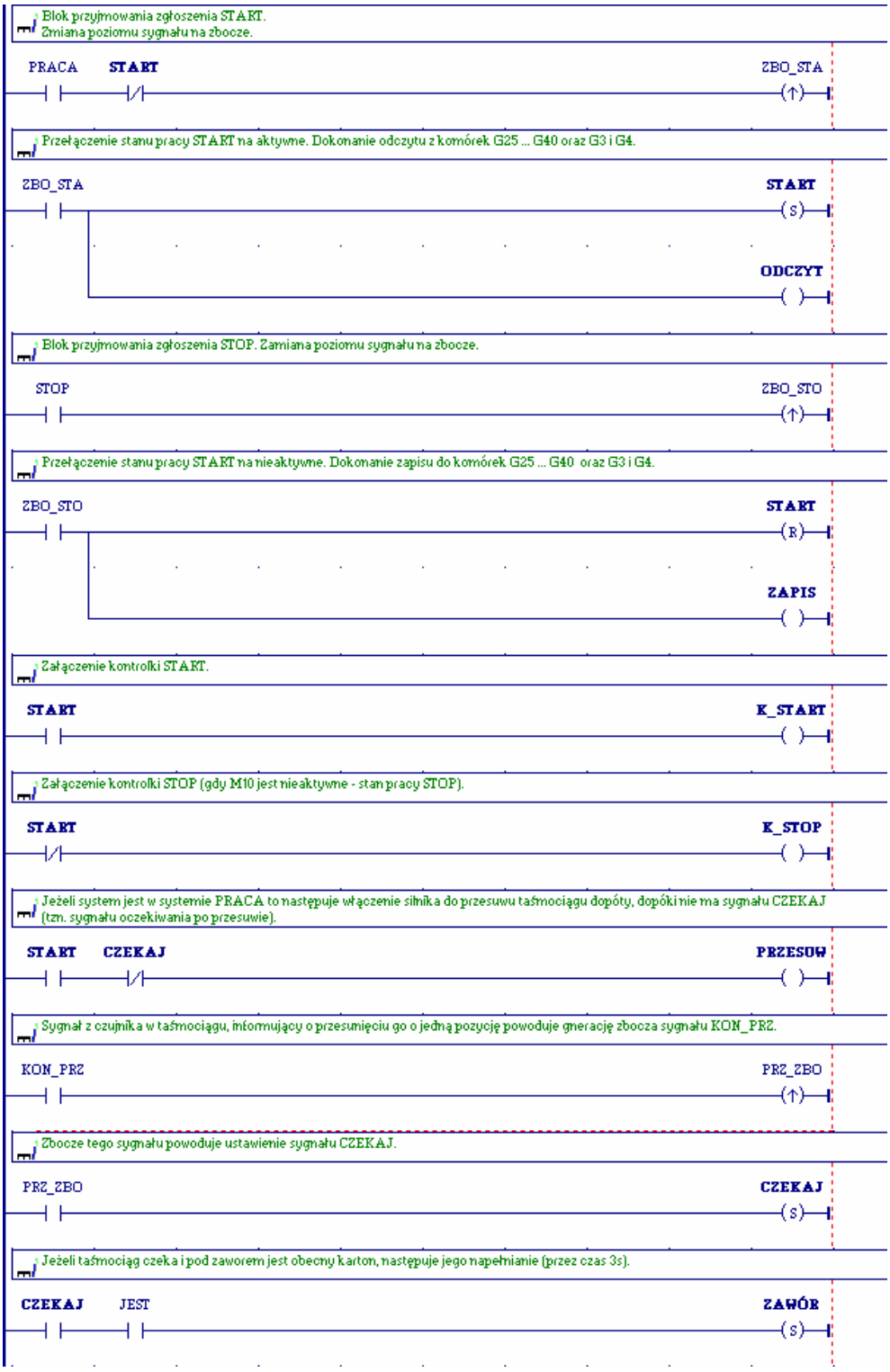


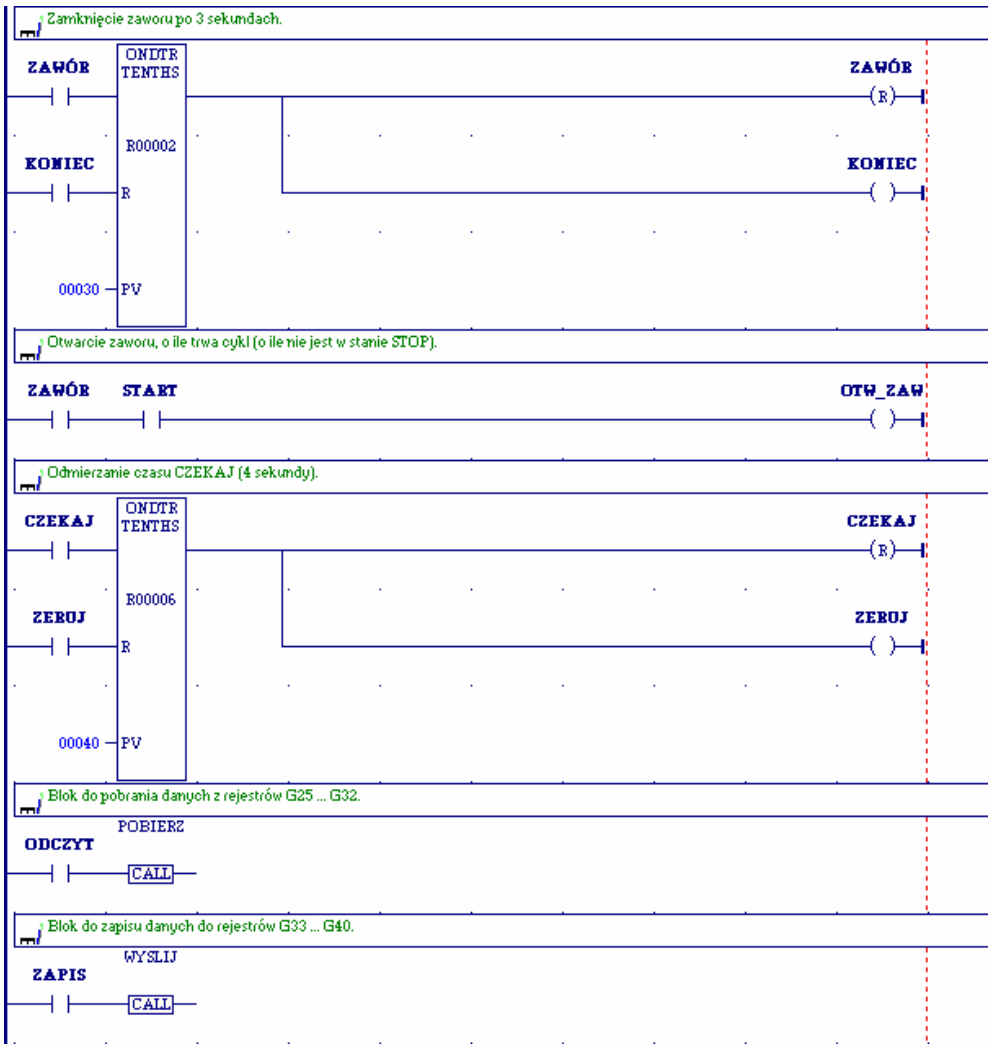
Zadanie 6.2 Sterowanie windą dwu poziomową z czujnikami



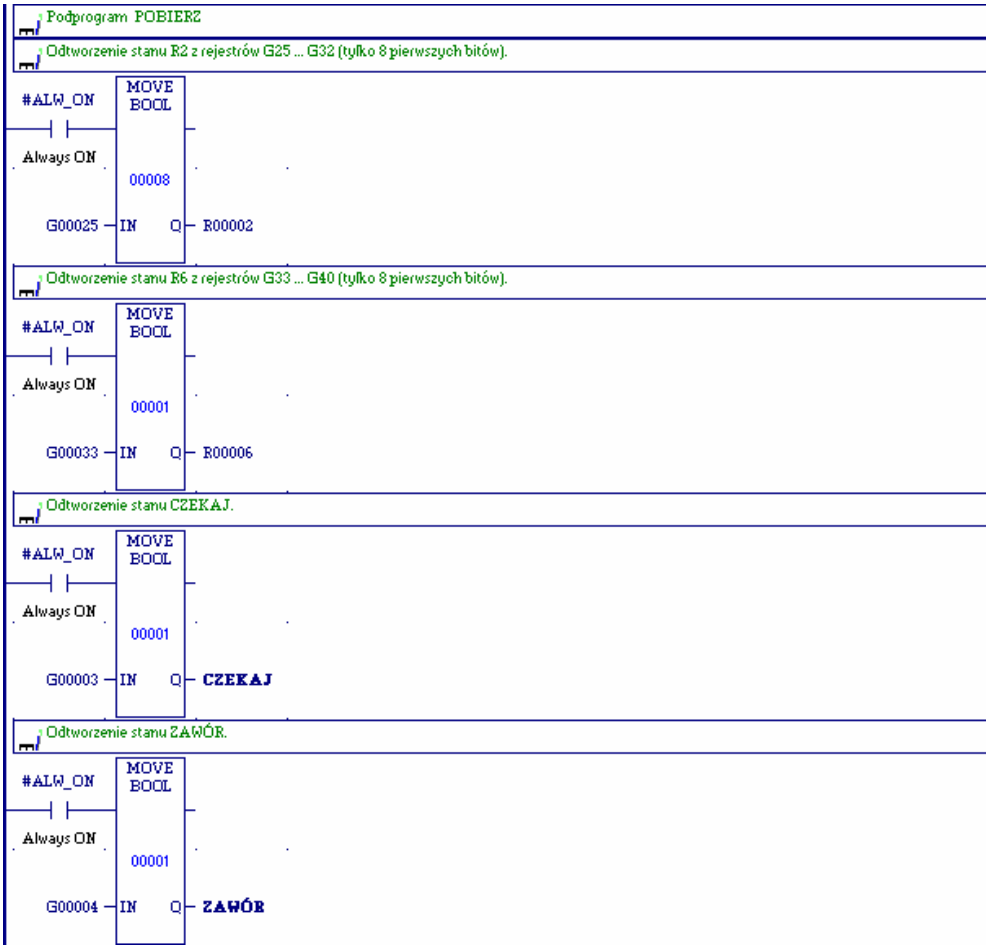

Po osiągnięciu poziomu drugiego następuje:
- wyłączenie silnika do jazdy wolnej w górę,
- wyłączenie kontrolki do jazdy w górę,
- włączenie układu czasowego.
GÓRA
%I00008
WOL_GÓR
(R)
%Q00004
KON_GÓR
(R)
%Q00002
LICZY_6
(S)
%M00003
Układ odmierzający czas 3 sekund i wyłączający później blokadę.
LICZY_6
%M00003
ONDTR
TENTHS
R00006
R
%R000...
00030
PV
LICZY_6
(R)
%M00003
KAS_R6
%M00004
KAS_R6
( )
%M00004
BLOKAD
(R)
%M01000
STOP - natychmiastowe zatrzymanie windy
Następuje wtedy wyłączenie wszystkich silników, kontrolek i blokady.
STOP
%I00004
BLOKAD
(R)
%M01000
KON_DÓŁ
(R)
%Q00001
KON_GÓR
(R)
%Q00002
WOL_DÓŁ
(R)
%Q00003
WOL_GÓR
(R)
%Q00004
SZY_DÓŁ
(R)
%Q00005
SIL_SZY
(R)
%Q00006

Zadanie 7 Linia napełniania kartonów z zabezpieczeniami

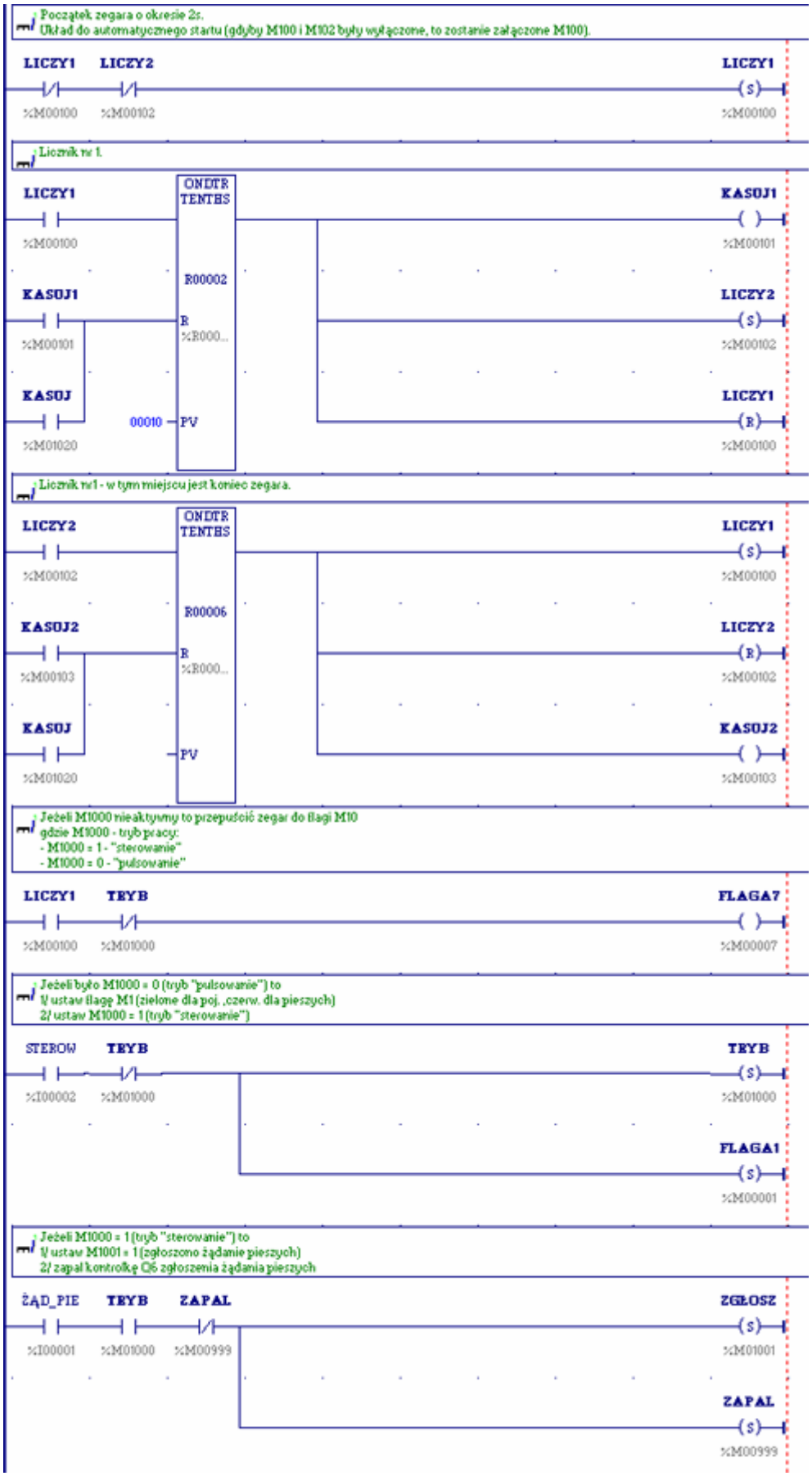


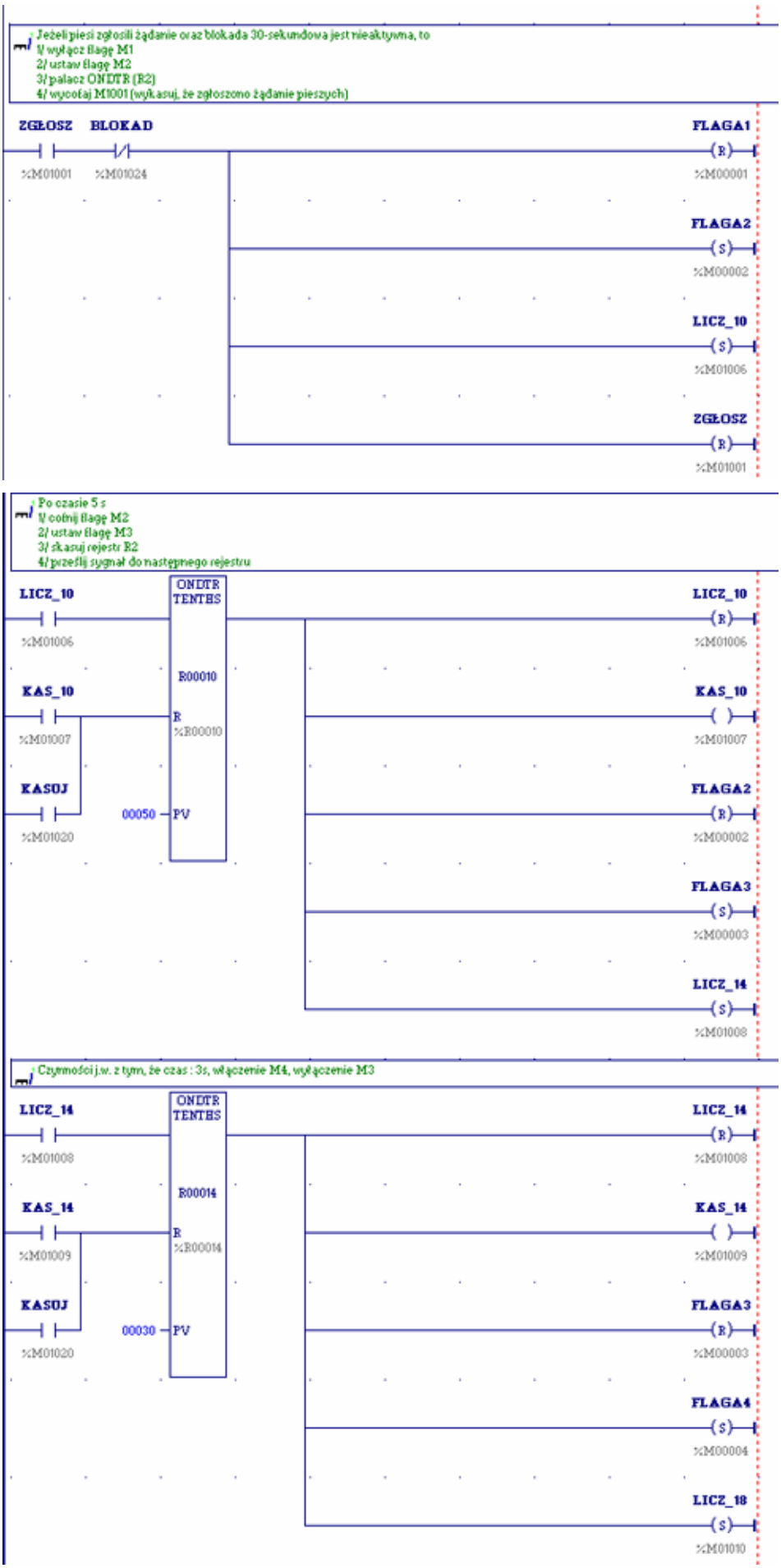

Zamknięcie zaworu po 3 sekundach.
ZAWÓR
ONDTR
TENTHS
ZAWÓR
(R)
R00002
KONIEC
KONIEC
R
00030
PV
Otwarcie zaworu, o ile trwa cykl (o ile nie jest w stanie STOP).
ZAWÓR
START
OTW_ZAW
Odmierzanie czasu CZEKAJ (4 sekundy).
CZEKAJ
ONDTR
TENTHS
CZEKAJ
(R)
R00006
ZEROJ
ZEROJ
R
00040
PV
Blok do pobrania danych z rejestrów G25 ... G32.
POBIERZ
ODCZYT
CALL
Blok do zapisu danych do rejestrów G33 ... G40.
WYSLIJ
ZAPIS
CALL

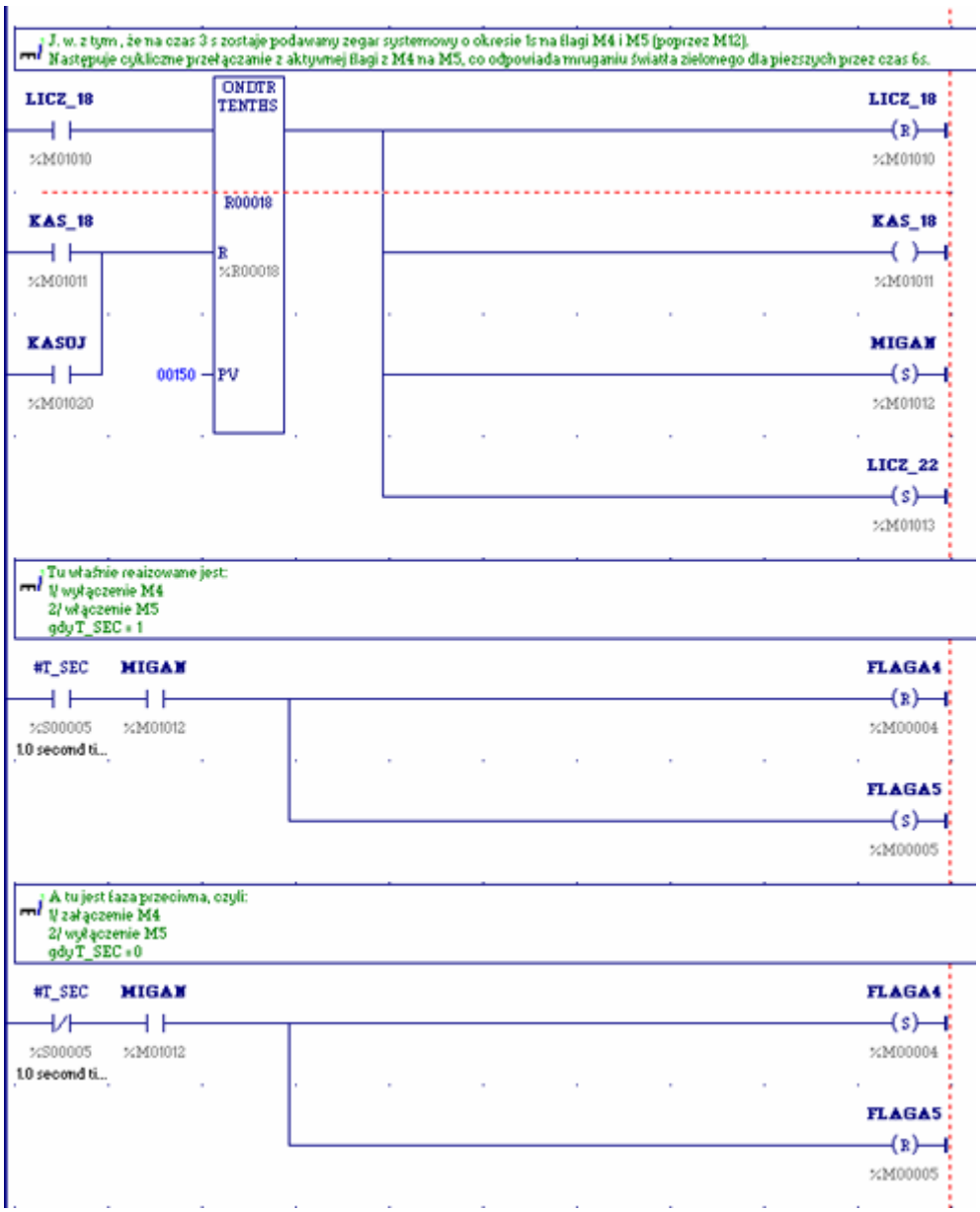

Podprogram WYŚLIJ
Zapis stanu R2 do rejestrów G25 ... G32 (tylko 8 pierwszych bitów).
#ALW_ON
Always ON
MOVE BOOL
00008
R00002
IN
Q
G00025
Zapis stanu R2 do rejestrów G33 ... G40 (tylko 8 pierwszych bitów).
#ALW_ON
Always ON
MOVE WORD
00008
R00006
IN
Q
G00033
Wysłanie zawartości rejestru M1 do rejestru G3.
#ALW_ON
Always ON
MOVE BOOL
00001
CZEKAJ
IN
Q
G00003
Wysłanie zawartości rejestru M2 do rejestru G4.
#ALW_ON
Always ON
MOVE BOOL
00001
ZAWÓR
IN
Q
G00004

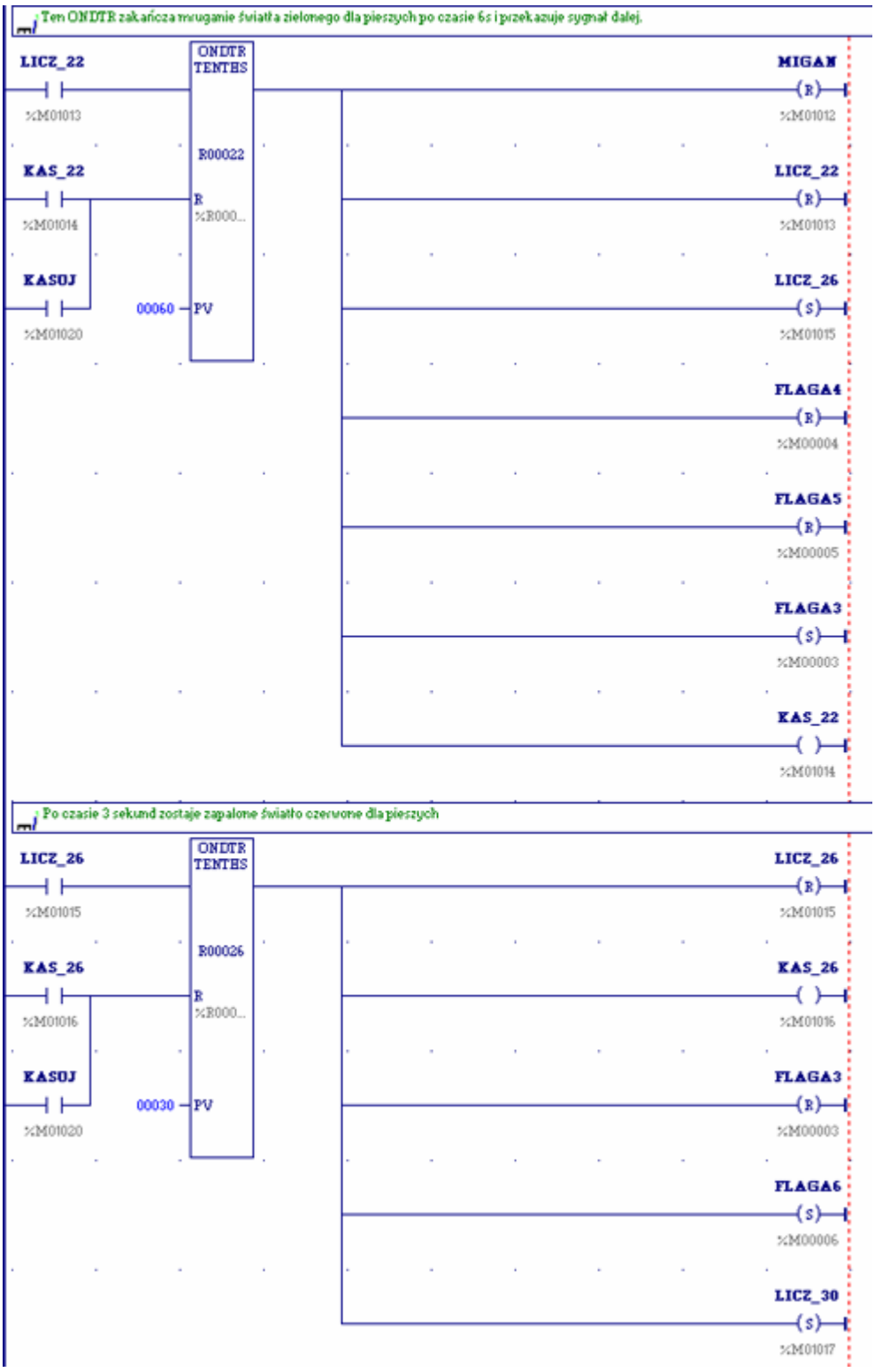

Podprogram POBIERZ
Odtworzenie stanu R2 z rejestrów G25 ... G32 (tylko 8 pierwszych bitów).
#ALW_ON
Always ON
MOVE BOOL
00008
G00025
IN
Q
R00002
Odtworzenie stanu R6 z rejestrów G33 ... G40 (tylko 8 pierwszych bitów).
#ALW_ON
Always ON
MOVE BOOL
00001
G00033
IN
Q
R00006
Odtworzenie stanu CZEKAJ.
#ALW_ON
Always ON
MOVE BOOL
00001
G00003
IN
Q
CZEKAJ
Odtworzenie stanu ZAWÓR.
#ALW_ON
Always ON
MOVE BOOL
00001
G00004
IN
Q
ZAWÓR

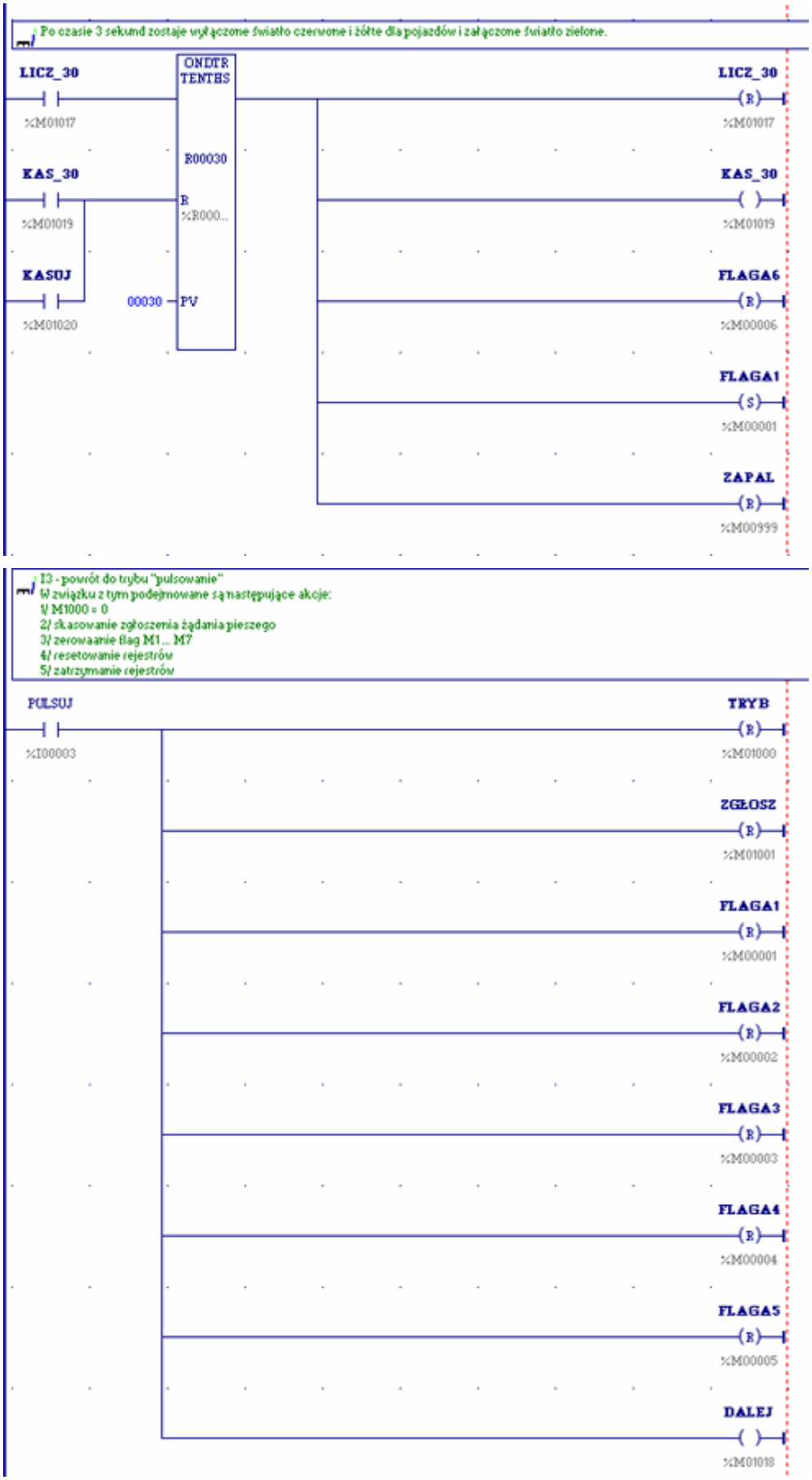
Zadanie 8 Sygnalizacja świetlna

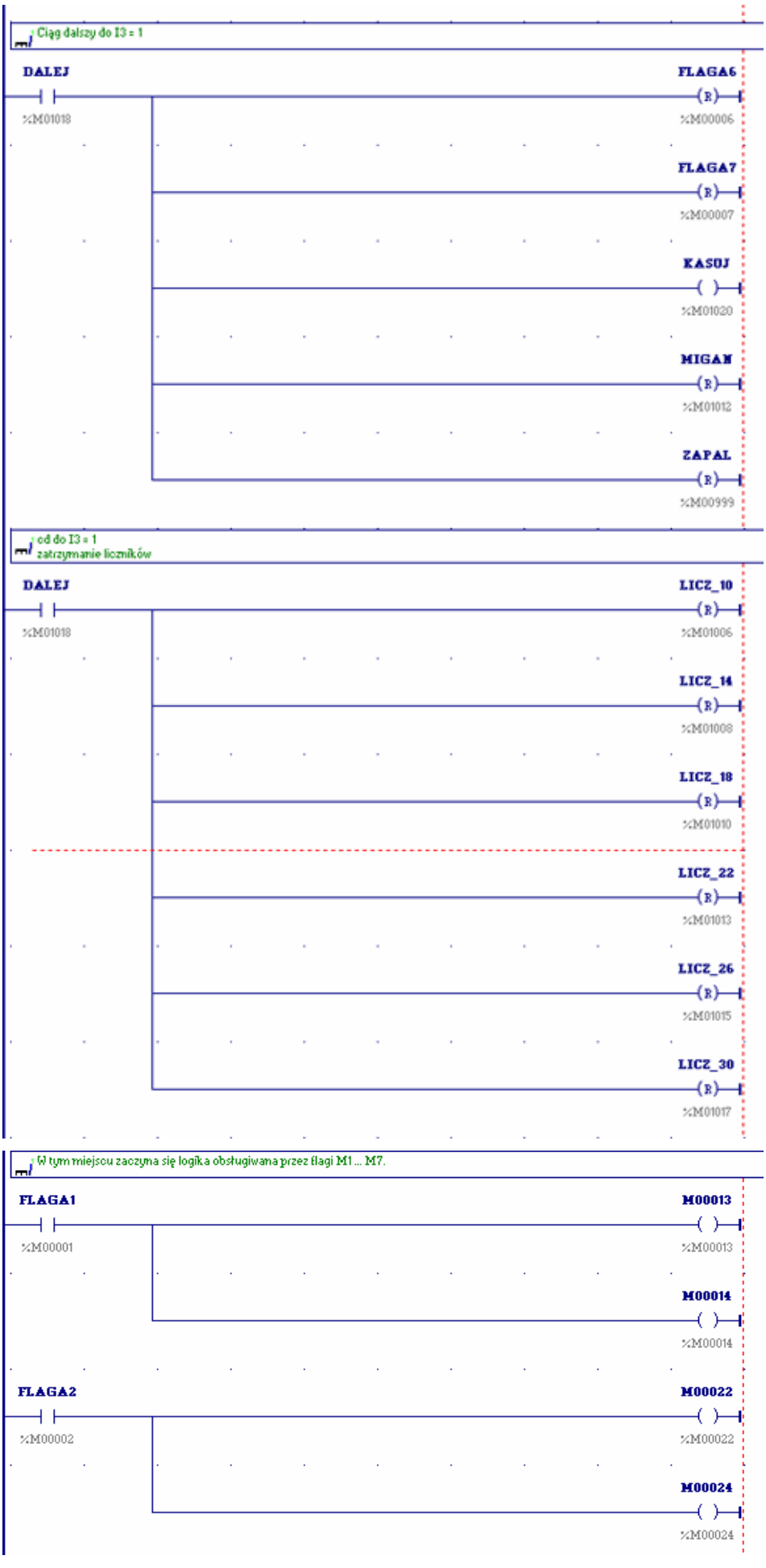


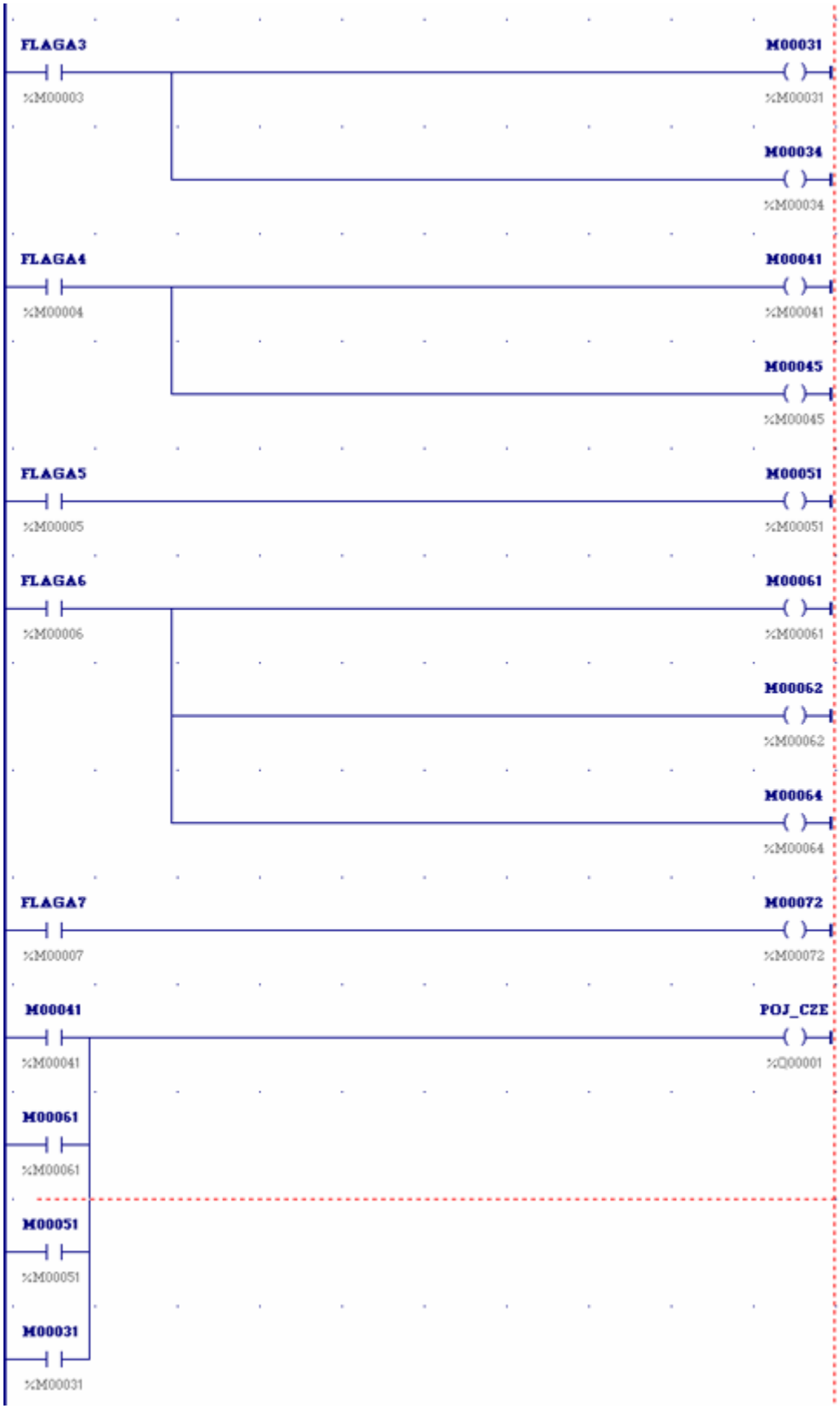

Jeżeli piesi zgłosili żądanie oraz blokada 30-sekundowa jest nieaktywna, to
1/ wyłącz flagę M1
2/ ustaw flagę M2
3/ palacz ONDTR (R2)
4/ wycofaj M1001 (wykasuj, że zgłoszono żądanie pieszych)
ZGŁOSZ
%M01001
BLOKAD
%M01024
FLAGA1
%M00001
FLAGA2
%M00002
LICZ_10
%M01006
ZGŁOSZ
%M01001
Po czasie 5 s
1/ cofnij flagę M2
2/ ustaw flagę M3
3/ skasuj rejestr R2
4/ prześlij sygnał do następnego rejestru
LICZ_10
%M01006
ONDTR
TENTHS
R00010
KAS_10
%M01007
R
%R00010
KASOJ
%M01020
00050
PV
LICZ_10
%M01006
KAS_10
%M01007
FLAGA2
%M00002
FLAGA3
%M00003
LICZ_14
%M01008
Czynności j.w. z tym, że czas : 3s, włączenie M4, wyłączenie M3
LICZ_14
%M01008
ONDTR
TENTHS
R00014
KAS_14
%M01009
R
%R00014
KASOJ
%M01020
00030
PV
LICZ_14
%M01008
KAS_14
%M01009
FLAGA3
%M00003
FLAGA4
%M00004
LICZ_18
%M01010

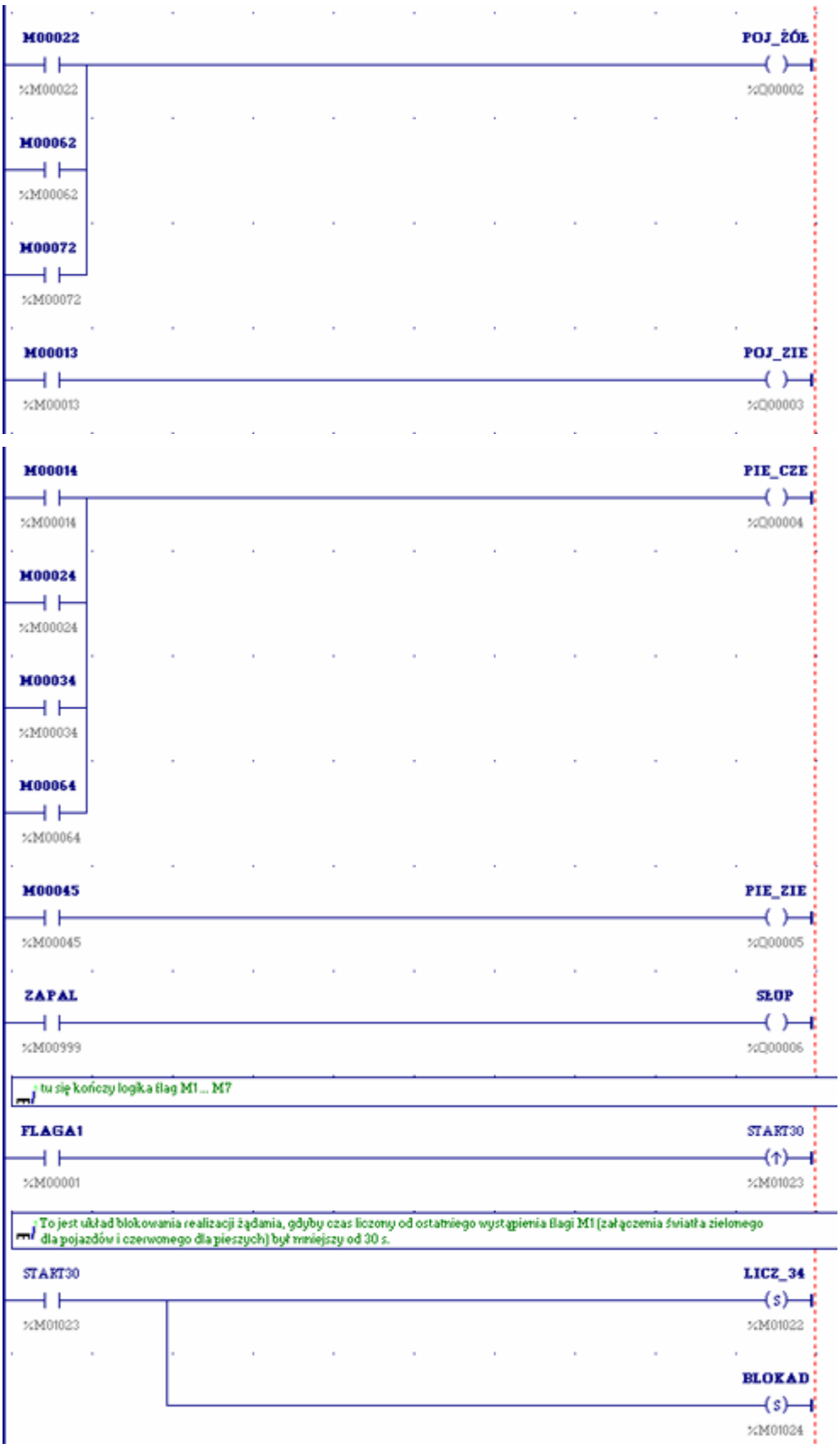

J. w. z tym , że na czas 3 s zostaje podawany zegar systemowy o okresie 1s na flagi M4 i M5 (poprzez M12).
Następuje cykliczne przełączanie z aktywnej flagi z M4 na M5, co odpowiada mruganiu światła zielonego dla pieszych przez czas 6s.
LICZ_18
%M01010
ONDTR
TENTHS
R00018
R
%R00018
PV
00150
KAS_18
%M01011
KASOJ
%M01020
LICZ_18
%M01010
KAS_18
%M01011
MIGAN
%M01012
LICZ_22
%M01013
Tu właśnie realizowane jest:
1/ wyłączenie M4
2/ włączenie M5
gdy T_SEC = 1
#T_SEC
%S00005
1.0 second ti...
MIGAN
%M01012
FLAGA4
%M00004
FLAGA5
%M00005
A tu jest faza przeciwna, czyli:
1/ załączenie M4
2/ wyłączenie M5
gdy T_SEC = 0
#T_SEC
%S00005
1.0 second ti...
MIGAN
%M01012
FLAGA4
%M00004
FLAGA5
%M00005

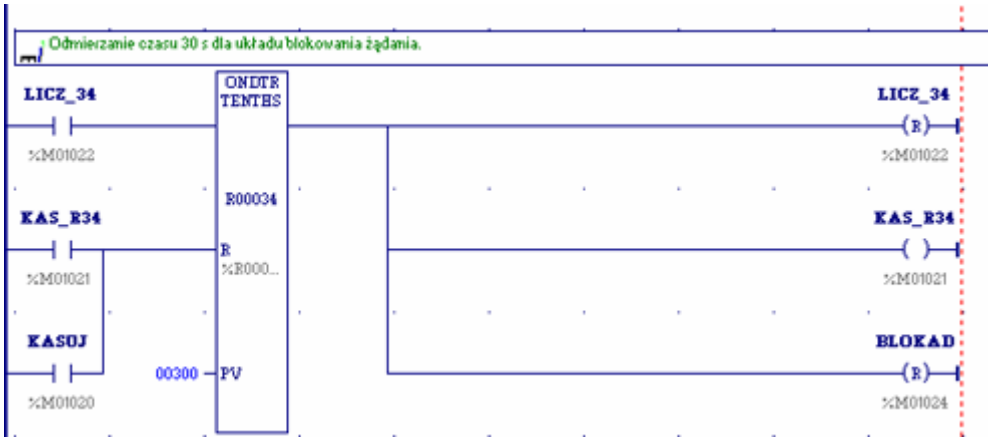

Ten ONDTR zakańcza mruganie światła zielonego dla pieszych po czasie 6s i przekazuje sygnał dalej.
LICZ_22
%M01013
ONDTR
TENTHS
R00022
R
%R000...
KAS_22
%M01014
KASOJ
%M01020
00060
PV
MIGAM
(R)
%M01012
LICZ_22
(R)
%M01013
LICZ_26
(S)
%M01015
FLAGA4
(R)
%M00004
FLAGA5
(R)
%M00005
FLAGA3
(S)
%M00003
KAS_22
( )
%M01014
Po czasie 3 sekund zostaje zapalone światło czerwone dla pieszych
LICZ_26
%M01015
ONDTR
TENTHS
R00026
R
%R000...
KAS_26
%M01016
KASOJ
%M01020
00030
PV
LICZ_26
(R)
%M01015
KAS_26
( )
%M01016
FLAGA3
(R)
%M00003
FLAGA6
(S)
%M00006
LICZ_30
(S)
%M01017


Po czasie 3 sekund zostaje wyłączone światło czerwone i żółte dla pojazdów i załączone światło zielone.
LICZ_30
%M01017
ONDTR
TENTHS
R00030
R
%R000...
00030
PV
LICZ_30
(R)
%M01017
KAS_30
%M01019
KAS_30
( )
%M01019
KASUJ
%M01020
FLAGA6
(R)
%M00006
FLAGA1
(S)
%M00001
ZAPAL
(R)
%M00999
I3 - powrót do trybu "pulsowanie"
W związku z tym podejmowane są następujące akcje:
1/ M1000 = 0
2/ skasowanie zgłoszenia żądania pieszego
3/ zerowaanie flag M1 ... M7
4/ resetowanie rejestrów
5/ zatrzymanie rejestrów
PULSUJ
%I00003
TRYB
(R)
%M01000
ZGŁOSZ
(R)
%M01001
FLAGA1
(R)
%M00001
FLAGA2
(R)
%M00002
FLAGA3
(R)
%M00003
FLAGA4
(R)
%M00004
FLAGA5
(R)
%M00005
DALEJ
( )
%M01018

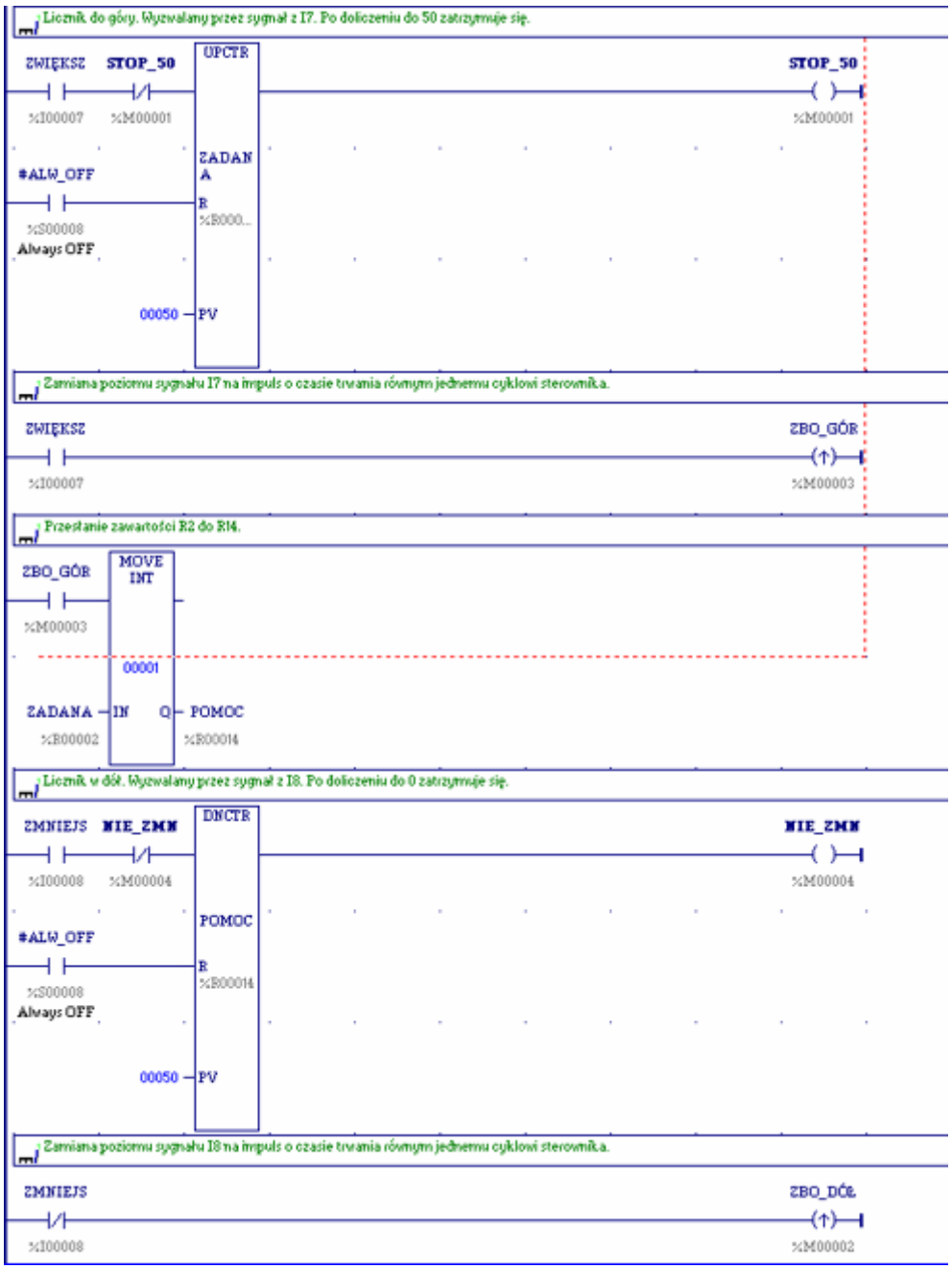

Ciąg dalszy do I3 = 1
DALEJ
%M01018
FLAGA6
(R)
%M00006
FLAGA7
(R)
%M00007
KASUJ
( )
%M01020
MIGAM
(R)
%M01012
ZAPAL
(R)
%M00999
od do I3 = 1
zatrzymanie liczników
DALEJ
%M01018
LICZ_10
(R)
%M01006
LICZ_14
(R)
%M01008
LICZ_18
(R)
%M01010
LICZ_22
(R)
%M01013
LICZ_26
(R)
%M01015
LICZ_30
(R)
%M01017
W tym miejscu zaczyna się logika obsługiwana przez flagi M1 ... M7.
FLAGA1
%M00001
M00013
( )
%M00013
M00014
( )
%M00014
FLAGA2
%M00002
M00022
( )
%M00022
M00024
( )
%M00024

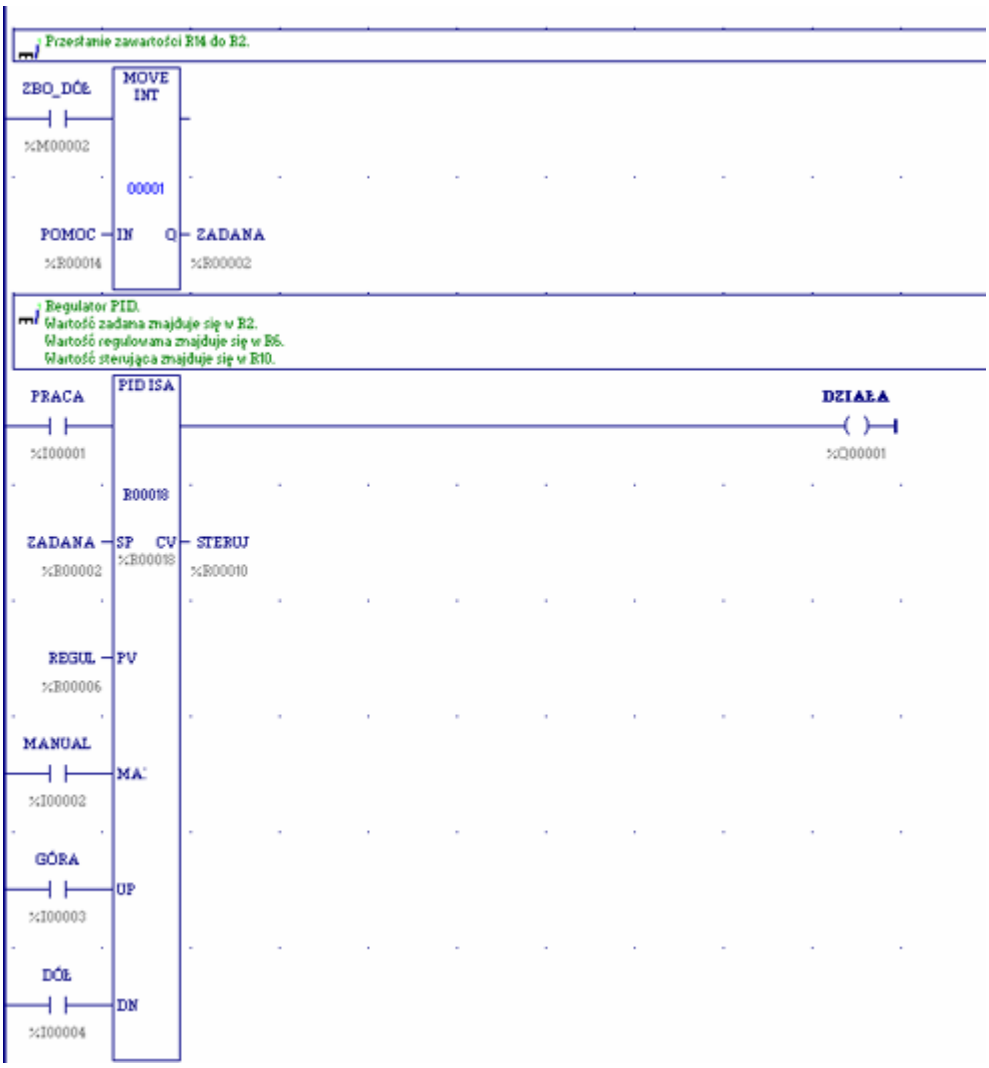

FLAGA3
%M00003
M00031
%M00031
M00034
%M00034
FLAGA4
%M00004
M00041
%M00041
M00045
%M00045
FLAGA5
%M00005
M00051
%M00051
FLAGA6
%M00006
M00061
%M00061
M00062
%M00062
M00064
%M00064
FLAGA7
%M00007
M00072
%M00072
M00041
%M00041
POJ_CZE
%Q00001
M00061
%M00061
M00051
%M00051
M00031
%M00031


M00022
POJ_ŻÓŁ
%M00022
%Q00002
M00062
%M00062
M00072
%M00072
M00013
POJ_ZIE
%M00013
%Q00003
M00014
PIE_CZE
%M00014
%Q00004
M00024
%M00024
M00034
%M00034
M00064
%M00064
M00045
PIE_ZIE
%M00045
%Q00005
ZAPAL
SŁUP
%M00999
%Q00006
tu się kończy logika flag M1 ... M7
FLAGA1
START30
%M00001
%M01023
To jest układ blokowania realizacji żądania, gdyby czas liczony od ostatniego wystąpienia flagi M1 (załączenia światła zielonego dla pojazdów i czerwonego dla pieszych) był mniejszy od 30 s.
START30
LICZ_34
%M01023
%M01022
BLOKAD
%M01024



Zadanie 9 Regulator PID

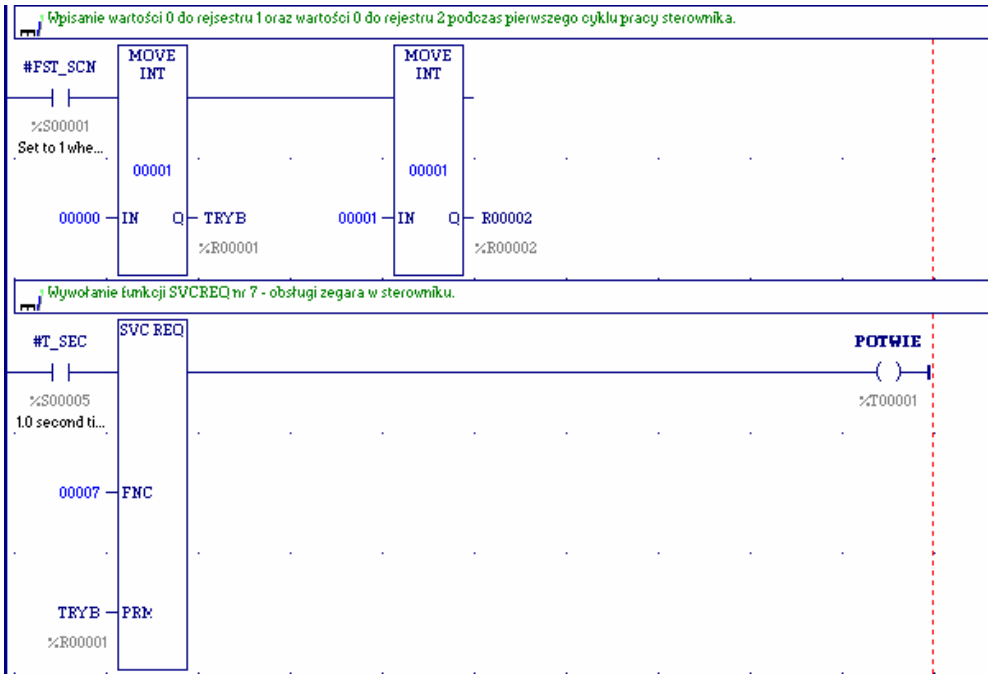




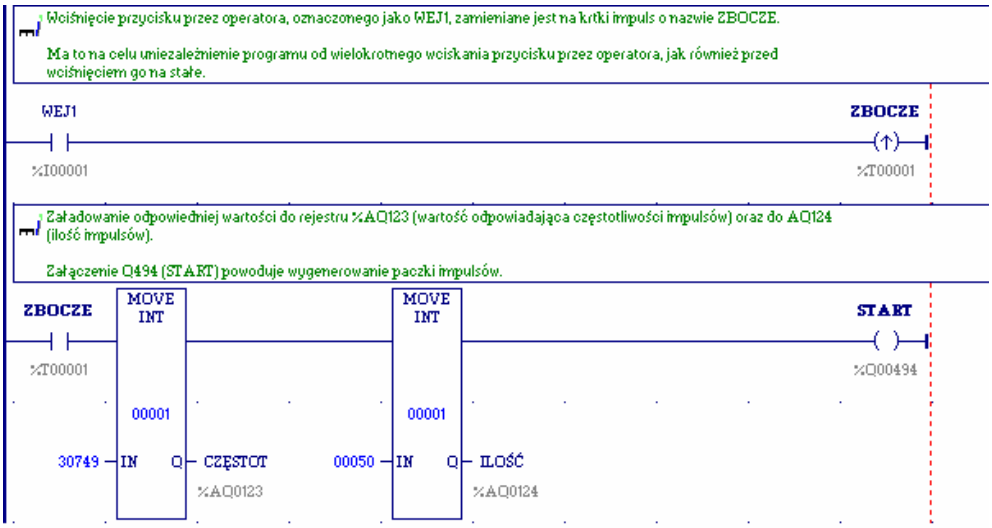
Proponowane parametry regulatora PID:

Proportional	1 % / %
Derivative	0 sec
Integral	0 rep/sec
Sample period	0 sec
Dead band +	0
Dead band -	0
Bias	0
Min slew time	10 sec
Upper clamp +	32000
Lower clamp -	0

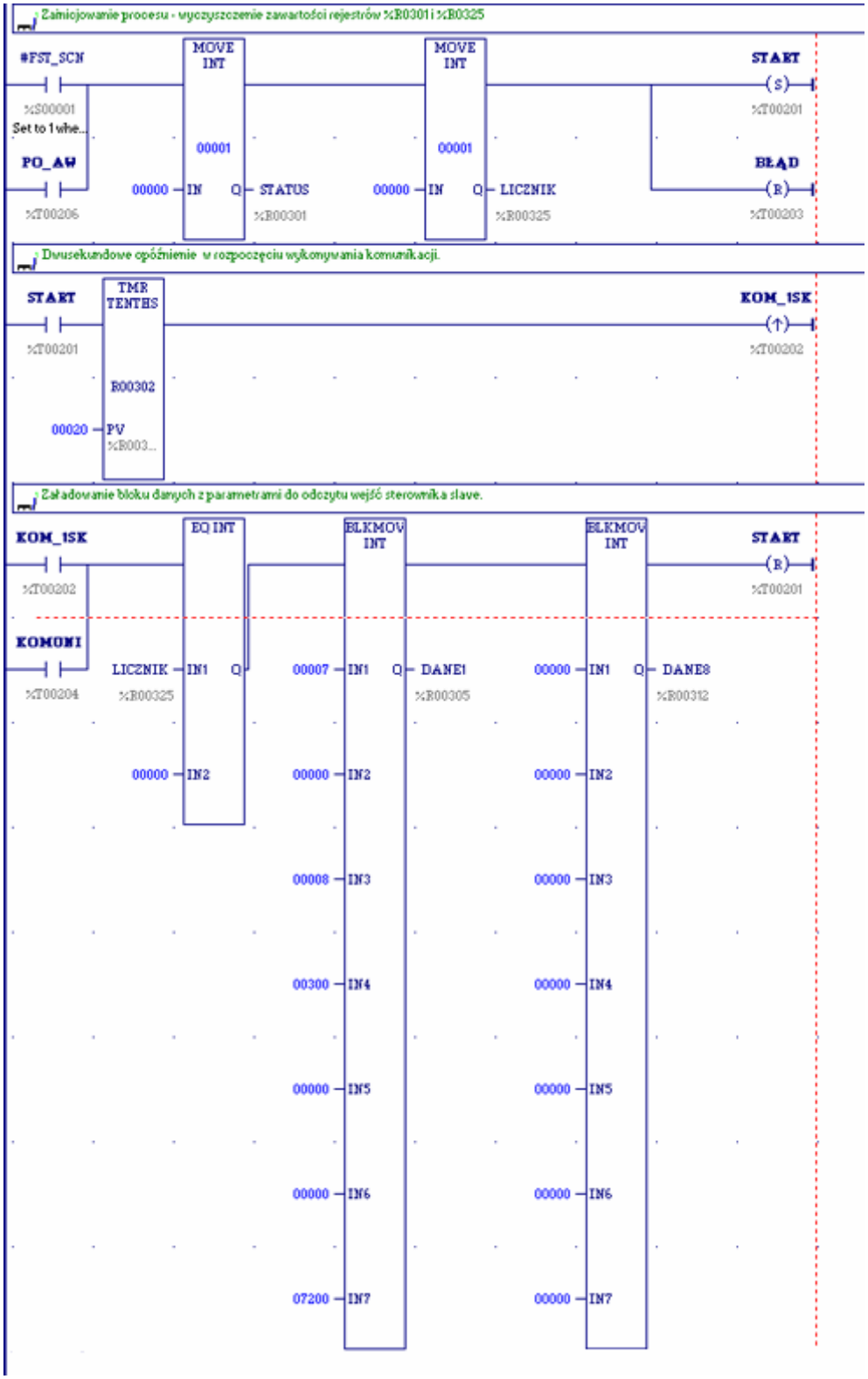
Zadanie 10 Odczyt daty i czasu z zegara kalendarzowego w sterowniku

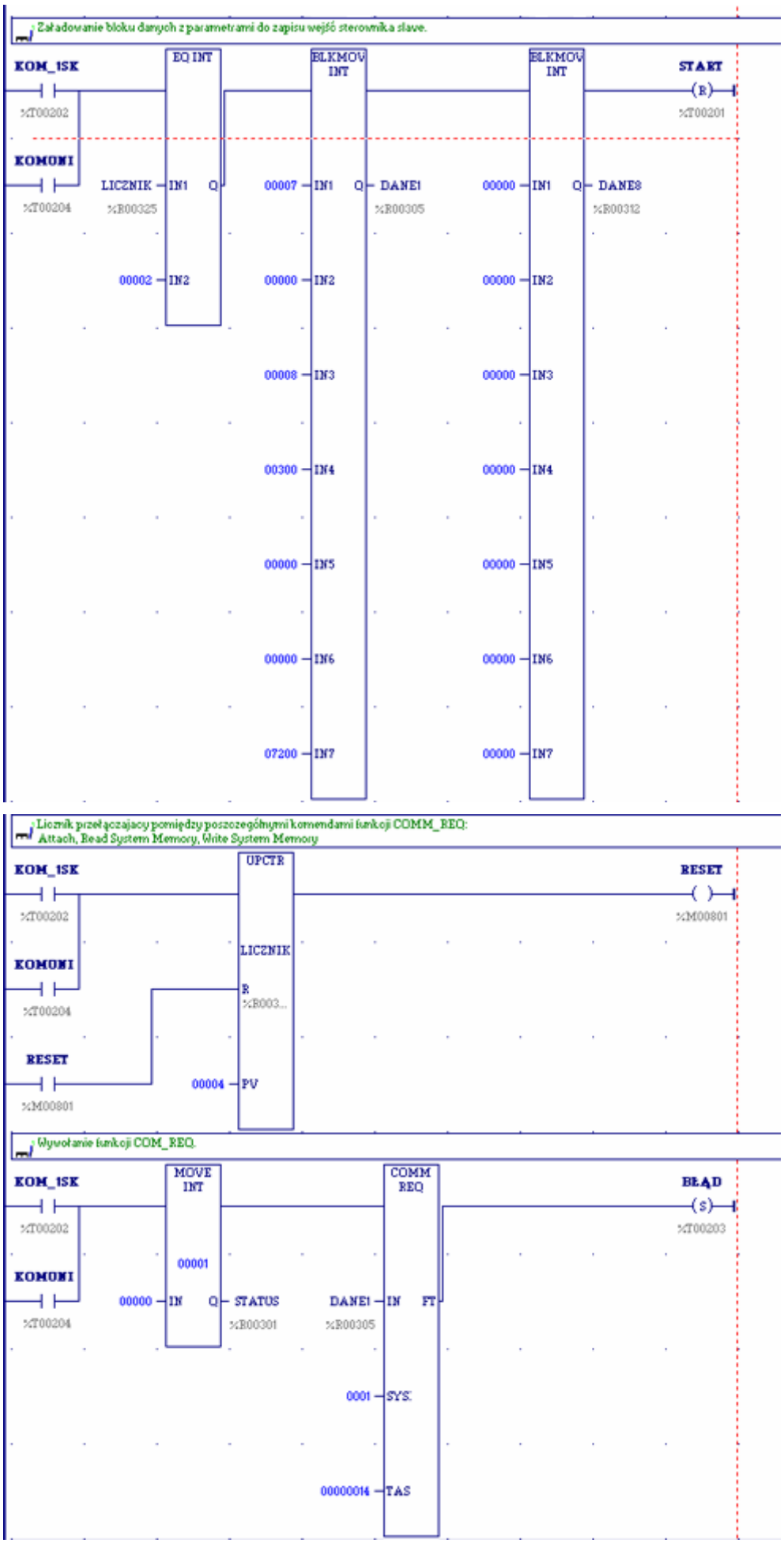


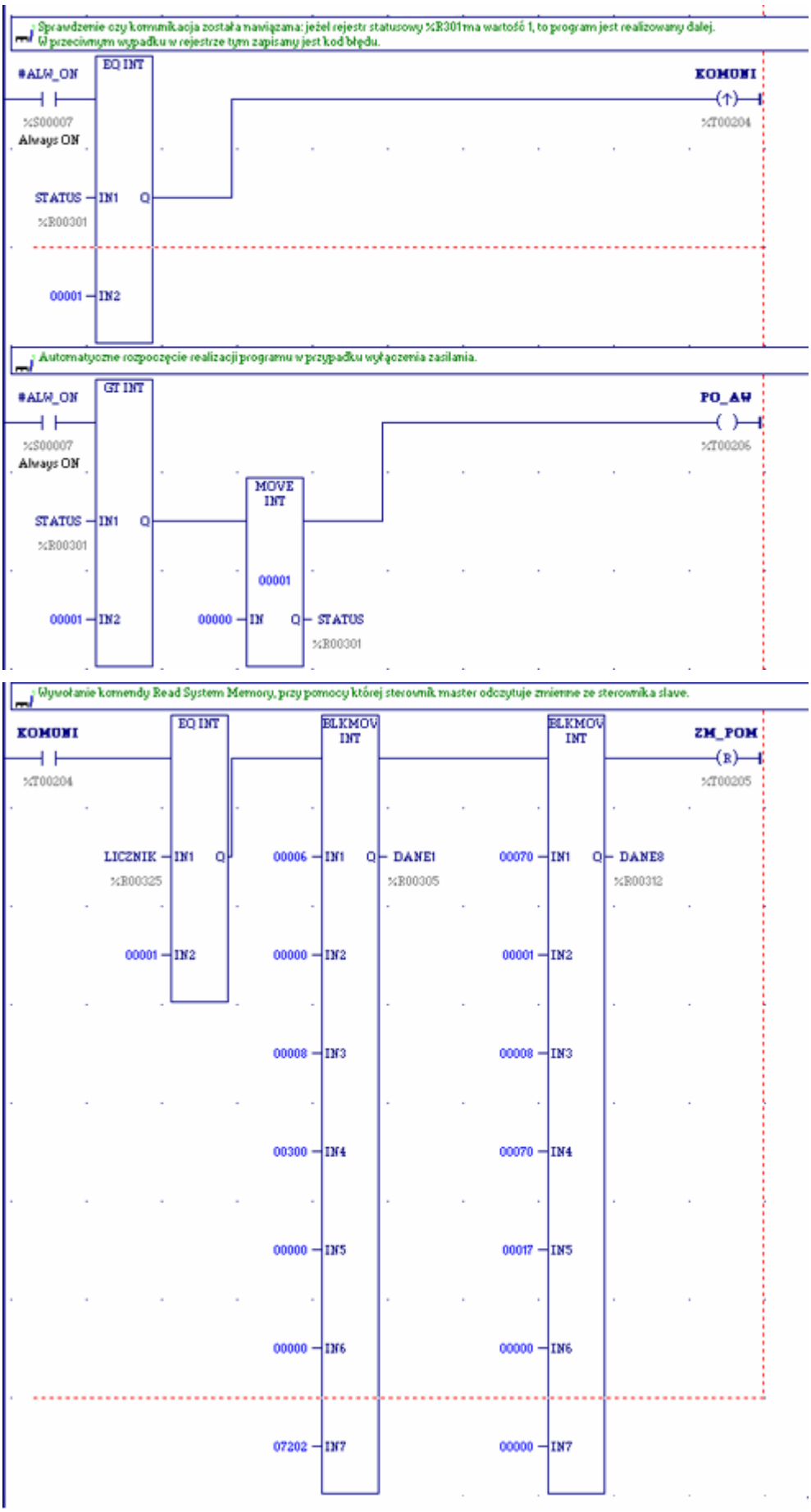
Zadanie 11 Sterowanie silnikami krokowymi

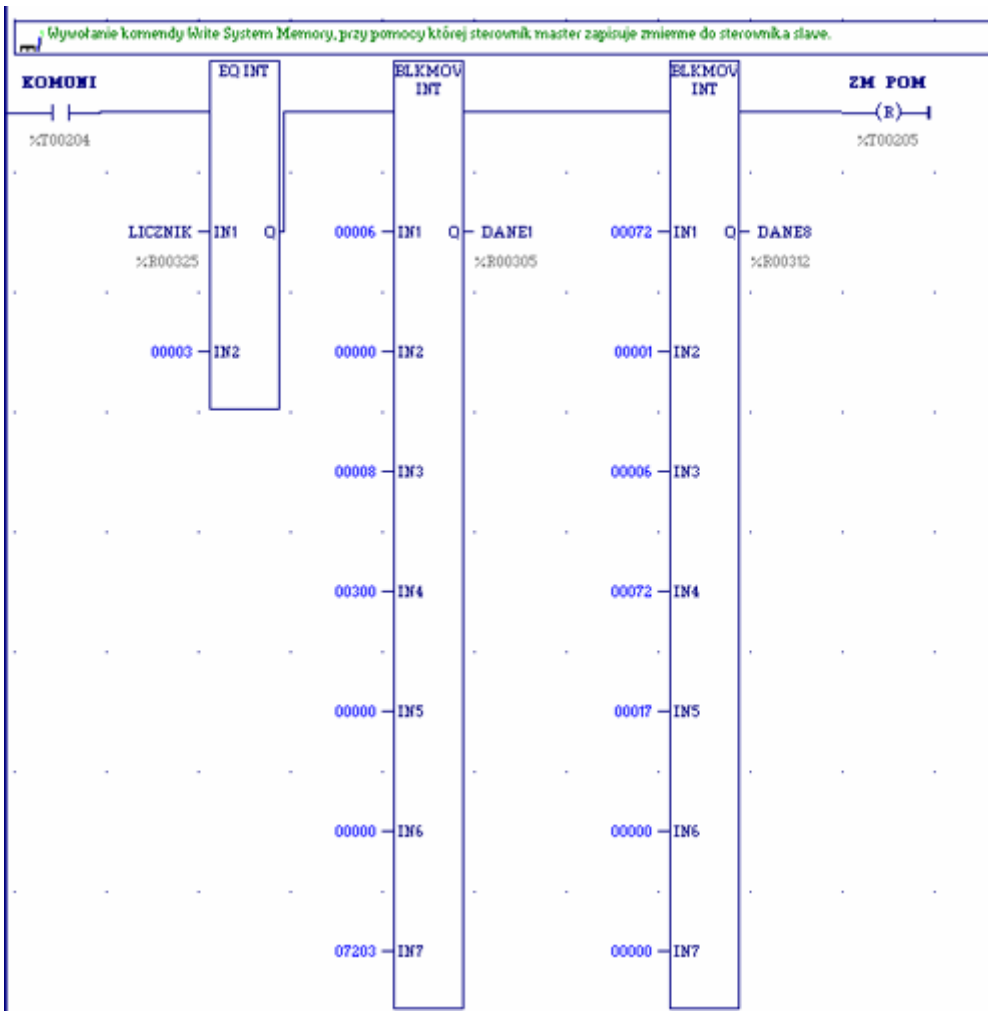


Zadanie 12 Komunikacja w protokole SNP

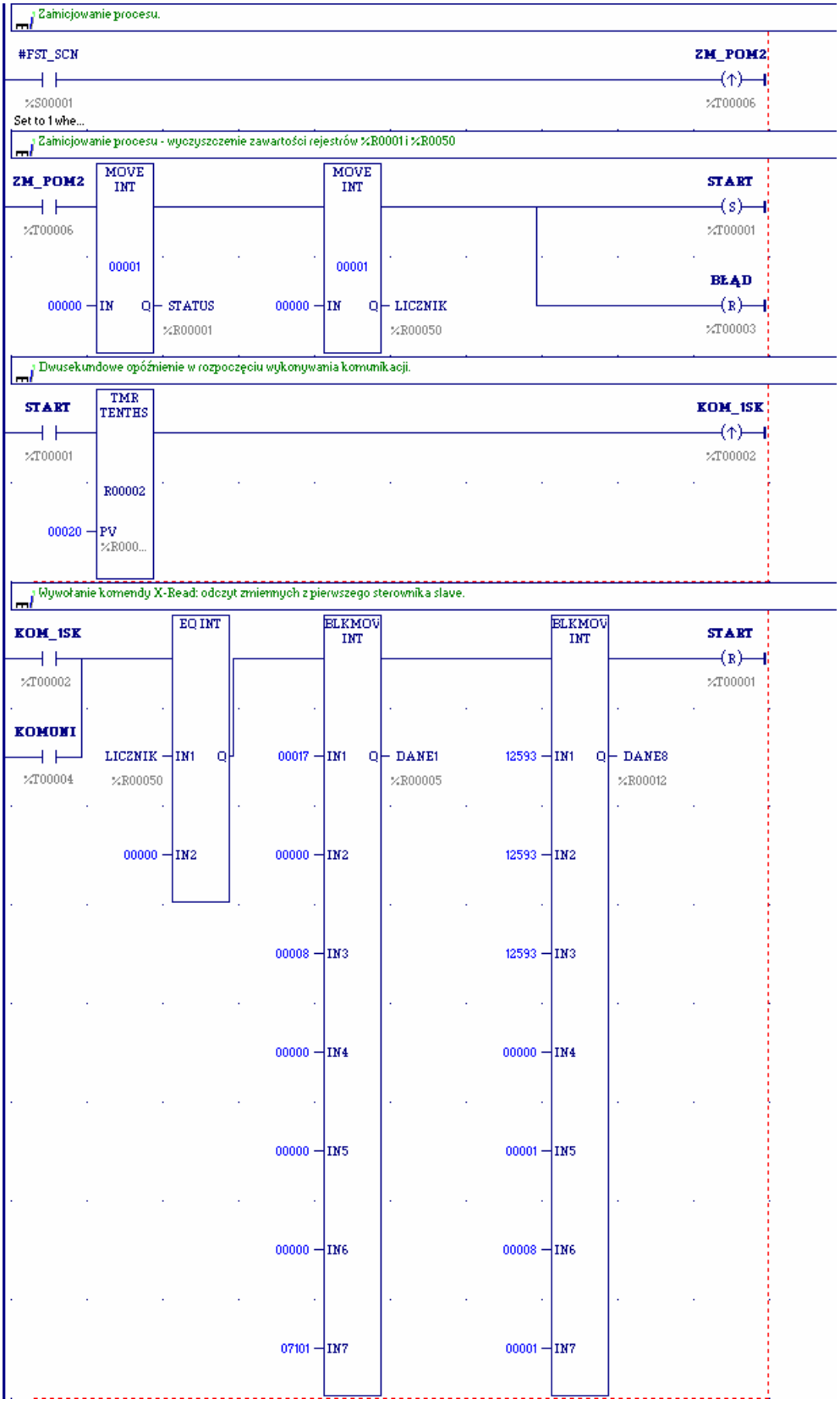


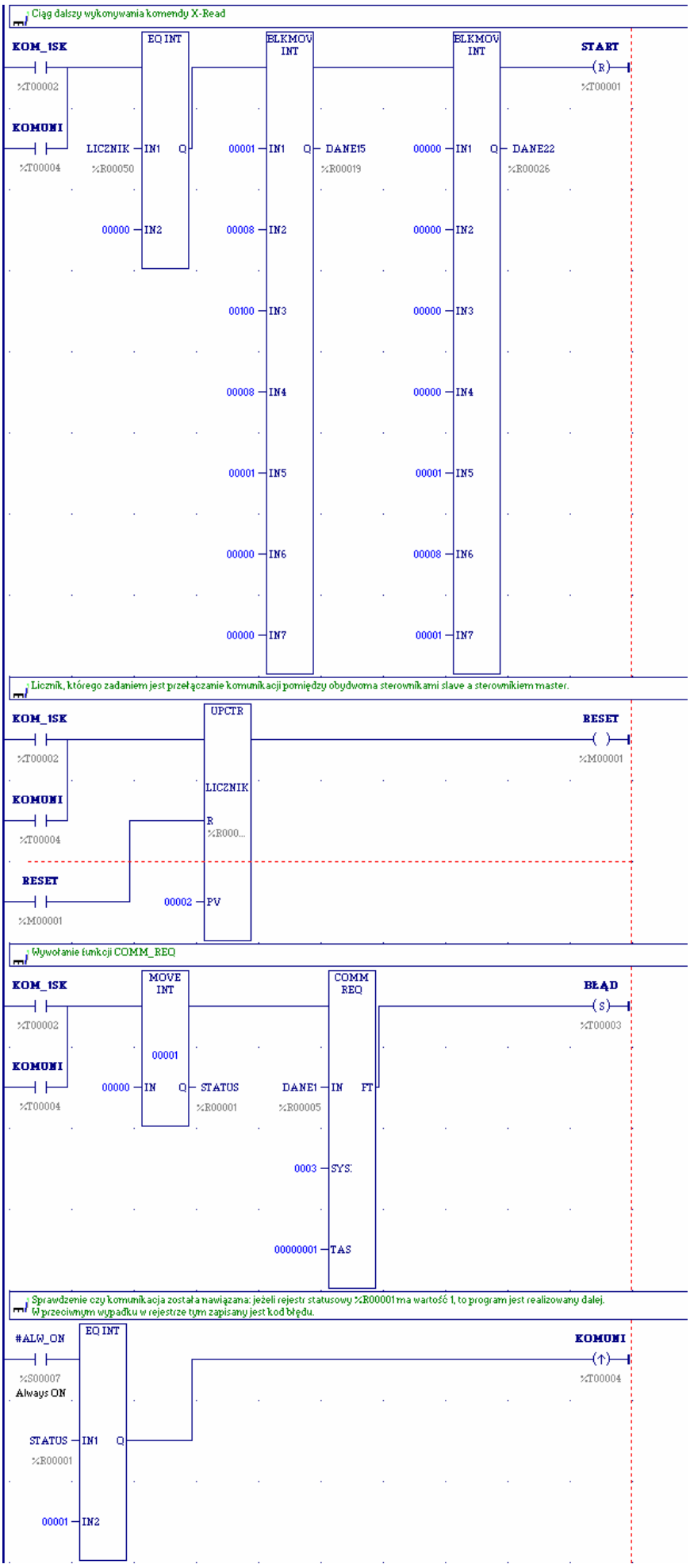

Załadowanie bloku danych z parametrami do zapisu wejść sterownika slave.
KOM_ISK
%T00202
KOMUNI
%T00204
EQ INT
LICZNIK
%R00325
IN1
Q
00002
IN2
BLKMOV INT
00007
IN1
Q
DANE1
%R00305
00000
IN2
00008
IN3
00300
IN4
00000
IN5
00000
IN6
07200
IN7
BLKMOV INT
00000
IN1
Q
DANE8
%R00312
00000
IN2
00000
IN3
00000
IN4
00000
IN5
00000
IN6
00000
IN7
START
(R)
%T00201
Licznik przełączający pomiędzy poszczególnymi komendami funkcji COMM_REQ:
Attach, Read System Memory, Write System Memory
KOM_ISK
%T00202
KOMUNI
%T00204
RESET
%M00801
UPCTR
LICZNIK
R
%R003...
00004
PV
RESET
( )
%M00801
Wywołanie funkcji COM_REQ.
KOM_ISK
%T00202
KOMUNI
%T00204
MOVE INT
00001
00000
IN
Q
STATUS
%R00301
COMM REQ
DANE1
%R00305
IN
FT
0001
SYS
00000014
TAS
BŁĄD
(S)
%T00203

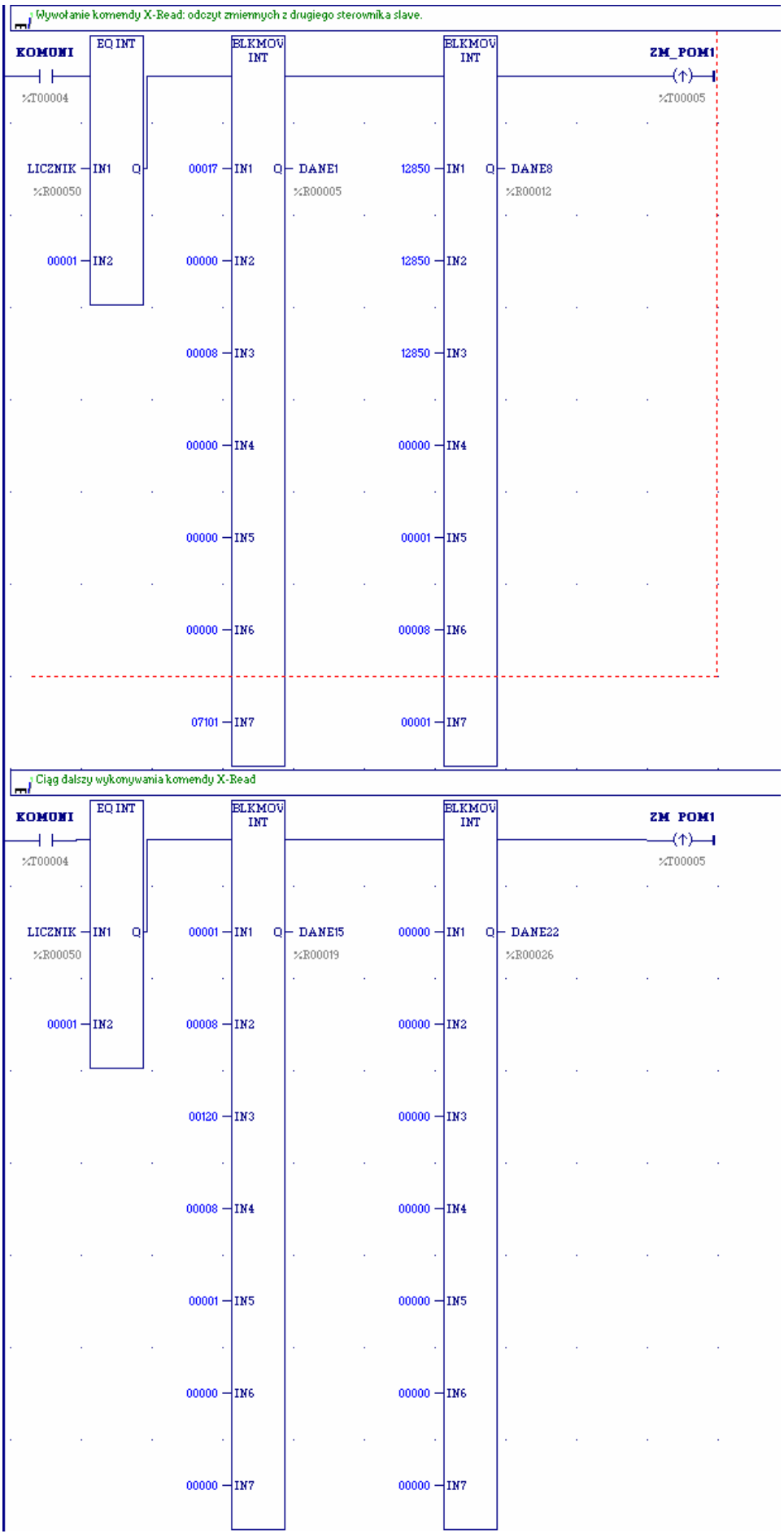

Sprawdzenie czy kommikacja została nawiązana: jeżel rejestr statusowy %R301ma wartość 1, to program jest realizowany dalej.
W przeciwnym wypadku w rejestrze tym zapisamy jest kod błędu.
#ALW_ON
%S00007
Always ON
EQ INT
STATUS
%R00301
IN1
Q
00001
IN2
KOMUNI
%T00204
Automatyczne rozpoczęcie realizacji programu w przypadku wyłączenia zasilania.
#ALW_ON
%S00007
Always ON
GT INT
STATUS
%R00301
IN1
Q
00001
IN2
MOVE INT
00001
00000
IN
Q
STATUS
%R00301
PO_AW
%T00206
Wywołanie komendy Read System Memory, przy pomocy której sterownik master odczytuje zmienne ze sterownika slave.
KOMUNI
%T00204
EQ INT
LICZNIK
%R00325
IN1
Q
00001
IN2
BLKMOV INT
00006
IN1
Q
DANE1
%R00305
00000
IN2
00008
IN3
00300
IN4
00000
IN5
00000
IN6
07202
IN7
BLKMOV INT
00070
IN1
Q
DANE8
%R00312
00001
IN2
00008
IN3
00070
IN4
00017
IN5
00000
IN6
00000
IN7
ZM_POM
%T00205


Wywołanie komendy Write System Memory, przy pomocy której sterownik master zapisuje zmienne do sterownika slave.
KOMUNI
%T00204
EQ INT
LICZNIK
%R00325
IN1
Q
00003
IN2
BLKMOV
INT
00006
IN1
Q
DANE1
%R00305
00000
IN2
00008
IN3
00300
IN4
00000
IN5
00000
IN6
07203
IN7
BLKMOV
INT
00072
IN1
Q
DANE8
%R00312
00001
IN2
00006
IN3
00072
IN4
00017
IN5
00000
IN6
00000
IN7
ZM POM
(R)
%T00205

Zadanie 13 Komunikacja w protokole SNP




Ciąg dalszy wykonywania komendy X-Read
KOM_ISK
%T00002
KOMUNI
%T00004
EQ INT
LICZNIK
%R00050
IN1
Q
IN2
00000
BLKMOV INT
00001
00008
00100
00008
00001
00000
00000
IN1
IN2
IN3
IN4
IN5
IN6
IN7
DANE15
%R00019
BLKMOV INT
00000
00000
00000
00000
00001
00008
00001
DANE22
%R00026
START
(R)
%T00001
Licznik, którego zadaniem jest przełączanie komunikacji pomiędzy obydwoma sterownikami slave a sterownikiem master.
KOM_ISK
%T00002
KOMUNI
%T00004
RESET
%M00001
UPCTR
LICZNIK
R
%R000...
00002
PV
RESET
( )
%M00001
Wywołanie funkcji COMM_REQ
KOM_ISK
%T00002
KOMUNI
%T00004
MOVE INT
00001
00000
IN
Q
STATUS
%R00001
COMM REQ
DANE1
%R00005
IN
FT
0003
SYS:
00000001
TAS
BŁĄD
(S)
%T00003
Sprawdzenie czy komunikacja została nawiązana: jeżeli rejestr statusowy %R00001 ma wartość 1, to program jest realizowany dalej.
W przeciwnym wypadku w rejestrze tym zapisany jest kod błędu.
#ALW_ON
%S00007
Always ON
EQ INT
STATUS
%R00001
IN1
Q
00001
IN2
KOMUNI
(↑)
%T00004

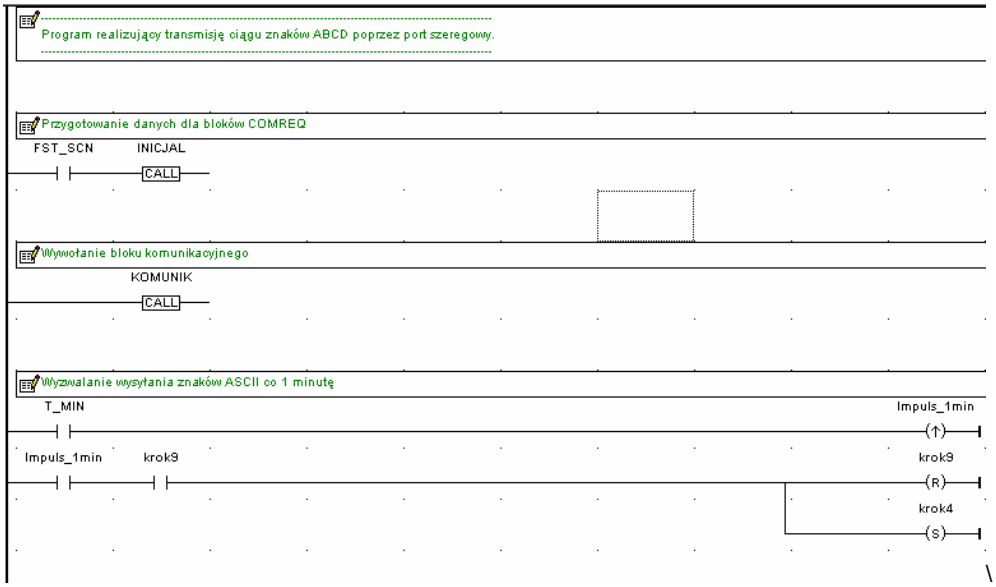

Wywołanie komendy X-Read: odczyt zmiennych z drugiego sterownika slave.
KOMUNI
%T00004
EQ INT
LICZNIK
%R00050
IN1
Q
00001
IN2
BLKMOV INT
00017
IN1
Q
DANE1
%R00005
00000
IN2
00008
IN3
00000
IN4
00000
IN5
00000
IN6
07101
IN7
BLKMOV INT
12850
IN1
Q
DANE8
%R00012
12850
IN2
12850
IN3
00000
IN4
00001
IN5
00008
IN6
00001
IN7
ZM_POM1
%T00005
Ciąg dalszy wykonywania komendy X-Read
KOMUNI
%T00004
EQ INT
LICZNIK
%R00050
IN1
Q
00001
IN2
BLKMOV INT
00001
IN1
Q
DANE15
%R00019
00008
IN2
00120
IN3
00008
IN4
00001
IN5
00000
IN6
00000
IN7
BLKMOV INT
00000
IN1
Q
DANE22
%R00026
00000
IN2
00000
IN3
00000
IN4
00000
IN5
00000
IN6
00000
IN7
ZM_POM1
%T00005

Zadanie 14 Przesyłanie danych przez port szeregowy

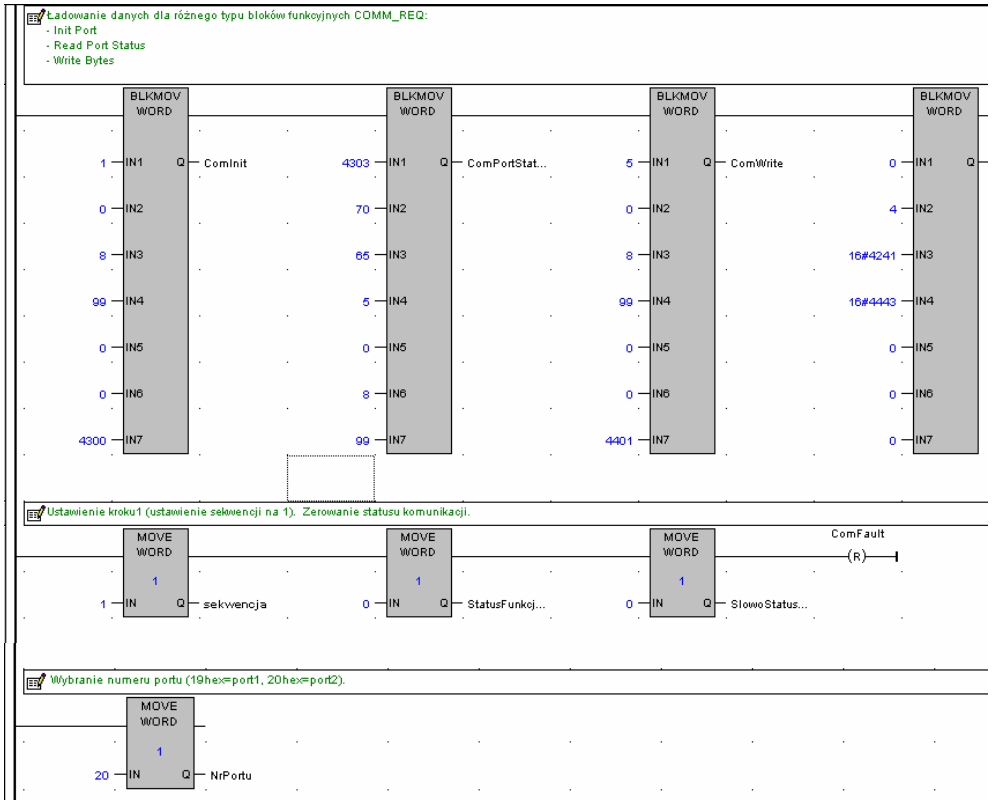
W podprogramie IINICJAL, w ostatnim szczeblu można port w sterowniku (19hex = port1, 20hex=port2).

Aby sprawdzić działanie programu można podłączyć do sterownika komputer z dowolnym oprogramowaniem typu terminal (np. HyperTerminal, Norton Terminal, SATERM, itp.).

Program główny:

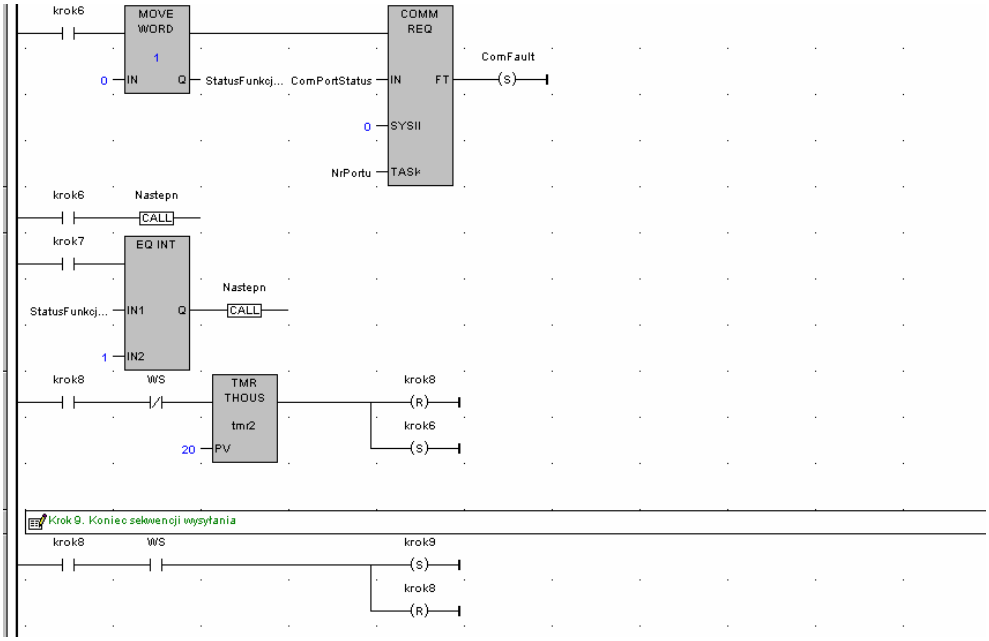


Podprogram INICJAL:

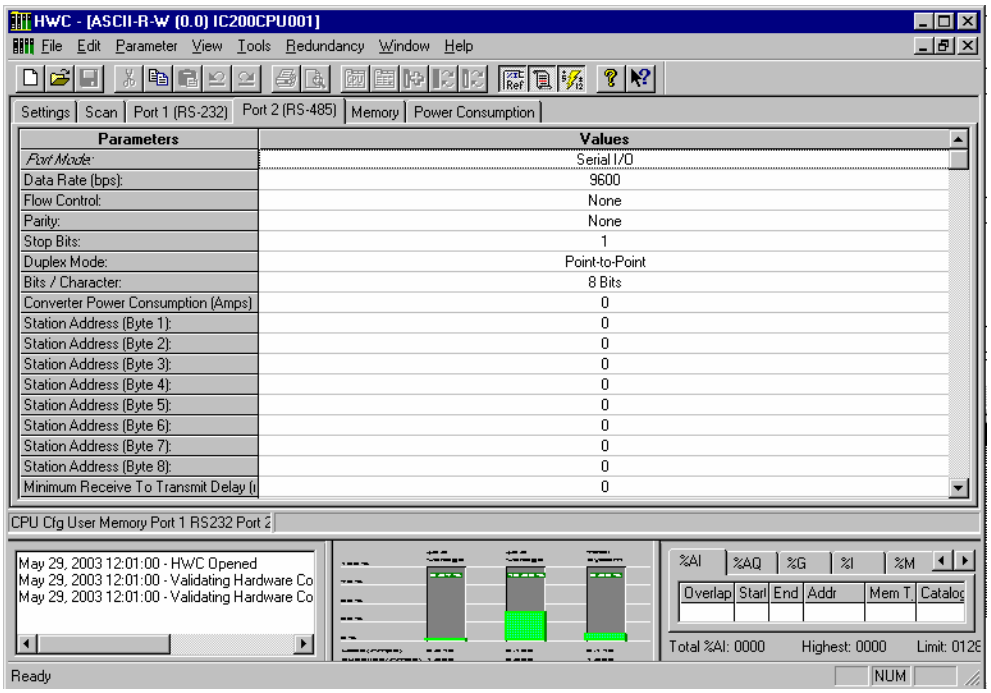


Podprogram KOMUNIK:

Podprogram NASTEMP:



Konfiguracja portu szeregowego



Tablica deklaracji zmiennych

Efekt działania programu obserwowany na terminalu SATERM (odebrano 3 paczki znaków ASCII):