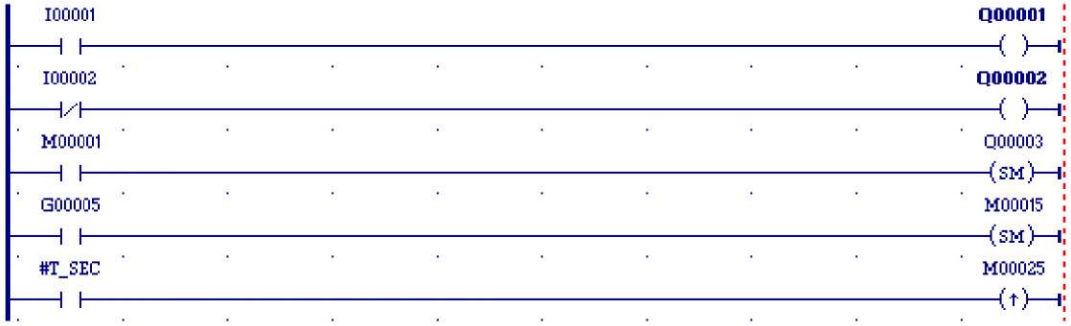
Przykłady użycia bloków funkcyjnych

Przykład 1. Elementy logiczne

Poniżej przedstawiono różne rodzaje przekaźników:



Przekaźnik I1 jest typu „styk otwarty" - przewodzi sygnał wtedy, gdy wartość logiczna przypisanej zmiennej jest 1. Przekaźnik I2 jest typu „styk zamknięty" - przewodzi sygnał, gdy przypisana dla niego zmienna ma wartość logiczną 0. Przekaźnik Q2 działa w ten sposób, że jego 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. Natomiast przekaźnik M15 różni się od Q3 tym, że stan jego jest „zapominany" po wyłączeniu 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.

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

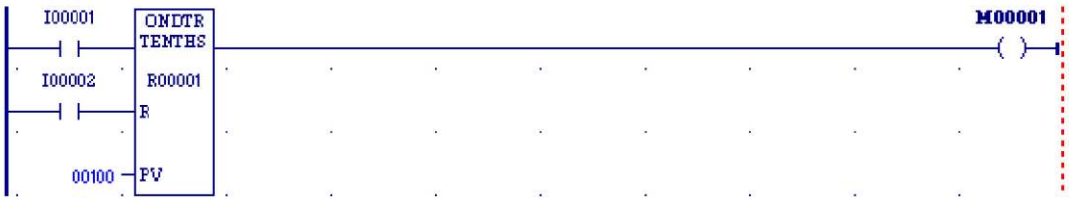
Przykład 2. Liczniki i przekaźniki czasowe – blok TMR

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

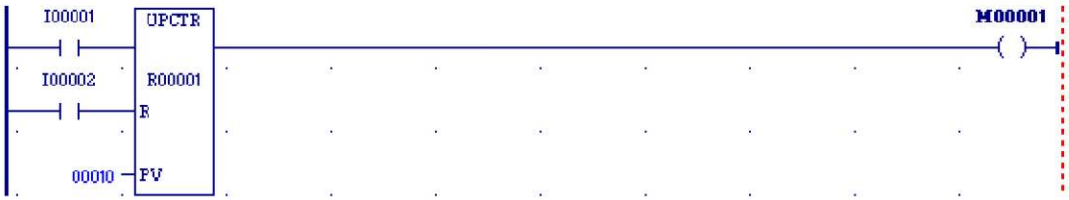
Program sygnalizujący, że sygnał aktywny na wejściu I1 trwał przynajmniej 10 sekund:



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

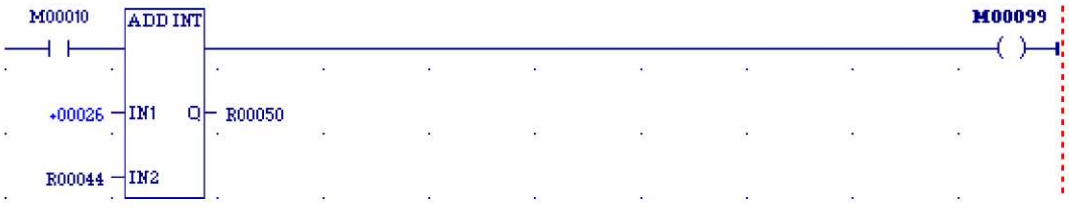
Program sygnalizujący, że do wejścia I1 dotarło przynajmniej 10 impulsów:



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

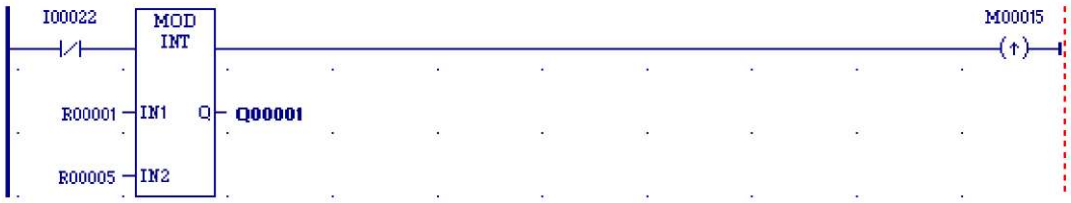
Dodawanie liczby 26 do liczby w rejestrze R44:



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:

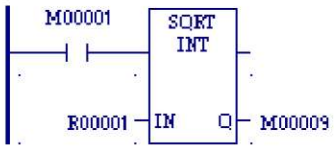
$$Q = I1 - (I1 \text{ DIV } I2) * I2$$

(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

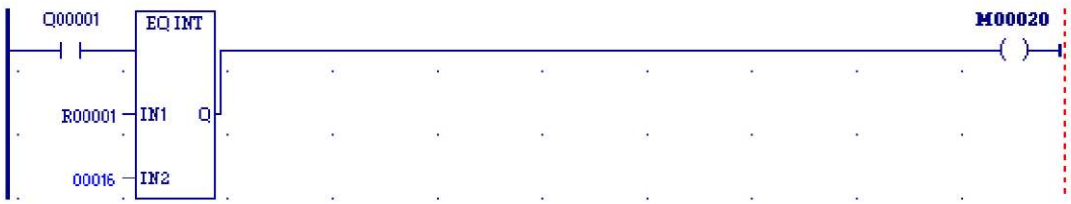
Obliczenie pierwiastka kwadratowego z liczby znajdującej się w rejestrze R1:



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

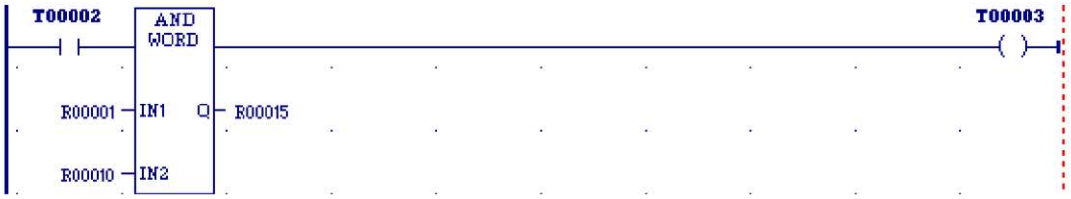
Sprawdzenie, czy liczba w rejestrze R1 to liczba 16:



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

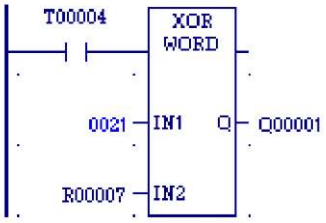
Operacja AND na dwóch słowach bitowych:



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

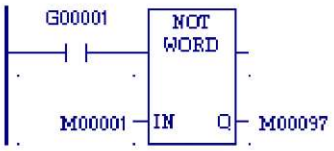
Sprawdzenie czy liczba w rejestrze R7 ma pięć najmniej znaczących bitów postaci: 10101 i które bity odbiegają od tego wzorca:



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.