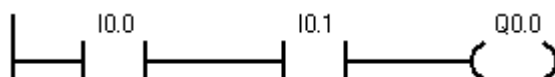


CZĘŚĆ A – UKŁADY KOMBINACYJNE

1. Wstęp

Na sterowniku PLC (Programmable Logic Controller) można realizować m.in. układy kombinacyjne. Podstawowym językiem dla układów PLC jest język drabinkowy, wykorzystujący logikę stykową. Podstawowe działania logiczne realizuje się na stykach w sposób następujący:



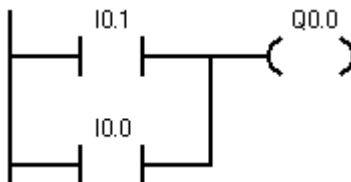
Rysunek 1. AND (I – iloczyn logiczny) na stykach: $Q0.0 = I0.0 \cdot I0.1$

Rysunek 1 najprościej wytłumaczyć tak:

- szyna po lewej to zasilanie, z niej „bierzemy” prąd,
- skoro tak, to aby wyjście Q0.0 „dostało” prąd, muszą się zewrzeć styki I0.0 i I0.1.

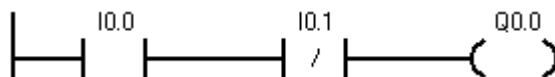
Jest to więc realizacja logicznej funkcji I, czyli AND.

Analogicznie można na stykach zrealizować funkcję LUB, czyli OR. Jeżeli styki zostały ułożone jak na Rysunku 2, to aby wyjście Q1 „dostało” prąd, wystarczy że zewrze się styk I0.0 lub I0.1:



Rysunek 2. OR (LUB – suma logiczna) na stykach: $Q0.0 = I0.0 + I0.1$

A co z negacją? Styki pokazane wyżej to tzw. „styki zwierne”. Zwierają się one, gdy reprezentowana przez nie zmienna jest równa 1, gdy jest ona równa zero, rozwierają się. Istnieje więc również „styk rozwierny”, działający odwrotnie, więc reprezentujący negację zmiennej którą symbolizuje. Spójrzmy więc na rysunek poniżej.



Rysunek 3. Styk zwierny, reprezentujący wejście wprost: I0.0, oraz styk rozwierny czyli wejście zanegowane: $\sim I0.1$

W układzie z rysunku powyżej realizowana jest funkcja $Q0.0 = I0.0 \cdot \sim I0.1$. Wyjście Q0.0 zadziała wtedy kiedy wejście I0.0 będzie zwarte, natomiast I0.1 rozwarne.

Dysponując powyższą wiedzą można zrealizować każdą funkcję logiczną, bądź to opisaną słownie (jak pokazano wyżej), bądź zadaną w postaci tabeli (niżej).

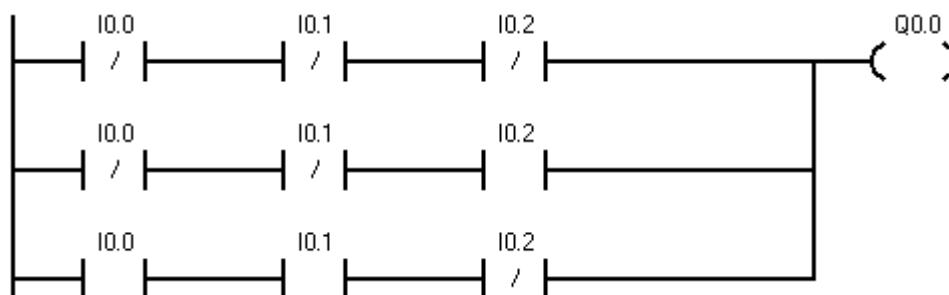
Popatrzmy więc na układ opisany następującą tabelą stanów (wejścia: A, B, C, wyjście OUT):

l.p.	A	B	C	OUT
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	x
4	1	0	0	x
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Na podstawie tej tabeli stanów uzyskujemy opis kanoniczny w zapisie Boola:

$$OUT = \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C},$$

z którego możemy bezpośrednio uzyskać realizację zadanej tą tabelą funkcji logicznej na sterowniku PLC:

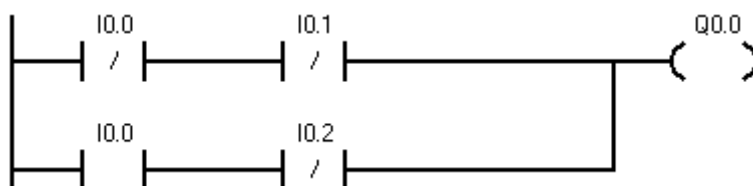


gdzie oczywiście: A=I0.0, B=I0.1, C=I0.2 oraz OUT=Q0.0.

Układ ten oczywiście można zminimalizować, metodą tablic Karnaugh (jak niżej), lub inną:

A	0	1
BC		
00	1	X
01	1	0
11	X	0
10	0	1

uzyskując opis kanoniczny: $OUT = \bar{A} \cdot \bar{B} + A \cdot \bar{C}$ i jego realizację:



gdzie podobnie jak poprzednio: A=I0.0, B=I0.1, C=I0.2 oraz OUT=Q0.0.

CZĘŚĆ B – WYKORZYSTANIE TIMERÓW

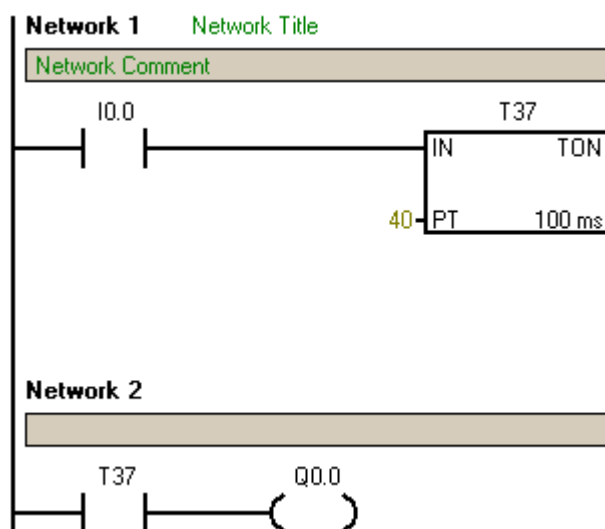
2. Wstęp

Timery w sterowniku PLC służą do odliczania w czasie. W trakcie laboratorium zapoznamy się z dwoma rodzajami timerów dostępnych w używanym sterowniku (**Simatic S7-200** produkcji firmy **Siemens**):

- On-Delay Timer (TON) jest timerem zliczającym czas gdy jego wejście (IN) jest w stanie wysokim i opóźniającym włączenie swojego wyjścia o określony czas (zadany parametrem PT),
- Off-Delay Timer (TOF) jest timerem opóźniającym wyłączenie swojego wyjścia o określony czas (zadany parametrem PT) od momentu gdy jego wejście będzie w stanie wysokim.

Używając ww. timerów można budować różnego rodzaju układy uzależnień czasowych. Podstawą budowy układów wykorzystujących timery jest fakt, że po wstawieniu do budowanego systemu timera o określonej nazwie, można następnie do układu wstawić styk o tej samej nazwie, zwierający się i rozwierający zgodnie ze stanem odpowiadającego mu timera.

Spójrzmy np. na poniższy układ:

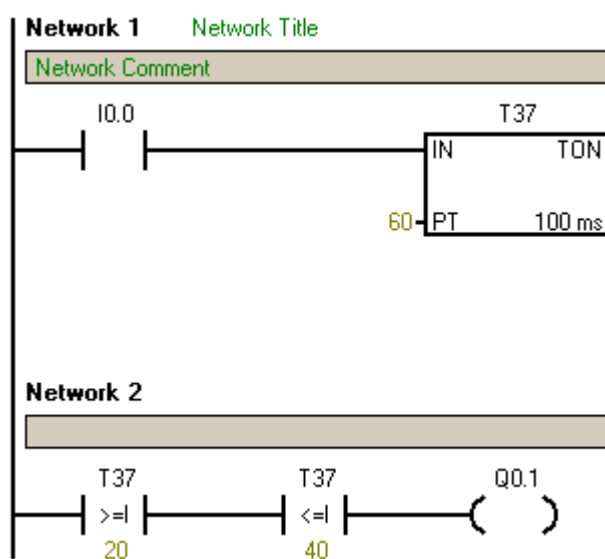


Jak on działa?

- Widzimy że timer T37 ma ustawiony parametr PT=40. Oznacza to, że odliczy on czas $40 \times 100\text{ms} = 4\text{s}$ i po tym czasie ustawi się w stan wysoki.
- Kiedy zacznie on liczyć? Wtedy kiedy na wejściu IN timera pojawi się sygnał, czyli wtedy kiedy użytkownik załączy wejście I0.0.
- W sieci 2 (Network 2) widzimy że wyjście Q0.0 jest zależne od T37, czyli że załączy się ono wtedy, kiedy timer T37 będzie w stanie wysokim.
- **Podsumowując** – wyjście Q0.0 po włączeniu układu będzie wyłączone. Kiedy stan wejścia I0.0 zmieni się na wysoki timer T37 zacznie zliczać i po zliczeniu do 4 sekund załączy on odpowiadający mu styk T37 i wtedy zaświeci się wyjście Q0.0.

Przy używaniu timerów należy pamiętać o tym, że ich okres podstawowy zależy wprost od numeru timera. Jeżeli nadamy timerowi TON lub TOF numer z zakresu 37-63 lub 101-255 to okres ten jest równy 100ms (widoczny w prawym dolnym rogu timera na rysunku powyżej), jeżeli będzie tu numer z zakresu 33-36 lub 97-100 to 10 ms, natomiast gdy 32 lub 96 to jest on równy 1ms. Oznacza to przykładowo, że gdybyśmy w układzie z rysunku powyżej nadali timerowi nazwę nie T37, a np. T33, to by zachować jego czas odliczania równy 4s, należałoby zmienić jego parametr PT na 400 (bo $400 \times 10\text{ms} = 4\text{s}$).

Przy używaniu układów timerów przydatne są styki z grupy *Compare*, czyli porównujące. Spójrzmy na układ na poniższym rysunku:

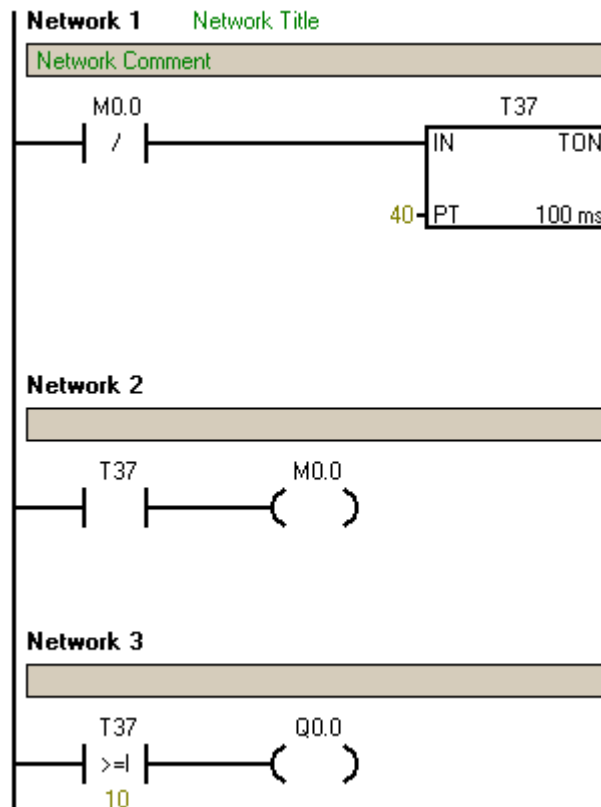


Jak on działa?

- Podobnie jak w poprzednim układzie – timer T37 zacznie liczyć po ustawieniu I0.0 w stan wysoki. Tym razem timer T37 ma parametr PT ustawiony na 60, czyli że ustawi się on w stan wysoki po $60 \times 100\text{ms} = 6\text{s}$.
- W sieci 2 (Network 2) widzimy że wyjście Q0.1 jest zależne od T37, ale tym razem nie wprost, ale przez dwa styki porównujące. Styk lewy zewrze się gdy T37 będzie większe/równe 20, natomiast styk prawy zewrze się gdy T37 będzie mniejsze/równe 40. Oznacza to że wyjście Q0.1 będzie się świecić tylko wtedy, gdy stan timera T37 będzie się mieścił w zakresie $\langle 20, 40 \rangle$.
- Podsumowując** – wyjście Q0.1 po włączeniu układu będzie wyłączone. Kiedy stan wejścia I0.0 zmieni się na wysoki timer T37 zacznie zliczać, po zliczeniu do 2 sekund wyjście Q0.1 zaświeci się, po czym wyłączy się po tym jak timer T37 doliczy do 4 sekund.

Z poprzedniego ćwiczenia pamiętamy, że jeżeli w układzie używamy jakiegoś wyjścia, to możemy go też używać jako styku. W sterownikach PLC niekoniecznie musi być to od razu wyjście fizyczne (czyli dla S7-200 któreś z Q0.0-Q0.5), może być to bit pamięci. Bity pamięci w sterowniku S7-200 są oznaczane jako Mx.x, w zakresie od M0.0 do M31.7.

Spójrzmy zatem jak wykorzystać bity pamięci do stworzenia timera samoresetującego się, działającego niezależnie od wejść układu:



Jak działa ten układ?

- Timer T37 ma narzucony czas opóźnienia włączenia $40 \times 100 \text{ ms} = 4 \text{ s}$. Tym razem zacznie on liczyć natychmiast po włączeniu sterownika, ponieważ załączający go styk M0.0 jest stykiem rozwiernym, czyli że domyślnie jest zwarty.
- W sieci 2 (Network 2) widzimy że bit M0.0 jest zależny od T37. Oznacza to, że w momencie gdy timer T37 doliczy do 4s, bit M0.0 ustawi się w stan wysoki. To spowoduje, że wyżej (w sieci 1) sterowany nim styk rozwierny (negacja M0.0) wyzeruje timer T37, który z kolei przywróci zero na M0.0 i układ wróci do stanu wyjściowego. Po kolejnym zliczeniu do 4s sytuacja się powtórzy. Zbudowaliśmy zatem układ generatora samoresetującego się.
- W sieci 3 (Network 3) widzimy że wyjście Q0.0 jest zależne od timera T37, podobnie jak w poprzednim przykładzie. Będzie ono działać następująco: sekunda wygaszenia, 3 sekundy świecenia, sekunda wygaszenia... i tak w kółko.

3. Jak zaprogramować sterownik S7-200 używając programu STEP 7-Micro/WIN?

3.1. Uruchomić program **STEP 7-Micro/WIN**.

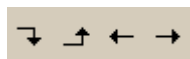
3.2. W oknie *Simatic LAD* zbudować projekt.

3.2.1. Po lewej stronie okna programu dostępne są elementy możliwe do wykorzystania, np. styki zwierny i rozwierny oraz wyjścia są w katalogu *Bit Logic*, styki porównujące w katalogu *Compare*, a timery w katalogu *Timers*.

3.2.2. Elementy wstawia się poprzez ich przeciągnięcie do sieci (*Network*).

3.2.3. Styki należy nazywać zgodnie z opisem na sterowniku (widoczne na obudowie), więc np. jeżeli wejściem ma być styk pierwszy od lewej należy domyślnie wstawiony nad stykiem opis „??.” zastąpić opisem „I0.0”. Podobnie jest z wyjściami.

3.2.4. Połączenia drabinki tworzą się w wierszach styków automatycznie, natomiast połączenia między wierszami najłatwiej tworzyć korzystając z przycisków nad oknem drabinki:



3.3. Gdy projekt jest gotowy należy go skompilować: *PLC->Compile* lub przycisk



3.3.1. Jeśli kompilacja była bezbłędna można projekt załadować do sterownika. W tym celu:

3.3.2. Zasilany i podłączony do komputera kablem PPI sterownik ustawić w **tryb STOP** (przełącznik jest pod klapką widoczną z przodu sterownika).

3.3.3. Przesłać program do sterownika: *File->Download* lub przycisk



3.4. Gdy program zostanie przesłany do sterownika (zależnie od wielkości programu zajmie to kilka lub więcej sekund) można przełączyć sterownik w **tryb RUN** i... powinien on już działać, oby zgodnie z projektem.

4. Przebieg ćwiczenia

4.1. CZĘŚĆ A - Należy zbudować i sprawdzić działanie układu kombinacyjnego zadanego przez prowadzącego.

4.2. CZĘŚĆ B - Należy zbudować i sprawdzić działanie wykorzystującego timery układu zadanego przez prowadzącego.

5. Literatura

- [1] Bryan L.A., Bryan E.A., „Programmable Controllers Theory And Implementation”, ITC, Atlanta 1997
- [2] Broel-Plater B., „Układy wykorzystujące sterowniki PLC : projektowanie algorytmów sterowania”, PWN, Warszawa 2009
- [3] Chochowski A., Cieślak H., Laskowski P., Mirski T., „Laboratorium automatyki”, Wydawnictwo SGGW, Warszawa 1999
- [4] Flaga S., „Programowanie sterowników PLC w języku drabinkowym”, Wydawnictwo btc, Legionowo 2010
- [5] Król A., Moczko-Król J., „S5/S7 Windows : programowanie i symulacja sterowników PLC firmy Siemens”, Nakom, Poznań 2003
- [6] Kwaśniewski J., „Sterowniki PLC w praktyce inżynierskiej”, Wydawnictwo btc, Legionowo 2008
- [7] Mikulczyński T., „Automatyzacja procesów produkcyjnych: metody modelowania procesów dyskretnych i programowania sterowników PLC”, WN-T, Warszawa 2006
- [8] Nowakowski W., „LOGO! w praktyce”, Wydawnictwo btc, Warszawa 2006
- [9] Sałat R., Korzys K., Obstawski P., „Wstęp do programowania sterowników PLC”, WKiŁ, Warszawa 2010