INTEX | Centrum Szkoleniowe Systemów Automatyki

PODRĘCZNIK

ĆWICZENIA W STL

INTEX
Centrum Szkoleniowe
Systemów Automatyki

Ważne wskazówki dotyczące bezpieczeństwa!

Przykładowe aplikacje i rozwiązanie zawarte w tym podręczniku należy traktować jako uproszczone i niekompletne pod względem przedstawionego schematu oraz warunków, jakie należy uwzględnić w rzeczywistej aplikacji. INTEX Sp. z o.o. nie odpowiada za poprawność i kompletność aplikacji tworzonych przez uczestników szkolenia. Ponieważ opisane w podręczniku ćwiczenia w trakcie szkolenia są przeprowadzane z wykorzystaniem dedykowanego stanowiska szkoleniowego, niezależnie od sposobu ich realizacji w żadnym wypadku nie dojdzie do uszkodzenia mienia ani zranienia osób.

Uczestnik szkolenia/użytkownik dokumentacji musi jednak mieć świadomość, że każda ingerencja w system sterowania maszyną/instalacją wiąże się z dużym zagrożeniem!

W wyniku ingerencji, istniejące funkcje bezpieczeństwa mogą zostać wyłączone lub pominięte. Część instalacji może zostać w sposób niezamierzony lub niebezpieczny uruchomiona, zatrzymana, zasilona lub wprawiona w ruch. Zdarzenia te w następstwie mogą doprowadzić do przerwy w produkcji, szkód materialnych czy też niebezpieczeństwa zranienia lub śmierci personelu obsługi.

Każda ingerencja w system sterowania maszyną/instalacją podlega z tego powodu szczególnym wymaganiom bezpieczeństwa i dlatego może być przeprowadzona tylko i wyłącznie pod ścisłym nadzorem doświadczonego i odpowiednio uprawnionego personelu technicznego!

Copyright © by INTEX Sp. z o.o.
Wszelkie prawa zastrzeżone.

Żadna część tej pracy nie może być powielana i rozpowszechniana w jakiejkolwiek formie i w jakikolwiek sposób włącznie z fotokopiowaniem lub przy użyciu innych systemów, bez pisemnej zgody wydawcy.
Autor dołożył wszelkich starań, by zawarte w tej pracy informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich.
Występujące w tekście zastrzeżone znaki firm są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.
Wszelkie nazwy własne, zastrzeżone znaki towarowe i handlowe należące do podmiotów trzecich, są używane przez firmę INTEX Sp.z o.o. wyłącznie w celach identyfikacyjnych i informacyjnych.

I.	Wprowadzenie	4
II.	Dlaczego tworzyć program jako lista instrukcji?	4
III.	Format zapisu programu w STL	5
IV.	Połączenie szeregowe – operacja AND	7
V.	Połączenie równoległe – funkcja OR	11
VI.	Operacje grupowania	15
VII.	Ustawianie i kasowanie w zapisie STL – przerzutniki S, R	17
VIII.	Wykrywanie zbocza – FP, FN	18
IX.	Ustawienie i kasowanie RLO – funkcje SET i CLR	19
X.	Negacja bieżącego stanu RLO – funkcja NOT	21
XI.	Wywoływanie bloków programowych – UC, CC, CALL	21
XII.	Zakończenie bloku – BEC, BEU	24
XIII.	Przenoszenie danych – L, T	25
XIV.	Zliczanie zdarzeń – liczniki	28
XV.	Operacje porównywania – komparatory	30
XVI.	Funkcje arytmetyczne	32
XVII.	Realizacja opóźnienia – układy czasowe	33
XVIII.	Operacje skoku – JU, JC, JCN	37
XIX.	Rozwiązania zadań	40

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl

NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

I. Wprowadzenie

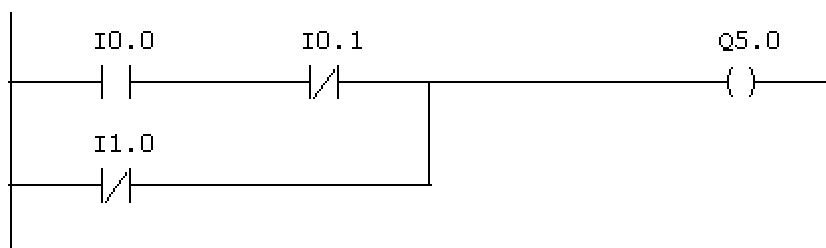
Celem tego podręcznika jest podsumowanie i ugruntowanie wiadomości w zakresie zapisu oraz analizy programu w języku STL wykorzystywanym w sterownikach SIEMENS SIMATIC.

Zestaw ćwiczeń przeznaczony jest do samodzielnej nauki, nie wymaga dostępu do komputera wyposażonego w oprogramowanie STEP7.

II. Dlaczego tworzyć program jako lista instrukcji ?

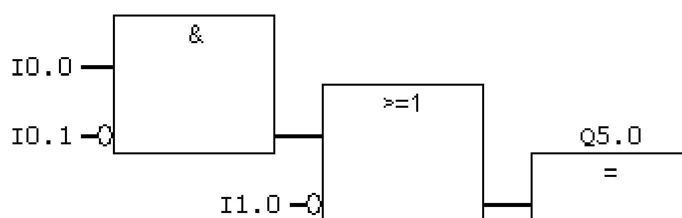
Środowisko STEP7 w podstawowej wersji pozwala na zapis programu na jeden z trzech sposobów.

Pierwszy sposób, najbardziej intuicyjny dla osób mających doświadczenie w projektowaniu obwodów elektrycznych, to schemat drabinkowy (stykowy) – LAD (**LA**dder **D**iagram). Wykorzystuje się tutaj elementy, które nawiązują do obwodów elektrycznych.



Przykład programu zapisanego w LAD

Drugi sposób to schemat funkcyjny – FBD (**F**unction **B**lock **D**iagram)**.** Ten sposób reprezentacji nawiązuje do projektowania układów elektronicznych. Programista ma więc do czynienia z bramkami AND, OR, XOR oraz bloczkami realizującymi określone funkcje.



Przykład programu w FBD

Trzeci sposób to STL (**ST**atement **L**ist), w swojej formie przypomina assembler (język instrukcji procesora). Ten sposób programowania będzie najbardziej intuicyjny dla osób, które mają doświadczenie w programowaniu w językach niskiego poziomu np. w programowaniu mikrokontrolerów.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

```
A      I      0.0
AN     I      0.1
ON     I      1.0
=      Q      5.0
```

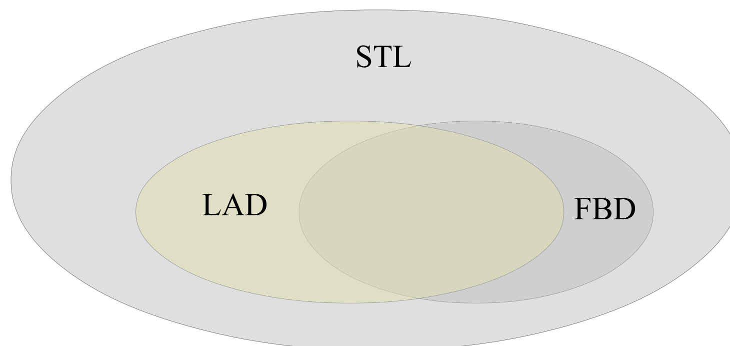
Przykład programu w STL

Który język zapisu programu wybrać ?

Jeżeli programiście pozostawiono wybór (nie jest narzucony przez standard zakładu, ani przez wymagania zdefiniowane przez klienta) można wybrać ten, w którym autor programu czuje się najswobodniej.

Istnieją jednak pewne ograniczenia. Dużą część programów można zapisać w LAD lub FBD, ale istnieją pewne operacje (np. adresowanie pośrednie), w których programista nie ma wyboru i musi skorzystać z zapisu STL.

Dostępność funkcji w poszczególnych językach można przedstawić na podstawie poniższego diagramu:



Obszary LAD i FBD w przeważającej części pokrywają się.
Jednak wszystko to, co można zrealizować w językach LAD i FBD, znacznie więcej, można zrobić w STL

Zalety języka STL
+ większe możliwości niż LAD czy FBD
+ większa swoboda programowania niż w językach graficznych
+ szybsze pisanie kodu (po osiągnięciu pewnej wprawy)
+ możliwość umieszczania komentarza do każdej linii kodu

Wady STL
- trudniejsza, mniej intuicyjna analiza kodu programu
- wymagane większe doświadczenie programisty, a przede wszystkim od osoby pracującej z programem (utrzymanie ruchu)

III. Format zapisu programu w STL

W linii kodu w STL można wyróżnić następujące elementy:

etyk: rozkaz parametr // komentarz do operacji

Linia kodu będzie najczęściej zawierać **rozkaz.** Może to być rozkaz bezparametrowy, np.:

```
SET        //ustaw RLO = '1'
```

częściej jednak wykorzystywane są rozkazy występują z **parametrami,** np.:

```
A   I 1.2  //sprawdź czy I1.2 = 'true'
```

jeżeli dana linia kodu ma być oznaczona **etykietą,** to jej deklaracja pojawi się z lewej strony, np.:

```
etyk:      A   I 1.2
```

Każda linia kodu może kończyć się **komentarzem,** dodanie komentarza pozwala na zwiększenie czytelności programu. Komentarz rozpoczyna się przy pomocy dwóch ukośnych kresek, np.:

```
etyk:      A   I 1.2       // sprawdź wejście START
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

IV. Połączenie szeregowe – operacja AND

Jedną z podstawowych funkcji logicznych iloczyn (AND) – funkcji tej odpowiada na schemacie stykowym szeregowe połączenie elementów. Funkcja ta zostanie przedstawiona na podstawie poniższego przykładu.

AND
połączenie szeregowe -
i - iloczyn - koniunkcja
są to określenia opisujące tę samą funkcję

Przykład: Układ wyzwalający prasę

Aby uchronić operatora prasy przed skaleczeniem rąk zastosowano specjalny układ wyzwalający. Wyzwolenie prasy może nastąpić tylko i wyłącznie w momencie równoczesnego naciśnięcia dwóch przycisków S1 i S2 rozmieszczonych tak, aby nie było możliwe naciśnięcie obydwu przycisków jedną ręką.

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych elementów:

Element	Funkcja	Adres
S1	Przycisk lewej ręki	I 0.3
S2	Przycisk prawej ręki	I 0.4
K1	Stycznik wyzwalający prasę	Q 4.3

Logika, jaka powinna być użyta do zrealizowania tego zadania może być przedstawiona za pomocą poniższego schematu elektrycznego:

Schemat elektryczny odpowiadający funkcji AND

Przypomnienie

Oznaczenie **I** pochodzi z języka angielskiego – **I**nput i oznacza **wejście**.

Oznaczenie **Q** pochodzi z języka angielskiego – **O**utput i oznacza **wyjście** (litera O zarezerwowana jest dla operacja OR).

Oznaczenie **M** jest odwołaniem do **pomocniczego obszaru pamięci** (ang. marker lub memory).

Należy więc sprawdzić czy wejście I 0.3 (S1) jest w stanie **wysokim oraz** czy wejście I 0.4 (S2) także jest w stanie **wysokim**.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Jeżeli te dwa warunki są jednocześnie spełnione należy wysterować (stan wysoki) wyjście Q 4.3 (K1).

Program w STL wygląda następująco:

```
A     I 0.3    // Jeżeli wejście I 0.3 jest w stanie wysokim
A     I 0.4    // i wejście I 0.4 jest w stanie wysokim
=     Q 4.3    // wtedy wysteruj wyjście Q 4.3
```

Jak to działa dokładnie ?

Operacja A realizuje iloczyn logiczny pomiędzy aktualnym stanem RLO oraz aktualnym stanem parametru (np I0.4). Wynik tej operacji jest zapisywany do RLO i dostępny dla następnej operacji.

Dla pierwszej operacji ma miejsce bezpośrednie przepisanie wyniku sprawdzenia do RLO.

W naszym przykładzie operacja

> **RLO**
> **R**esult of **L**ogic **O**perations
>
> Bieżący wynik operacji logicznych:
> **stan wysoki** - **prawda - „1”**
> lub
> **stan niski - fałsz - „0”**

```
A     I 0.3    // Jeżeli wejście I 0.3 jest w stanie wysokim
```

przepisuje stan wejścia I 0.3 bezpośrednio do RLO (ponieważ jest to pierwsza operacja).

W następnej operacji dokonywana jest operacja logiczna AND (iloczyn, połączenie szeregowe) pomiędzy RLO oraz stanem wejścia I 0.4 - wynik tej operacji jest zapisywany do RLO:

```
A     I 0.4    // i wejście I 0.4 jest w stanie wysokim
```

W ostatniej operacji bieżący stan RLO przepisany jest na wyjście Q 4.3:

```
=     Q 4.3    // wtedy wysteruj wyjście Q 4.3
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Przykład: Sygnalizacja uszkodzenia czujnika

W zbiorniku zainstalowane są dwa czujniki poziomu. Jeżeli „zalany” zostanie dolny czujnik CZ1, to zwróci on stan wysoki. Jeżeli poziom cieczy w zbiorniku osiągnie poziom górnego czujnika - CZ2 - zwróci on stan wysoki.

Należy wykryć sytuację, gdy dolny czujnik zwraca stan niski, natomiast górny czujnik zwraca stan wysoki. Jest to sytuacja sygnalizująca uszkodzenie któregoś z czujników.

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych elementów:

Element	Funkcja	Adres
CZ1	Dolny czujnik poziomu	I 10.0
CZ2	Górny czujnik poziomu	I 10.1
USZKODZENIE	Sygnał uszkodzenia czujnika	Q 0.2

Z treści zadania wynika, że należy sprawdzić, czy sygnał CZ2 jest aktywny (stan **wysoki**) **oraz** jednocześnie CZ1 jest nieaktywne (stan **niski**).
Jeżeli zaistnieje taka sytuacja należy aktywować (stan **wysoki**) wyjście USZKODZENIE.

Ponieważ obydwa warunki muszą być spełnione równocześnie, należałby zastosować funkcję AND czyli szeregowe połączenie elementów.

Jednak jeden z warunków jest spełniony kiedy wejście (I 10.0) jest w stanie niskim, co w przypadku operacji iloczynu A (A I10.0) spowoduje zapisanie stanu niskiego do RLO.

Rozwiązanie stanowi **negacja** stan sygnału wejściowego, co uzyskamy poprzez zastosowanie operacja AN (AND NOT):

```
A     I 10.1          // Jeżeli górny czujnik zwraca stan wysoki
AN    I 10.0          // i dolny czujnik zwraca stan niski
=     Q 0.2           // wtedy sygnalizuj uszkodzenie
```

Pamiętaj !

Negacja stanu sygnału wejściowego w połączeniu szeregowym uzyskiwana jest poprzez operację AN.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Zadanie 1: Zezwolenie na jazdę przenośnika

Zezwolenie na jazdę przenośnika taśmowego w trybie ręcznym zawiera szereg warunków: aby przenośnik mógł być wysterowany należy zagwarantować, że wyłącznik alarmowy ESTOP jest nieaktywny, przycisk JAZDA jest naciśnięty oraz kluczyk AUTO_MANU jest w pozycji pracy ręcznej (MANUal).

Jeżeli każdy z tych warunków zostanie spełniony, wtedy zostanie wysterowany stycznik K23 sterujący napędem przenośnika.

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych elementów:

Element	Funkcja	Adres
ESTOP	Wyłącznik awaryjny (normalnie zwarty)	I 14.1
JAZDA	Start przenośnika (normalnie otwarty)	I 12.4
AUTO_MANU	Tryb pracy przenośnika (tryb ręczny = 0)	I 2.7
K23	Stycznik sterujący napędem przenośnika	Q 0.2

Proponowane rozwiązanie znajduje się na końcu podręcznika.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

V. Połączenie równoległe – funkcja OR

Drugą podstawową funkcją jest suma logiczna (OR) odpowiadajaca połączeniu równoległemu. Funkcja ta również zostanie omówiona na podstawie przykładu.

OR
połączenie równoległe
- lub - suma - alternatywa
są to określenia opisujące tę samą funkcję

Przykład: Sterowanie wentylatorem

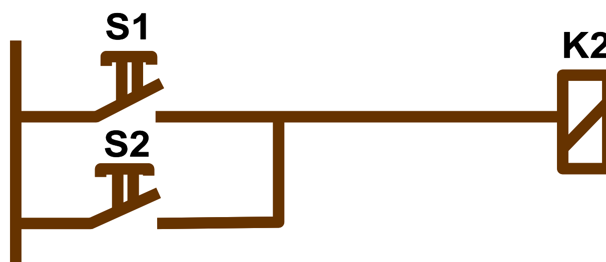
W hali zainstalowane są dwa generatory spalinowe. Praca generatorów jest nadzorowana przez czujniki. Jeżeli generator pracuje czujnik zwraca stan wysoki. Praca pierwszego generatora sygnalizowana jest czujnikiem S1, zaś drugiego przez czujnik S2.

W przypadku pracy któregokolwiek z generatorów powinien pracować wentylator K2 zapewniający prawidłowe przewietrzenie hali.

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych elementów:

Element	Funkcja	Adres
S1	Generator 1	I 0.1
S2	Generator 2	I 0.2
K2	Wentylator	Q 8.3

Wentylator powinien pracować, gdy pracuje pierwszy lub drugi generator (ewentualnie obydwa). Logika jaka powinna być użyta do zrealizowania tego zadania może być przedstawiona za pomocą poniższego schematu elektrycznego:



Schemat elektryczny odpowiadający funkcji sumy logicznej (OR)

Funkcja logiczna, która zrealizuje taką logikę to OR, czyli połączenie równoległe.

Program w języku STL opisujący tę logikę ma następującą postać:

```
O     I 0.1      // Jeżeli wejście I 0.1 jest w stanie wysokim
O     I 0.2      // lub wejście I 0.2 jest w stanie wysokim
=     Q 8.3      // wtedy wysteruj wyjście Q 8.3
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Przykład: Przenośnik taśmowy

Przenośnik taśmowy transportujący palety z magazynu do stanowiska pakowania napędzany silnikiem uruchamianym przez stycznik K1.

Silnik jest uruchamiany po naciśnięciu przycisku ZAŁ – od tego momentu silnik pracuje bez względu na stan przycisku ZAŁ (naciśnięty, czy też zwolniony, działanie przycisku powinno być podtrzymywane w programie). Praca silnika zostaje przerwana po dostarczeniu palety na wymagane miejsce, sygnalizowane wysokim stanem czujnika NOWA.

Silnik można wyłączyć w dowolnym momencie naciskając przycisk WYŁ.

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych elementów:

Element	Funkcja	Adres
ZAŁ	Przycisk załączający przenośnik (normalnie otwarty)	I 1.0
WYŁ	Przycisk wyłączający przenośnik (normalnie zwarty)	I 1.1
NOWA	Wykrywanie osiągnięcie punktu rozładunkowego	I 1.3
K1	Stycznik sterujący pracą napędu	Q 4.3

Program można napisać zaczynając od najprostszej logiki - jeżeli wciśnięty jest przycisk ZAŁ wtedy wysterowany jest napęd poprzez stycznik K1:

```
A     I 1.0     // jeżeli ZAŁ wciśnięty
=     Q 4.3     // uruchom K1
```

Do tak stworzonego programu należy dopisać warunek **podtrzymania** sygnału. Warunkiem pracy napędu jest naciśnięty przycisk albo dotychczasowa praca silnika:

```
O     I 1.0     // ZAŁ wciśnięty
O     Q 4.3     // lub K1 pracował
=     Q 4.3     // uruchom K1
```

Dzięki takiemu programowi przenośnik wystartuje po naciśnięciu przycisku ZAŁ i nawet po jego zwolnieniu, silnik będzie dalej pracować – bez końca.

Należy teraz uwzględnić w programie czujnik NOWA. Jeżeli ten czujnik zwraca „0” oznacza to, że paleta nie została wykryta, a więc przenośnik powinien pracować dalej. Stan „1” czujnika NOWA oznacza wykrycie palety i powinien spowodować zatrzymanie napędu. W tym programie został wypracowany warunek dla **pracy** przenośnika, a więc należy dodać kolejny warunek, który będzie sprawdzał, czy czujnik NOWA zwraca „0”:

```
O     I 1.0     // ZAŁ wciśnięty
O     Q 4.3     // lub K1 pracował
AN    I 1.3     // i NOWA = 0(czujnik nie wykrywa palety)
=     Q 4.3     // uruchom K1
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Ostatni warunek jaki należy dopisać, to sprawdzenie styku WYŁ. Warunkiem pracy przenośnika jest niewciśnięty przycisk WYŁ, jest to styk normalnie zwarty, a więc jeżeli nie jest aktywny na wejściu sterownika pojawia się stan wysoki:

```
O     I 1.0      // ZAŁ wciśnięty
O     Q 4.3      // lub K1 pracował
AN    I 1.3      // i NOWA = 0 (czujnik nie wykrywa palety)
A     I 1.1      // i WYŁ nie wciśnięty
=     Q 4.3      // uruchom K1
```

Jest to już końcowe rozwiązanie tego problemu.

Jako element pamięci zastosowano układ podtrzymujący z dominującym wejściem wyłączającym.

Dominacja wyłączania oznacza, że jeżeli jednocześnie będzie wciśnięty przycisk ZAŁ, jak i WYŁ, to w tym rozwiązaniu przenośnik nie będzie pracować.

Pamiętaj !

Podobnie jak w zapisie LAD, kolejność elementów w połączeniu równoległym nie ma znaczenia, operacje OR w STL mogą być zapisane w dowolnej kolejności.

Bezpośrednie przepisanie wyniku sprawdzenia do RLO ma miejsce dla tak zwanego „pierwszego odpytania" czyli dla rozkazu, który znajduje się na początku programu, lub po rozkazie unieważniającym stan RLO.

Dlatego program

```
O     I 1.0
O     Q 4.3
=     Q 4.3
```

lub

```
A     I 1.0
O     Q 4.3
=     Q 4.3
```

realizują tę samą logikę.

Negacja stanu sygnału wejściowego w połączeniu równoległym uzyskiwana jest poprzez operację **ON**.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Zadanie 2: Rejestracja zdarzeń

Lampka ALARM sygnalizuje wystąpienie sytuacji awaryjnej w systemie ogrzewania. Sytuacje te są wykrywane przez czujniki PALIWO, TEMP oraz SPALINY (aktywne stanem **niskim**).

Sygnalizator powinien zapamiętać wystąpienie sytuacji alarmowej do momentu potwierdzenia jej przyjęcia przez dyspozytora. Alarm może zostać skasowany (lampka ALARM zostanie wyłączona) pod warunkiem zaniku przyczyny wywołania alarmu przy pomocy przycisku POTW (przycisk normalnie otwarty).

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych sygnałów:

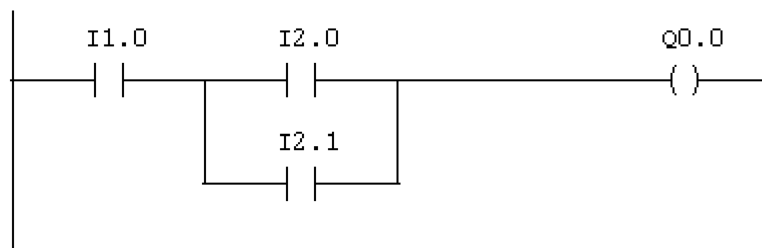
Element	Funkcja	Adres
PALIWO	Sygnalizator wyczerpania oleju opałowego (normalnie zwarty)	I 1.0
TEMP	Sygnalizator zbyt wysokiej temp. ogrzewanego medium (normalnie zwarty)	I 1.1
SPALINY	Sygnalizator przekroczenia dopuszczalnego stężenia CO_2 w spalinach (normalnie zwarty)	I 1.2
POTW	Kasowanie alarmu (normalnie otwarty)	I 1.7
ALARM	Sygnalizacja alarmu	Q 5.0

Proponowane rozwiązanie znajduje się na końcu podręcznika.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

VI. Operacje grupowania

W programach pisanych do tej pory nie było konieczności użycia operacji grupowania, czyli nawiasów. Jeżeli zaistniałaby konieczność zapisania następującej logiki w STL:



można by zapisać ją w następujący sposób:

```
A      I 1.0
A(
O      I 2.0
O      I 2.1
)
=      Q 0.0
```

Czyli w nawiasie został wypracowany wynik sumy dwóch sygnałów przemnożony logicznie przez sygnał I 1.0.

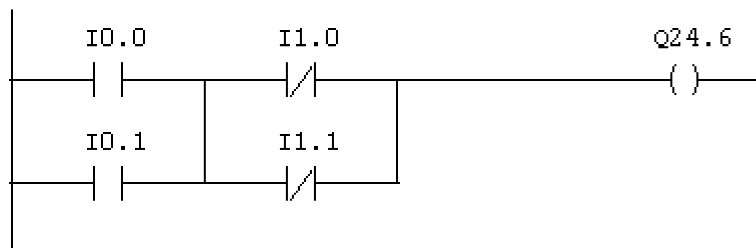
W tym konkretnym przykładzie – odwracając kolejność składników iloczynu – można by zrezygnować z użycia nawiasów:

```
O      I 2.0
O      I 2.1
A      I 1.0
=      Q 0.0
```

ale nie zawsze będzie to możliwe.

Przykład: Operacje grupowania w STL

Proszę przekształcić następującą logikę zapisaną w LAD na STL:



Zaczynając od pierwszej połączenie równolegle otrzymamy:

```
O     I 0.0
O     I 0.1
```

drugie połączenie równolegle odpowiada:

```
ON     I 1.0
ON     I 1.1
```

Oba elementy są połaczone szeregowo, czyli

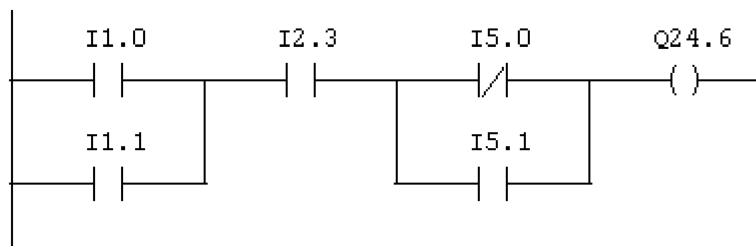
```
O     I 0.0
O     I 0.1
A(
ON     I 1.0
ON     I 1.1
)
```

wynik tych połączeń determinuje stan wyjścia:

```
 =     Q 24.6
```

Zadanie 3: Operacje grupowania w STL

Proszę przekształcić następującą logikę zapisaną w LAD na STL:



Proponowane rozwiązanie znajduje się na końcu podręcznika.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

VII. Ustawianie i kasowanie w zapisie STL – przerzutniki S, R

Zapamiętanie stanu sygnału można zrealizować przy pomocy układu podtrzymującego, tak jak to zostało przedstawione wcześniej, albo przy pomocy przerzutników.
STEP7 daje użytkownikowi do dyspozycji przerzutniki S i R.
Poniższy przykład pokazuje sposób ich użycia.

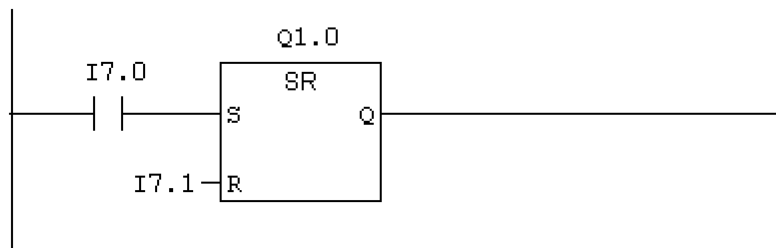
Przykład: Załączanie i wyłączanie oświetlenia na hali

Hala produkcyjna oświetlana jest szeregiem lamp jarzeniowych. Istnieje możliwość załączania i wyłączania ich przy pomocy dwóch niestabilnych przycisków ZAŁĄCZ i WYŁĄCZ. Wyjście sterujące oświetleniem oznaczone jest jako ŚWIATŁO.

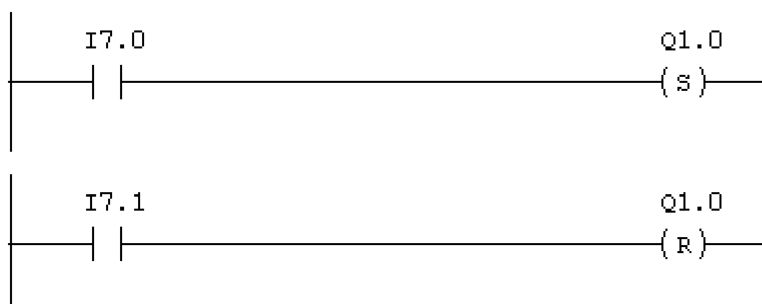
Przyporządkowanie adresów wejścia/wyjścia do poszczególnych sygnałów:

Element	Funkcja	Adres
ZAŁĄCZ	Niestabilny przycisk załączający światło (normalnie otwarty)	I 7.0
WYŁĄCZ	Niestabilny przycisk wyłączający światło (normalnie otwarty)	I 7.1
ŚWIATŁO	Wyjście na stycznik sterujący oświetleniem	Q 1.0

Pisząc ten program w LAD można by go zapisać w jednym segmencie (networku):



lub w dwóch oddzielnych segmentach przy pomocy cewek (S) i (R):



INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Analogiczną postać będzie miał program w zapisie STL, to jest najpierw nastąpi sprawdzenie odpytywanego sygnału, a kolejna operacja to **S**et lub **R**eset:

```
A      I 7.0      // sprawdzenie stanu sygnału
S      Q 1.0      // warunkowe ustawienie (SET) wyjścia

A      I 7.1      // sprawdzenie stanu sygnału
R      Q 1.0      // warunkowe kasowanie (RESET) wyjścia
```

W STL możemy zapisać ten kod w dwóch oddzielnych segmentach lub też w jednym. Operacje S oraz R są realizowane warunkowo tzn. tylko i wyłacznie w sytuacji, kiedy RLO = 'true'.

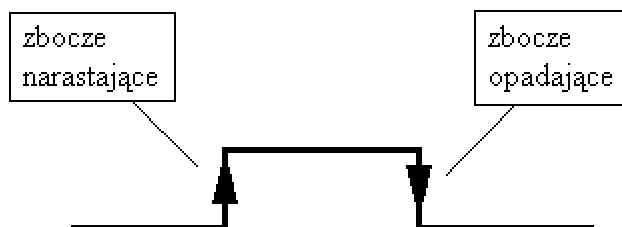
Które działanie jest dominujące ?

Tak samo, jak w LAD, tak i w STL dominujący jest **ostatni** rozkaz.

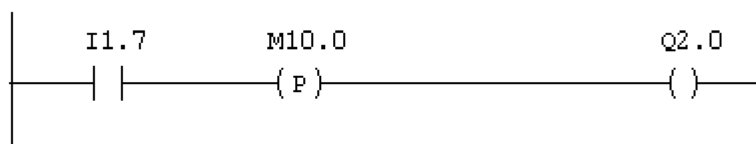
VIII. Wykrywanie zbocza – FP, FN

W praktyce przemysłowej np. uruchamiając cykl maszyny przy pomocy przycisku niestabilnego np. START nie powinien być sprawdzany **stan** wejścia (niski lub wysoki), ale **zmiana stanu** (zbocze).

Dzięki temu łatwiej jest się zabezpieczyć przed np. celowym mechanicznym zablokowaniem przycisku.



W opisie sposobu wykorzystania tej funkcji w STL znowu zostanie przywołana analogia do LAD:



INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Powyższy segment rozpoczyna się od sprawdzenia stanu testowanego sygnału, czyli I 1.7

Dalej znajduje się rozkaz wykrywający narastające zbocze (cewka P), nad tym rozkazem umieszczony jest adres pomocniczej komórki pamięci, którą cewka P wykorzystuje do przechowywania stanu wejścia I 1.7 z poprzedniego obiegu pętli programowej.

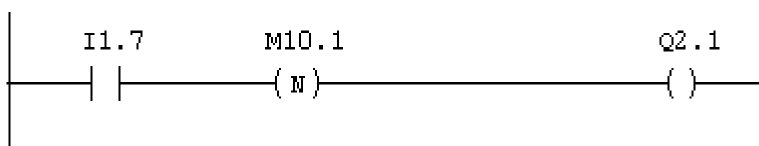
Ten sam kod w STL będzie miał postać:

Jak długo będzie wystawiany sygnał narastającego zbocza na wyjściu ?

Sygnał zbocza, obojętnie czy narastającego, czy opadającego trwa **jeden cykl pracy procesora** (typowo czas rzędu kilku - kilkunastu milisekund).

```
A       I 1.7
FP      M 10.0
=       Q 2.0
```

Podobnie, wykrycie opadającego zbocza w LAD można przedstawić jak poniżej:



natomiast w STL:

```
A       I 1.7
FN      M 10.1
=       Q 2.1
```

IX. Ustawienie i kasowanie RLO – funkcje SET i CLR

Rozkazy SET i CLR pozwalają na bezwarunkowe ustawienie RLO w stan wysoki lub niski. Mogą być wykorzystywane np. do inicjowania wartości na początku funkcji.

Przykład: Ustawienie bitu

W programie, który aktualnie jest pisany przez programistę zaszła konieczność na początku programu bezwarunkowego ustawienia bitu M 20.4 w stan wysoki. Jest to bit odpowiedzialny za określenie statusu wykonania funkcji.

W jaki sposób ustawić bezwarunkowo bit M 20.4 w stan wysoki ?

Rozkazem ustawiającym (setującym) bit jest S.

Czy wystarczy napisać

```
S     M 20.4
```

... żeby mieć pewność, że M20.4 zostanie ustawiony ?

Zdecydowane nie, realizacja operacji ustawienia uzależniona jest od wcześniej wypracowanego stanu RLO.

Jeżeli wynik jest pozytywny rozkaz ustawiania zostanie wykonany, w przeciwnym przypadku nie (S to operacja wykonywana warunkowo).

Jak więc zagwarantować ustawienie RLO ?

Rozwiązaniem jest operacja **SET**, która **bezwarunkowo** ustawia RLO.

Można więc napisać:

```
SET              // ustawienie RLO
S     M 20.4     // warunkowe ustawienie bitu
```

aby mieć pewność, że komórka M 20.4 będzie w stanie wysokim.

Analogiczny rozkaz wpisujący do RLO stan niski to **CLR**.
Jak więc można skasować bezwarunkowo M 20.5 ?

```
CLR
R     M 20.5
```

Czy taki program zagwarantuje skasowanie M 20.5 ? Zdecydowane **nie** !
Aby wykonać warunkowe kasowanie, RLO musi być **ustawione**:

```
SET              // ustawienie RLO
R     M 20.5     // warunkowe skasowanie bitu
```

Zadanie 4: Skasowanie stanu bitu

Jak przy pomocy rozkazu **CLR** skasować wyjście Q 15.0 ?

Proponowane rozwiązanie znajduje się na końcu podręcznika.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

X. Negacja bieżącego stanu RLO – funkcja NOT

Programista ma za zadanie napisanie programu sterującego sygnalizacją świetlną dla operatorów wózków widłowych umieszczoną w bramie wjazdowej do hali magazynowej. Sygnalizator zawiera dwa światła: zielone i czerwone.
Logika sterująca światłami jest poza obszarem bieżących rozważań.

Należy zapewnić, że w przypadku uruchomienia zielonego sygnalizatora – czerwony pozostanie wyłączony i odwrotnie.

Jeżeli wypracowana została już logika dla światła zielonego, np.:

```
A     I 0.1       // ...szereg warunków
AN    M 20.4      //    dla światła
AN    T 2         //    zielonego

=     Q 8.2       // światło zielone
```

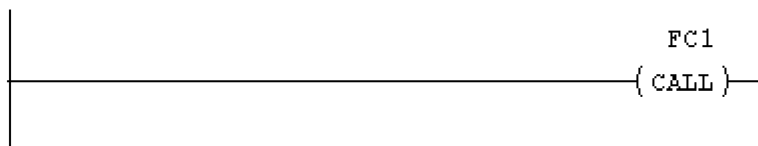
to światło czerwone powinno się świecić w **przeciwnym przypadku**.
Wystarczy więc tylko odwrócić tę logikę dopisując:

```
NOT               // negacja bieżącego stanu RLO
=     Q 8.3       // światło czerwone
```

XI. Wywoływanie bloków programowych – UC, CC, CALL

Bezwarunkowe wywołanie funkcji bezparametrowej

Chcąc bezwarunkowo wywołać np. funkcję FC1 można w języku LAD wykorzystać operację CALL:



Bezwarunkowe wywołanie funkcji w LAD przy pomocy operacji CALL

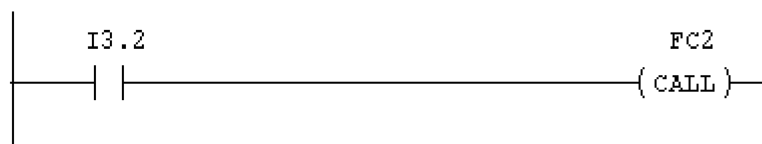
Analogiczna operacja w STL będzie miała postać:

```
UC    FC 1     // Unconditional Call – wywołanie bezwarunkowe
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Warunkowe wywołanie funkcji bezparametrowej

W przypadku wywołania warunkowego np. funkcji FC2, w LAD należałoby użyć tej samej opercji poprzedzonej warunkiem:



Warunkowe wywołanie funkcji w LAD przy pomocy operacji CALL

W STL w tym samym miejscu pojawi się już inna operacja:

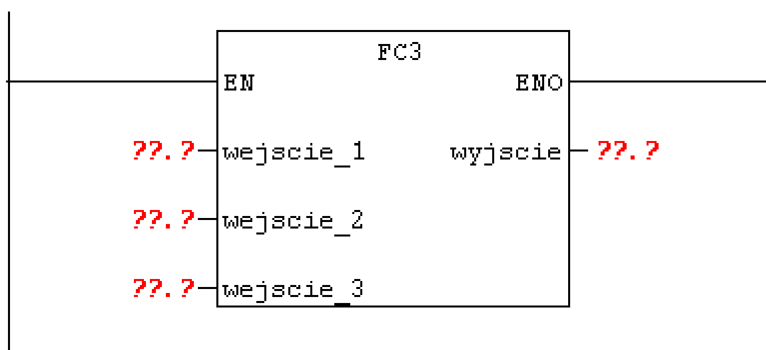
```
A      I   3.2     // Warunek wywołania funkcji
CC     FC  2       // Conditional Call – wywołanie warunkowe
```

W powyższym programie funkcja FC 2 zostanie wykonana pod warunkiem, że wejście I 3.2 będzie w stanie wysokim.

Bezwarunkowe wywołanie funkcji z parametrami

Istotną cechą operacji UC i CC jest fakt, że mogą wywoływać wyłącznie bloki **bezparametrowe**.

Jeżeli funkcja ma już zadeklarowane wejścia lub wyjścia należy wywołać ją w inny sposób. Przykładowe bezwarunkowe wywołanie funkcji FC3 z parametrami w LAD:



Bezwarunkowe wywołanie funkcji z parametrami w LAD

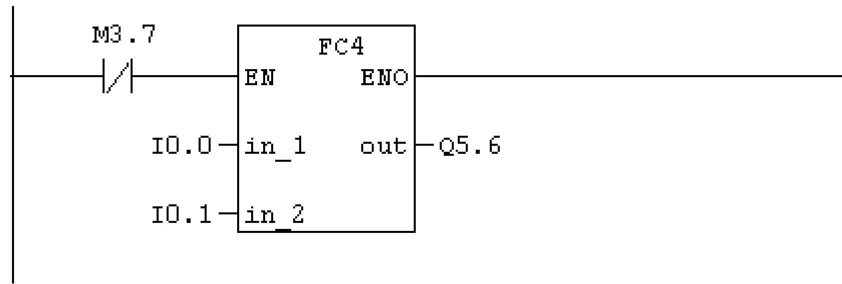
Analogiczne wywołanie w STL:

```
CALL   FC      3
  wejscie_1:=
  wejscie_2:=
  wejscie_3:=
  wyjscie   :=
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Warunkowe wywołanie funkcji z parametrami

Jeżeli funkcja z parametrami ma zostać wywołana warunkowo w LAD wystarczy dodać warunek:



Warunkowe wywołanie funkcji z parametrami w LAD

Jak teraz stworzyć analogiczne wywołanie w STL ? Czy wystarczy napisać:

```
AN    M 3.7

CALL  FC 4
   in_1:= I0.0
   in_2:= I0.1
   out := Q5.6
```

> **Pamiętaj !**
>
> Rozkazy **UC** i **CC** mogą być użyte tylko i wyłącznie w stosunku do bloków programowych, które nie posiadają parametrów. Dla bloków z parametrami należy użyć rozkazu **CALL.**

Jeżeli M 3.7 będzie w stanie niskim funkcja FC 4 będzie wykonana.
Jeżeli M 3.7 będzie w stanie wysokim funkcja FC 4 ... również będzie wykonana !

Wynika to z faktu, że operacja CALL wykonywana jest **bezwarunkowo,** niezależnie od stanu RLO w momencie wywołania.

Jak więc poprawnie wywołać warunkowo funkcję z parametrami w STL ?
Tematyka ta będzie poruszana w dalszej części tego podręcznika, ale wyprzedzając nieco można już odpowiedzieć, że należy posłużyć się **operacją skoku**:

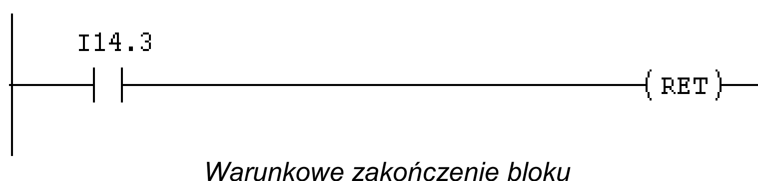
```
      AN    M 3.7       // Sprawdź warunek wykonania funkcji
      JCN   et1         // Jeżeli warunek nie jest spełniony - skocz
                        //    do etykiety et1
      CALL  FC 4        // Wywołanie funkcji
        in_1:= I0.0
        in_2:= I0.1
        out := Q5.6

et1:  NOP   0           // Etykieta et1 i dalsza część programu
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

XII. Zakończenie bloku – BEC, BEU

Warunkowe zakończenie bloku

W celu sterowania wykonywaniem zadań zapisanych w bieżącym bloku programowym programista może posłużyć się rozkazami zakończenia realizacji bloku.
W LAD warunkowe zakończenie bloku ma następującą postać:



Warunkowe zakończenie bloku

W STL taka logika powinna zostać zapisana w następujący sposób:

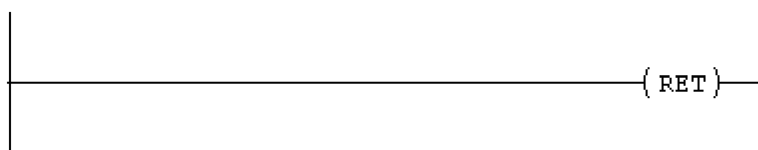
```
A       I 14.3
BEC                 // Block End Conditional -
                    // warunkowe zakończenie bloku
```

Jeżeli wejście I14.3 będzie w stanie wysokim zakończona zostanie realizacja kolejnych operacji zapisanych w bieżącym bloku programowym.

W przypadku, gdy wejście będzie w stanie niskim (a dokładnie, gdy RLO = 0) rozkaz BEC zostanie pominięty i realizowana będzie dalsza część operacji zapisanych w bloku.

Bezwarunkowe zakończenie bloku

Istnieje też możliwość bezwarunkowego zakończenia bloku. Funkcję realizującą to zadanie mógłby opisywać poniższy program:



Taką postać mogłoby mieć bezwarunkowe zakończenie bloku w LAD

Jednak edytor LAD nie pozwala na zapisanie takiego programu.

Logikę taką można jednak zapisać w STL:

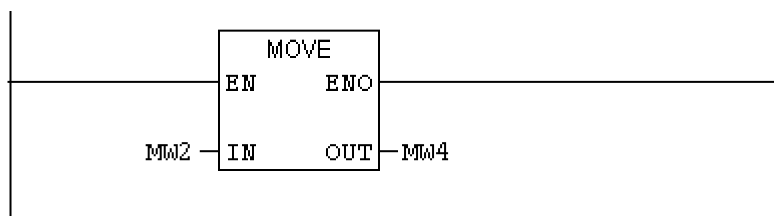
```
BEU                 // Block End Unconditional -
                    // bezwarunkowe zakończenie bloku
```

Wystąpienie takiej operacji w programie powoduje zakończenie realizacji bieżącego bloku programowego niezależnie od stanu RLO (bezwarunkowo).

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

XIII. Przenoszenie danych – L, T

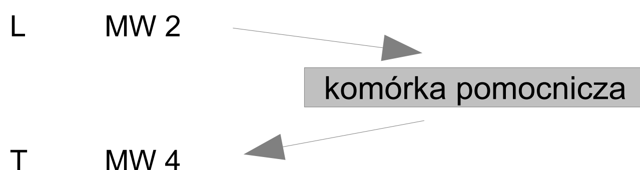
W LAD aby przepisać zawartość MW2 na MW4, należałoby użyć funkcji MOVE:



Aby wykonać tę operację w STL należałoby zapisać:

```
L     MW 2 // Załaduj (Load) zawartość MW2 do komórki pomocniczej
T     MW 4 // Wyślij (Transfer) zawartość komórki pomocniczej
           // do MW4
```

Ale jeżeli „załaduj”, to... dokąd ? I jeżeli „wyślij” to skąd pobrać wartość ? Gdzie po pierwszym rozkazie jest przechowywana zawartość komórki MW2 ?



Komórką pomocniczą, wykorzystywaną przy operacjach Load i Transfer jest **Akumulator**, a dokładnie Akumulator 1.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Przykład: Testowanie wyjść

Programista w czasie rozruchu maszyny chce sprawdzić poprawność adresowania i konfiguracji sprzętowej modułu wyjść cyfrowych. W tym celu chce wystawić stan wysoki na wszystkie wyjścia. Jak to zrobić ?

Zadeklarowane wyjścia to słowo zaczynające się od bajtu o adresie 8.

Jeżeli mają być wystawione jedynki na każde z wyjść, to znaczy, że należy wysłać na słowo, w którym wszytkie bity są ustawione co można zapisać w kodzie binarnym jako 2#1111111111111111 lub wygodniej w zapisie szesnastkowym W#16#FFFF, czyli:

```
L     W#16#FFFF       // 16 jedynek w zapisie szesnastkowym
T     QW 8            // Słowo wyjściowe o adresie 8
```

Zadanie 5: Przepisanie wejść na wyjścia

Jak zapisać program w STL, który przepisze bajt wejściowy o adresie 4 na bajt wyjściowy o adresie 13 ?

Proponowane rozwiązanie znajduje się na końcu podręcznika.

Warunkowe przepisanie

Rozkazy ładowania L i transferu T wykonywane są **bezwarunkowo**, a więc niezależnie od wcześniej wypracowanego wyniku logicznego, czyli stanu RLO.
To znaczy wykonywane są w każdym obiegu pętli programowej, można je jedynie ominąć przy pomocy skoków. Nawet gdyby został zapisany następujący kod:

```
A     I 0.7

L     0
T     MW12
```

to niezależnie, czy wejście I 0.7 będzie w stanie wysokim czy niskim, operacje L i T **zostaną wykonane**. Aby faktycznie uzależnić wykonanie tych operacji od stanu wejścia I 0.7 należy użyć operację skoku:

```
      A     I 0.7      // Sprawdzenie, czy wejście I 0.7 jest ustawione
      JCN   et3        // Jeżeli warunek jest niespełniony,
                       // a więc RLO = 0 wykonaj skok do etykiety et3

      L     0          // Załaduj wartość 0 do akumulatora
      T     MW 12      // Wyślij zawartość akumulatora do MW12

et3: NOP 0             // Etykieta et3 oraz rozkaz "No Operation"
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

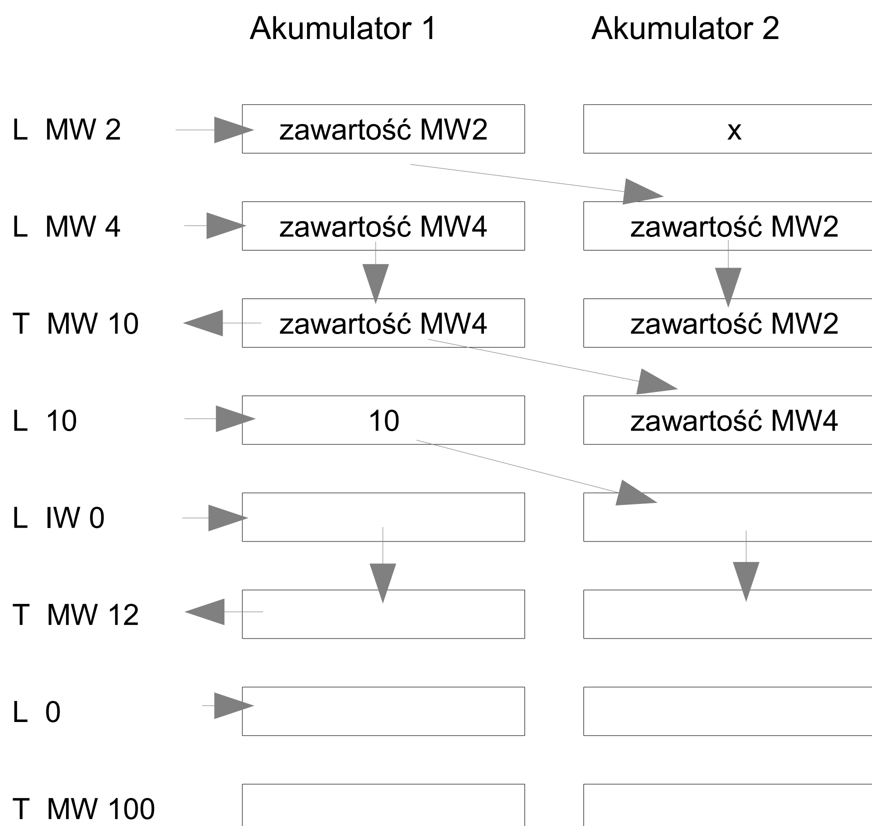
Wpływ operacji ładowania i transferu na zawartości akumulatorów

Operacja ładowania L zawsze ładuje zawartość parametru lub stałą do akumulatora 1.
Jednak wcześniej, poprzednia zawartość akumulatora 1 jest kopiowana do akumulatora 2.
Natomiast poprzednia zawartość akumulatora 2 jest tracona.

Operacja transferu T kopiuje zawartość akumulatora 1 do komórki wskazywanej przez parametr operacji. Operacja transferu nie modyfikuje zawartości akumulatorów.

Zadanie 6: Wpływ operacji ładowania i transferu na zawartości akumulatorów

Modyfikację akumulatorów najlepiej prześledzić na podstawie poniższego przykładu:



Proszę uzupełnić puste pola w powyższym diagramie.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

XIV. Zliczanie zdarzeń - liczniki

Dzięki licznikom można w znaczy sposób uprościć realizację zadań związanych z monitorowaniem ilości wystąpień określonego zdarzenia. Alternatywą dla liczników mogą być operacje arytmetyczne.

Przykład: Monitorowanie zawartośći bufora

Zrealizować system nadzorujący ilość elementów w buforze.

Założenia:

- informacja o nowym elemencie dostarczana jest w postaci impulsu przez fotokomórkę WEJŚCIE znajdującą się przy wejściu do bufora
- wydanie elementu jest sygnalizowane impulsem generowanym przez fotokomórkę WYJŚCIE zainstalowaną przy wyjściu z bufora
- maksymalna liczba elementów w buforze to 999
- bieżąca liczba elementów w buforze powinna zostać wyświetlona na WYŚWIETLACZu (w kodzie BCD)

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych sygnałów:

Element	Funkcja	Adres
WEJSCIE	Fotokomórka sygnalizująca wprowadzenie elementu do bufora	I 17.0
WYJSCIE	Fotokomórka sygnalizująca wydanie elementu z bufora	I 17.1
WYSWIETLACZ	Wyświetlacz informujący o aktualnej liczbie elementów w buforze	QW 24

Elementem pamiętającym ilość elementów w buforze jest licznik. Wprowadzenie nowego elementu do bufora oznacza inkrementację (zwiększenie o 1) stanu licznika, natomiast wydanie elementu będzie dla programu oznaczać dekrementację (zmniejszenie o 1) stanu licznika.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl

NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Operacje związane z działaniami na licznikach w STL zebrane zostały w poniższej tabeli:

Mnemonik	**Instrukcja**
CU C_nr	warunkowa inkrementacja licznika
CD C_nr	warunkowa dekrementacja licznika
S C_nr	warunkowe ustawienie stanu licznika do wartości określonej przez zawartość akumulatora 1
R C_nr	warunkowe zerowanie licznika
L C_nr	załadowanie do akumulatora 1 wartości licznika (binarnie)
LC C_nr	załadowanie do akumulatora 1 wartości licznika (w kodzie BCD)

Realizując powyższe zadanie można rozpocząć od zdarzenia opisującego wprowadzenie elementu do bufora, może ono zostać zapisane jako warunkowa inkrementacja licznika o 1:

```
A     I 17.0     // warunek inkrementacji - WEJSCIE
CU    C2         // warunkowa inkrementacja licznika C2
```

Pierwsza linia kodu sprawdza warunek zwiększenia stanu licznika (w tym wypadku wybrano licznik C2), jeżeli na wejściu **I** 17.0 pojawi się narastające zbocze, wtedy stan licznika zostanie zwiększony o 1 (uwaga: samo wykrywanie zbocza jest realizowane przez licznik).

Analogicznie można uzupełnić powyższy program o dekrementację w przypadku, gdy element jest wydawany z bufora:

```
A     I 17.1     // warunek dekrementacji - WYJSCIE
CD    C2         // warunkowa dekrementacja licznika C2
```

Jest to kod wystarczający do poprawnego zliczania ilości elementów w buforze. Ostatnim warunkiem zdefiniowanym w treści zadania jest wysłanie bieżącej ilości elementów do słowa wyjściowego QW 24 w postaci kodu BCD:

```
LC    C2         // pobranie stanu licznika w postaci kodu BCD
T     QW 24      // wysłanie stanu licznika na wyświetlacz
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

XV. Operacje porównywania – komparatory

Operacje porównywania pozwalają na określenie zależności pomiędzy dwoma argumentami, np. sprawdzenie większości, czy też równości.

Zadanie 7: Monitorowanie ilości elementów w buforze – ciąg dalszy

Jest to kontynuacja poprzedniego zadania. Należy wprowadzić sygnalizację z wykorzystaniem trzech lampek:

- lampka L0 powinna się świecić, gdy bufor jest pusty
- lampka L1 oznacza liczbę elementów w przedziale od 1-900
- lampka L2 świeci się, gdy liczba elementów zawiera się w przedziale 901-999.

Przyporządkowanie pozostałych adresów wyjść:

Element	Funkcja	Adres
L0	Liczba elementów = 0	Q 20.0
L1	Liczba elementów w przedziale 1-900	Q 20.1
L2	Liczba elementów w przedziale 901-999	Q 20.2

Operacje porównywania wykonywane są bezwarunkowo, to znaczy niezależnie od stanu RLO (wyniku poprzednich operacji logicznych) i zawsze sprawdzana jest zależność pomiędzy zawartością akmulatora 2 w stosunku do akumulatora 1. Przykładowo operacja **>I** sprawdza, czy zawartość akumulatora 2 jest większa od zawartości akumulatora 1 (w obydwu akumulatorach powinny być zapisane liczby INT). W zależności od wyniku sprawdzenia modyfikowany jest stan RLO.
Należy również pamiętać, aby porównywane argumenty były tego samego typu (np. INT z INT, a Real z Real). Funkcje porównywania realizowane przez sterowniki SIMATIC S7 zebrane zostały w poniższej tabeli.

Funkcja	Typ danych		
	Integer	**D**ouble Integer	**R**eal
sprawdzanie równości	**==I**	**==D**	**==R**
sprawdzanie różności	**<>I**	**<>D**	**<>R**
sprawdzane większości	**>I**	**>D**	**>R**
sprawdzane mniejszości	**<I**	**<D**	**<R**
sprawdzane większości lub równości	**>=I**	**>=D**	**>=R**
sprawdzane mniejszości lub równości	**<=I**	**<=D**	**<=R**

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Przykład sprawdzenia, czy warunek MW2 > 100 jest prawdziwy:

```
L     MW 2       // Załadowanie pierwszej sprawdzanej wartości
L     100        // Załadowanie drugiej sprawdzanej wartości
>I               // Operacja porównania (czy ACCU2 > ACCU1 ?)
=     M 87.0     // Jeżeli wynik jest prawdziwy M 87.0
                 //    zostanie ustwione
```

Taki schemat będzie się powtarzać dla każdej z operacji porównywania, czyli najpierw należy załadować pierwszy argument, później należy załadować drugi argument, w trzecim rozkazie wykonujemy operację porównywania.

Wracając do zadania z tego rozdziału, należy napisać kod analogiczny do powyższego schematu, czyli:

```
L     C 2        // Załadowanie stanu licznika –
                 //    - pierwszy argument
L     900        // Załadowanie drugiego argumentu
<=I              // Sprawdzenie, czy C2 <= 900
=     Q 20.1     // Jeżeli powyższe sprawdzenie jest prawdziwe
                 // to zostanie wysterowana lampka L1 (Q20.1)
```

Należy uzupełnić pozostałą część programu (odpowiedzialną za wysterowanie L0 i L2).

Proponowane rozwiązanie znajduje się na końcu podręcznika.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

XVI. Funkcje arytmetyczne

Dzięki tym operacjom możliwe jest wykonanie w programie podstawowych operacji matematycznych, to jest dodawania, odejmowania, mnożenia, dzielenia.

Funkcja	Typ danych		
	Integer	Double Integer	Real
dodawanie	**+I**	**+D**	**+R**
odejmowanie	**-I**	**-D**	**-R**
mnożenie	***I**	***D**	***R**
dzielenie	**/I**	**/D**	**/R**
wyliczenie reszty z dzielenia		**MOD**	

Przykład wykorzystania operacji arytmetycznych:

	ACCU1	ACCU2
`L MW 20`	(MW20)	x
`L 100`	100	(MW20)
`/I`	(MW20) / 100	(MW20)
`T MW 12`	(MW20) / 100	(MW20)
`L MW 22`	MW22	(MW20) / 100
`+I`	(MW20) / 100 + (MW22)	(MW20) / 100
`T MW 42`	(MW20) / 100 + (MW22)	(MW20) / 100

Opis zawartości akumulatorów jest oczywiście symboliczny, w rzeczywistości zobaczymy np. w ACCU1 konkretną wartość, np. „34”.

W konsekwencji powyższy program zapisze w MW 12 wartość ilorazu:

MW12 = MW20 / 100

natomiast w MW42 wynik:

MW42 = MW20 / 100 + MW22

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Zadanie 8: Operacje arytmetyczne

Należy uzupełnić poniższą tabelkę (zawartość akumulatorów na poziomie każdego z rozkazów), analogicznie jak we wcześniejszym przykładzie.

	ACCU1	ACCU2
`L IW 4`		
`L 2`		
`+I`		
`L 50`		
`/I`		
`T MW 120`		

Funkcja realizowana przez ten program to:

MW 120 =

Rozwiązanie znajduje się na końcu podręcznika.

XVII. Realizacja opóźnienia – układy czasowe

Układy czasowe pozwalają na realizację w programie zależności czasowych.
W sterownikach SIMATIC S7 dostępnych jest 5 rodzajów układów czasowych (zależności czasowych jakie można wygenerować). Operacje wywołujące poszczególne funkcje zebrane są w poniższej tabeli.

Mnemonik	Działanie
SP T_nr	warunkowe wyzwolenie timera w trybie *Impuls* (S_PULSE)
SE T_nr	warunkowe wyzwolenie timera w trybie *Impuls z pamięcią* (S_PEXT)
SD T_nr	warunkowe wyzwolenie timera w trybie *Opóźnienie załączenia* (S_ODT)
SS T_nr	warunkowe wyzwolenie timera w trybie *Opóźnienie załączenia z pamięcią* (S_ODTS)
SF T_nr	warunkowe wyzwolenie timera w trybie *Opóźnienie wyłączenia* (S_OFFDT)
R T_nr	warunkowe zerowanie timera

T_nr oznacza numer układu czasowego np. T2.

Różnica pomiędzy poszczególnymi funkcjami została omówiona na kursie PODSTAWOWY S7.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Przykład: Sterowanie prasą z opóźnieniem

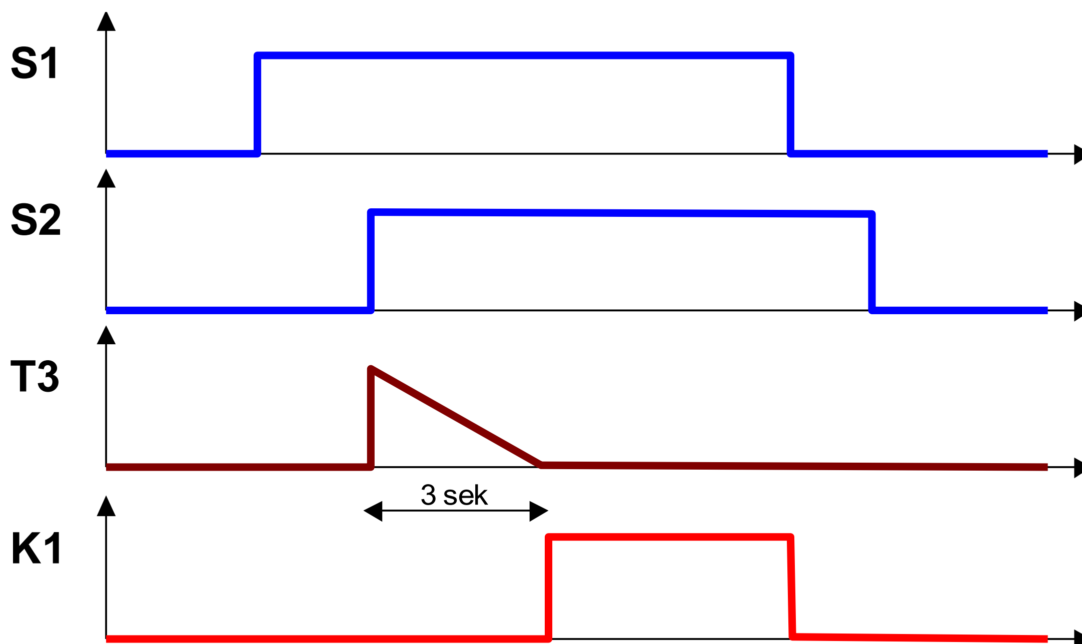
Aby uchronić operatora prasy przed skaleczeniem rąk zastosowano specjalny układ wyzwalający. Wyzwolenie prasy może nastąpić tylko i wyłącznie w momencie równoczesnego naciśnięcia dwóch przycisków S1 i S2 rozmieszczonych tak, aby nie było możliwe naciśnięcie obydwu przycisków jedną ręką.
Prasa wyzwalana jest po upływie 3 sek. od momentu naciśnięcia obydwu przycisków.

Po wyzwoleniu prasy sterowanej silnikiem poprzez stycznik K1 wykonywany jest skok roboczy. Po zwolnieniu jednego z przycisków prasa wraca samoczynnie do położenia początkowego.

Przyporządkowanie adresów wejścia/wyjścia do poszczególnych elementów:

Element	Funkcja	Adres
S1	Przycisk dla lewej ręki	I 0.3
S2	Przycisk dla prawej ręki	I 0.4
K1	Stycznik silnika sterującego prasą	Q 4.3

Zadanie to jest rozwinięciem pierwszego problemu w tym podręczniku. Należy zrealizować układ z opóźnieniem czasowym o charakterystyce, którą można przedstawić na następujących przebiegach czasowych:



INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Układem czasowym, który w najprostszy sposób pozwoli zrealizować zależność czasową przedstawioną na rysunku jest timer pracujący w trybie SD (S_ODT). Przykładowe wyzwolenie układu T3 w tym trybie będzie miało następującą postać:

```
SD    T3
```

Operacja ta uruchomi timer typu *Opóźnienie załączenia*, ale pod jakim warunkiem, na jaki czas? Przed wyzwoleniem układu czasowego należy zadeklarować **warunek wywołania timera**:

```
A     I 0.3
A     I 0.4
SD    T3
```

... dzięki temu timer rozpocznie odmierzanie czasu po jednoczesnym naciśnięciu obydwu przycisków. Zanim jednak układ czasowy zostanie wyzwolony należy zapewnić, że w momencie wyzwolenia układu czasowego, odpowiednia **wartość czasu** w formacie S5Time będzie dostępna w akumulatorze 1. W tym zadaniu należy odczekać 3 sekundy, a więc:

```
L     S5T#3s
A     I 0.3
A     I 0.4
SD    T3
```

lub

```
A     I 0.3
A     I 0.4
L     S5T#3s
SD    T3
```

... ponieważ operacje A wpływają na RLO, zaś L na akumulator 1.

Zapisując czas jako parametr układu czasowego należy go poprzedzić identyfikatorem S5T# lub S5TIME#.

Minimalny czas jaki układ czasowy może odmierzyć to 10 ms, natomiast maksymalny to 2 godziny 46 minut i 30 sekund.

Minimalna i maksymalna wartość czasu wynika ze sposobu przechowywania czasu w komórce związanej z układem czasowym – patrz kurs Podstawowy S7.

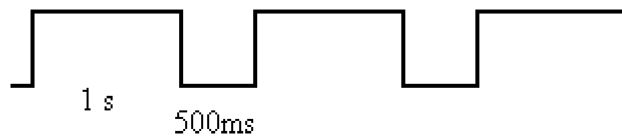
Powyższe instrukcje zapewniają prawidłowe uruchomienie układu czasowego. Należy je uzupełnić o odwołanie, które sprawdziłoby stan układu czasowego. Gdy timer SD zakończy pracę, na swoim wyjściu wystawia stan wysoki (tak dugo jak długo jeszcze będzie aktywne wejście). Kompletne rozwiązanie tego problemu może mieć więc taką postać:

```
L     S5T#3s
A     I 0.3
A     I 0.4
SD    T3

A     T3
=     Q 4.3
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Zadanie 9: Generator fali prostokątnej

Należy napisać program, który będzie realizował funkcję generatora częstotliwości o następującym przebiegu czasowym (niesymetryczne wypełnienie impulsów):



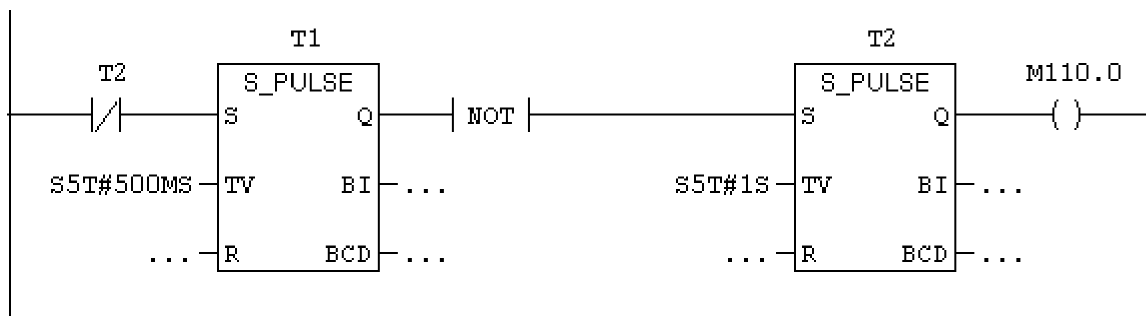
Przebieg powinien być generowany w bicie M110.0

Proponowane rozwiązanie znajduje się na końcu podręcznika.

Podpowiedzią do zadania może być sposób wyzwalania timerów (najprościej użyć 2 timery), to jest:

- pierwszy będzie pracować, gdy nie pracuje drugi,
- drugi będzie pracować gdy nie pracuje pierwszy.

Kolejną podpowiedzią będzie program realizujący to zadanie, napisany w LAD:



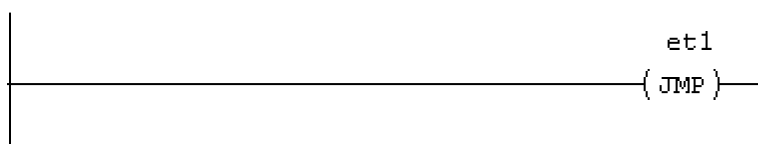
INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

XVIII. Operacje skoku – JU, JC, JCN

Dzięki operacjom skoku możliwie jest warunkowe lub bezwarunkowe omijanie pewnych fragmentów programu lub powrót do wcześniej wykonanych rozkazów. Dzięki temu programista może, w oparciu o warunki skoków, sterować realizacją programu.
Parametrem rozkazu skoku jest etykieta, wskazująca miejsce w programie, od którego powinna zostać rozpoczęta realizacja dalszych operacji. W języku STEP7 etykieta może składać się maksymalnie z 4 znaków i pierwszy nie może być liczbą (istotna jest wielkość liter).

Skok bezwarunkowy - JU

Pierwszy ze skoków, jaki zostanie omówiony to skok bezwarunkowy, który w LAD miał następującą postać:



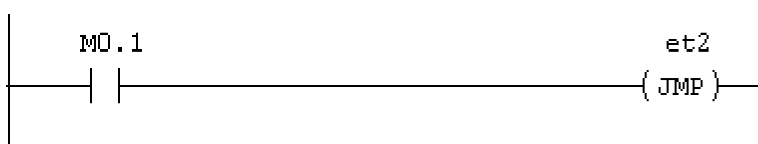
Bezwarunkowy skok do etykiety et1

W STL taki rozkaz ma następujący odpowiednik:

```
JU    et1         // Jump Unconditional - ang. skok bezwarunkowy
```

Skok warunkowy (dla spełnionego warunku) - JC

Jeżeli wykonanie skoku powinno być uzależnione od spełnienia warunku, w LAD używany jest ten sam element (JMP) poprzedzony warunkiem realizacj:



Warunkowy skok do etykiety et2 (dla M 0.1 = 1)

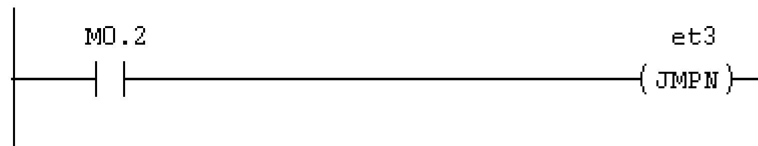
W STL rozkaz skoku warunkowego ma inny zapis (w porównaniu ze skokiem bezwarunkowym). Zapis operacji skoku warunkowego przedstawiony jest poniżej.

```
A     M 0.1
JC    et2         // Jump Conditional - ang. skok warunkowy
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Skok warunkowy (dla niespełnionego warunku) - JCN

Jeżeli konieczne jest wykonanie skoku, w sytuacji kiedy warunek nie jest spełniony, w LAD dostępna jest cewka JMPN:



Warunkowy skok do etykiety et3 (gdy warunek = 0)

w zapisie STL operacja ta zostałaby zapisana w następujący sposób:

```
 A      M 0.2
JCN    et3        // Jump Conditional if Not
```

Przykład: Struktura albo - albo

W praktyce programistycznej często spotyka się sytuację, że w zależności od stanu bitu, np. M10.0 należy wykonać jeden z dwóch fragmentów kodu programu.

W zadaniu dla M10.0 = '1' należy wykonać część A, a jeżeli M10.0 = '0' część B.

Na początek należy sprawdzić stan odpytywanego bitu:

```
A     M 10.0
```

jeżeli M10.0 jest ustawiony, wykonana ma być część A, można to zapisać opisowo:

```
A     M 10.0
// część A
```

jeżeli natomiast odpytywana flaga znajduje się w stanie niskim, należy ominąć część A i skoczyć do części B:

```
        A     M 10.0
        JCN   cz_B       // skok jeżeli RLO = „0”
        // część A
cz_B:   // część B
```

Czy program będzie działał poprawnie ? Jeżeli M 10.0 = „0”, wtedy wykonany zostanie skok do etykiety cz_B i wykonana zostanie część B programu. Natomiast, jeżeli M 10.0 będzie ustawiony, wtedy nie zostanie wykonany skok, zrealizowane zostaną kolejne instrukcje, a więc część A... i część B.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Jak więc zatrzymać wykonywanie programu po części A ?

Skokiem do kolejnej etykiety lub operacji bezwarunkowego zakończenia bloku. Czyli:

```
            A     M 10.0
            JCN   cz_B       // skok jeżeli RLO = „0”
                             // część A
            JU kon
    cz_B:                    // część B
     kon:                    // dalsza część programu
```

lub

```
            A     M 10.0
            JCN   cz_B       // skok jeżeli RLO = „0”
                             // część A
            BEU              // bezwarunkowe zakończenie bloku
    cz_B:                    // część B
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

XIX. Rozwiązania zadań

Zadanie 1: Zezwolenie na jazdę przenośnika

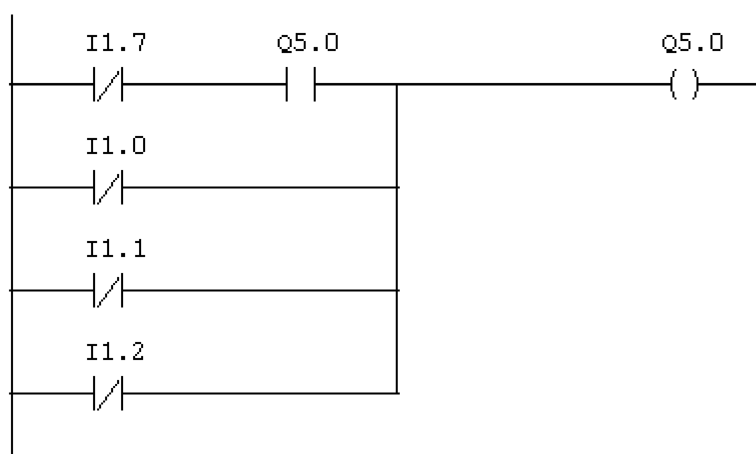
Aby sprawdzić jednoczesne występowanie trzech warunków należy wykorzystać operację iloczynu (połączenie szeregowe) trzech argumentów:

```
A     I 14.1     // ESTOP nieaktywny, na wejściu sterownika -> „1”
A     I 12.4     // I przycisk JAZDA wciśnięty,
                 //   na wejściu sterownika -> „1”
AN    I 2.7      // I kluczyk wyboru trybu pracy
                 //   w pozycji pracy ręcznej AUTO_MAN = „0”
=     Q 0.2      // Wysterowanie napędu poprzez stycznik K23
```

Sygnał I 14.1 można rozumieć jako ESTOP_OK, czyli jeżeli na wejściu sterownika pojawia się stan wysoki, oznacza to, zezwolenie na pracę (stan poprawny).

Zadanie 2: Rejestracja zdarzeń

Jest to zadanie bardziej złożone, dlatego pewną podpowiedzią może być proponowane rozwiązanie zapisane w postaci LAD:



W tym zadaniu, jak i w przykładzie „Przenośnik taśmowy” potrzebne jest użycie funkcji podtrzymywania sygnału. Jednak teraz w przypadku jednoczesnego występowania alarmu, jak i próby jego kasowania, alarm powinien **pozostać aktywny**. Mamy więc do czynienia z dominacją załączenia.

Dominacja załączenia oznacza, że jeżeli jednocześnie będzie spełniony warunek ZAŁ, jak i WYŁ, to w tym rozwiązaniu dominujące będzie załączanie.

INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Rozwiązanie w STL:

```
AN    I      1.7    // nie naciśnięty przycisk POTW
A     Q      5.0    // podtrzymanie sygnału ALARM
ON    I      1.0    // zgłoszenie alarmu od czujnika PALIWO
ON    I      1.1    // zgłoszenie alarmu od czujnika TEMP
ON    I      1.2    // zgłoszenie alarmu od czujnika SPALINY
=     Q      5.0    // wysterowanie wyjścia ALARM
```

Zadanie 3: Operacje grupowania w STL

Jedno z możliwych rozwiązań ma następującą postać:

```
O     I      1.0
O     I      1.1

A     I      2.3

A(
ON    I      5.0
O     I      5.1
)

=     Q     24.6
```

Zadanie 4: Skasowanie stanu bitu

Skasowanie bitu przy pomocy rozkazu CLR:

```
CLR              // Ustawienie RLO = „0”
=     Q 15.0     // Wysłanie „0” na wyjście
```

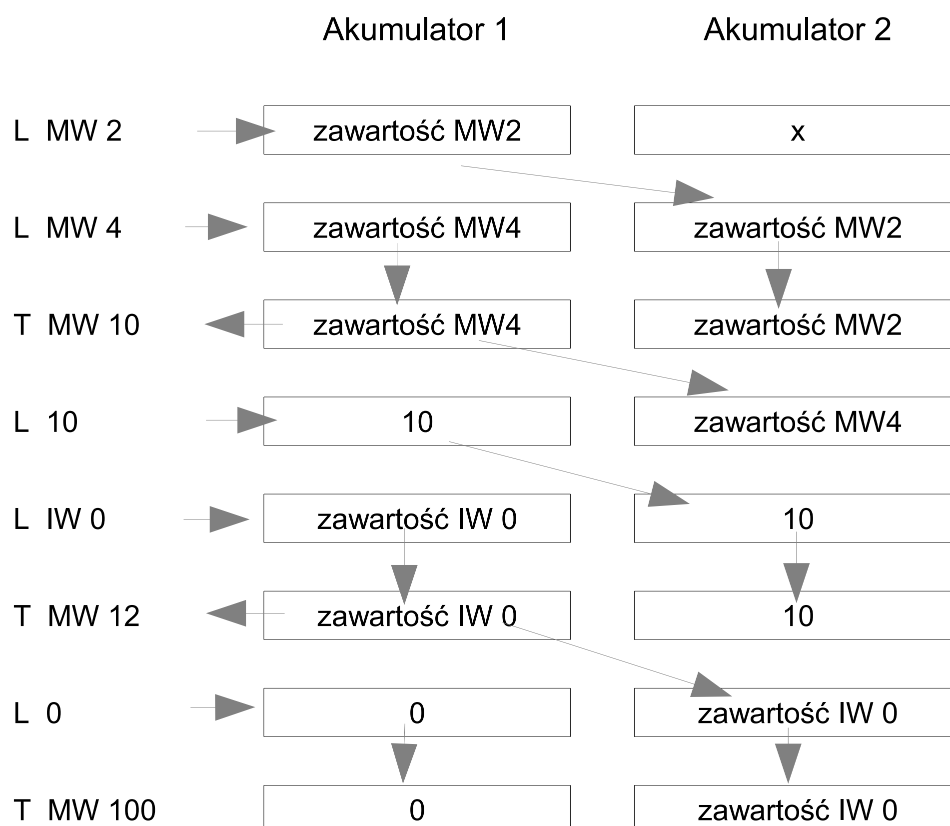
Zadanie 5: Przepisanie wejść na wyjścia

Przepisanie bajtu wejściowego o adresie 4 na bajt wyjściowy o adresie 13:

```
L     IB 4
T     QB 13
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Zadanie 6: Wpływ operacji ładowania i transferu na zawartości akumulatorów



INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Zadanie 7: Monitorowanie ilości elementów w buforze – ciąg dalszy

Podobnie jak pierwszy zakres, można sprawdzić zakres 999:

```
L     C 2         // Załadowanie pierwszego argumentu
L     999         // Załadowanie drugiego argumentu
<= I              // Sprawdzenie, czy C2 <= 999
```

Jednak to jeszcze nie jest warunek wystarczający do zaświecenia lampki C2, należy jeszcze sprawdzić, czy nie świeci się już L1:

```
AN    Q 20.1      // oraz C2 >= 900
=     Q 20.2      // Jeżeli prawdziwe, to zaświeć L2
```

Zostało jeszcze sprawdzenie, czy bufor jest pusty, co w kontekście licznika oznacza sprawdzenie, czy stan licznika wynosi 0. Można wprowadzić tutaj kolejny komparator (i porównać stan licznika z 0), ale prościej jest wykorzystać wyjście statusu układu licznikowego:

```
AN    C 2         // Jeżeli licznik zwraca status 0, oznacza to,
                  //   że stan licznika również wynosi 0
=     Q 20.0      // L0
```

W związku z tym końcowa postać całego programu w bieżącym zadaniu będzie następująca:

```
// inkrementacja
      A     I 17.0      // warunek inkrementacji - WEJSCIE
      CU    C2          // warunkowa inkrementacja licznika C2

// dekrementacja
      A     I 17.1      // warunek dekrementacji - WYJSCIE
      CD    C2          // warunkowa dekrementacja licznika C2

// stan w kodzie BCD
      LC    C2          // pobranie stanu licznika w postaci kodu BCD
      T     QW 24       // wysłanie stanu licznika na wyświetlacz

// L1 (C2<=900)
      L     C 2         // Załadowanie pierwszego argumentu
      L     900         // Załadowanie drugiego argumentu
      <= I              // Sprawdzenie, czy C2 <= 900
      =     Q 20.1      // L1

// L2 (901 <= C2 <= 999)
      L     C 2         // Załadowanie pierwszego argumentu
      L     999         // Załadowanie drugiego argumentu
      <= I              // Sprawdzenie, czy C2 <= 999
      AN    Q 20.1      // oraz C2 > 900
      =     Q 20.2      // L2

// L0 (C2=0)
      AN    C 2         // Zanegowany status licznika
      =     Q 20.0      // L0
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN

Zadanie 8: Operacje arytmetyczne

	ACCU1	ACCU2
`L IW 4`	(IW 4)	x
`L 2`	2	(IW 4)
`+I`	(IW 4) + 2	(IW 4)
`L 50`	50	(IW 4) + 2
`/I`	(IW4 + 2) / 50	(IW 4) + 2
`T MW 120`	(IW4 + 2) / 50	(IW 4) + 2

MW120 = (IW4 + 2) / 50

Zadanie 9: Generator fali prostokątnej

```
AN     T 2              // Jeżeli nie pracuje T2, niech pracuje T1
L      S5T#500MS        // Załadowanie czasu do odmierzenia do ACCU1
SP     T      1         // Wyzwolenie timera T1

AN     T 1              // Jeżeli nie pracuje T1, niech pracuje T2
L      S5T#1S           // Załadowanie czasu do odmierzenia do ACCU1
SP     T 2              // Wyzwolenie timera T1

A      T 2
=      M 110.0          // Sterowanie bitem
```


INTEX Sp. z o.o. | ul. Portowa 4 | 44-102 Gliwice | tel.: +48 32 230 75 16 | intex@intex.com.pl | www.intex.com.pl
NIP 631-000-88-84 | Zarej. pod nr KRS 0000134132 w Sądzie Rejonowym w Gliwicach, X Wydział Gospodarczy Krajowego Rejestru Sądowego | Kapitał zakładowy 200.000 PLN