ISSN 1232-3829
tts 10/2013
Technika Transportu Szynowego koleje ▪ tramwaje ▪ metro

SUMOREK Andrzej, BUCZAJ Marcin, HORYŃSKI Marek, BOGUTA Artur, STYŁA Sebastian

DOBÓR JĘZYKA PROGRAMOWANIA KONTROLERA PLC

Streszczenie

Jeśli zadanie stworzenia oprogramowania dla kontrolera PLC dotyczy nowego, oryginalnego układu sterowania, to programista lub automatyk stają przed problemem doboru adekwatnego języka programowania tego kontrolera. W wielu przypadkach język i środowisko programowania narzucane są przez producenta rodziny kontrolerów. Tam gdzie decyzja należy do inżyniera, na nim spoczywa odpowiedzialność za dobór najefektywniejszej metody programowania.

W tekście zamieszczono charakterystykę wszystkich typowych metod programowania kontrolerów PLC. Opis zilustrowano przykładem praktycznym, w którym sterowanie urządzenia zrealizowano opisywanymi wcześniej metodami.

WSTĘP

Widok kontrolerów PLC w rozdzielnicach przemysłowych czy domowych nie jest aktualnie niczym nadzwyczajnym. Pierwsze prace projektowe nad kontrolerami PLC datuje się na rok 1968. Od momentu, w którym General Motors podjęło decyzję o opracowaniu sterowników swobodnie programowalnych, rozwinięto je w zakresie łatwości programowania i przeprogramowywania, niezawodności oraz obniżono koszty produkcji i użytkowania [10, 11]. Za pierwszy komercyjny sterownik uznaje się model Modicon 084 z roku 1969 [13].

Równocześnie z rozwojem platformy sprzętowej podążał rozwój metod programowania kontrolerów. Początkowo możliwość programowania ograniczona była do osób posługujących się językami niskiego poziomu. Programowanie kontrolera dostępne było z poziomu dedykowanego komputera. Jednocześnie nie występował standard programowania tej grupy urządzeń. Funkcjonalność uzależniona była od podejścia producenta. Programowanie wymagało szczegółowego zapoznania się z parametrami technicznymi konkretnego kontrolera.

Za rzeczywisty powód rozpowszechnienia się kontrolerów należy uznać pojawienie się możliwości ich programowania za pomocą standardowych komputerów osobistych. Od kwalifikacji i doświadczenia użytkownika zależy wybór metody programowania. Przykładowo grupa posiadająca wcześniej duże doświadczenie w programowaniu asemblerowym wybiera tekstowe języki programowania [9]. Kadra techniczna z wykształceniu o profilu elektrycznym wybierać będzie graficzny język drabinkowy, w którym znajdzie odzwierciedlenie znanych sobie takich elementów jak styki i cewki.

Dostępnych jest pięć metod programowania kontrolerów PLC, z których cztery można uznać za metody programowania tekstowego lub graficznego. Są to następujące sposoby deklarowania działania kontrolera: Instruction List, Structured Text, Ladder Diagram, Function Block Diagram, Sequential Function Chart [3, 12].

1. CHARAKTERYSTYKA METOD PROGRAMOWANIA

Pierwszym czynnikiem, który przyczynił się do spopularyzowania kontrolerów PLC jest możliwość ich programowania za pośrednictwem komputerów osobistych. Drugim czynnikiem było wydanie normy IEC 1131. Z jednej strony definiuje ona kontroler PLC od strony sprzętowej. Z drugiej definiuje metody programowania kontrolerów pod kątem opisu modelu oprogramowania, stosowanych instrukcji, typów danych, składni, wizualizacji [7, 9, 15]. W normie wyróżniono następujące tekstowe i graficzne metody programowania kontrolerów:

– IL - Instruction List - lista instrukcji, język tekstowy;
– ST - Structured Text - tekst strukturalny, język tekstowy;
– LD, LAD - Ladder Diagram - schemat drabinkowy, język graficzny;
– FBD - Function Block Diagram - schemat blokowy, język graficzny;
– SFC - Sequential Function Chart - graf sekwencji, graficzny sposób organizacji kodu. Ponieważ dedykowany jest głównie do organizowania (porządkowania) kodu, a nie samodzielnego tworzenia programów, nie będzie przedmiotem dalszego opisu i analizy.

1.1. Instruction List

Język Instruction List jest podobny do asemblera, który jest językiem niskiego poziomu. Zawiera operacje logiczne, arytmetyczne, relacje porównać, obsługuje funkcje timerów, przerzutników, liczników. Kod języka jest mało przejrzysty w porównaniu z ST czy LD. Zaletą IL jest to, że wygenerowany kod zajmuje miejsca [8].

Podobnie jak w asemblerze, procesy oparte są na akumulatorze procesora. Akumulator to rejestr w pamięci. Do akumulatora wprowadzane są wartości zmiennych oraz wykonywane na nich operacje, których wynik zapisywany jest w pamięci.

Program składa się z szeregu następujących instrukcji wyspecyfikowanych w kolejnych wierszach. Instrukcja składa się z operatora i operandu. Operator określa, jakie kolejne czynności mają być wykonane w programie. Operand jest elementem reprezentującym wartości stałych i zmiennych. Możliwe jest zamieszczanie komentarzy, które opisują zawartość knkretnej linii programu. Poniżej zamieszczono przykładowe wiersze programu w języku IL.

```
Operator Operand Komentarz
LD       I0.0    (* załaduj wartość z wejścia I0.0 *)
ST       Q1.0    (* wyślij wartość na wyjście Q1.0 *)
```

IL należy do języków niskiego poziomu. Jest rzadko stosowany do zapisu całego algorytmu. Nadaje się do kodowania złożonych algorytmów obliczeniowych, które zajmują wtedy mniej miejsca, są zwarte i proste oraz są szybciej wykonywane przez procesor [13].

1.2. Structured Text

Język Structured Text kwalifikowany jest do grupy języków tekstowych wysokiego poziomu. Elementami podstawowymi dla tego języka są operatory arytmetyczne i logiczne oraz instrukcje. Dostępne są operatory dzielenia, mnożenia, dodawania, odejmowania. Możliwe są działania na zmiennych binarnych, stosowanie pętli, czy instrukcji wyboru (np.: AND, OR, XOR, if ... then ... else, case ... of, while ... do ..., for ... to ... do, repeat ... until). Poniżej zamieszczono przykładowe wiersze programu w języku ST wykorzystujące instrukcję warunkową oraz pętlę.

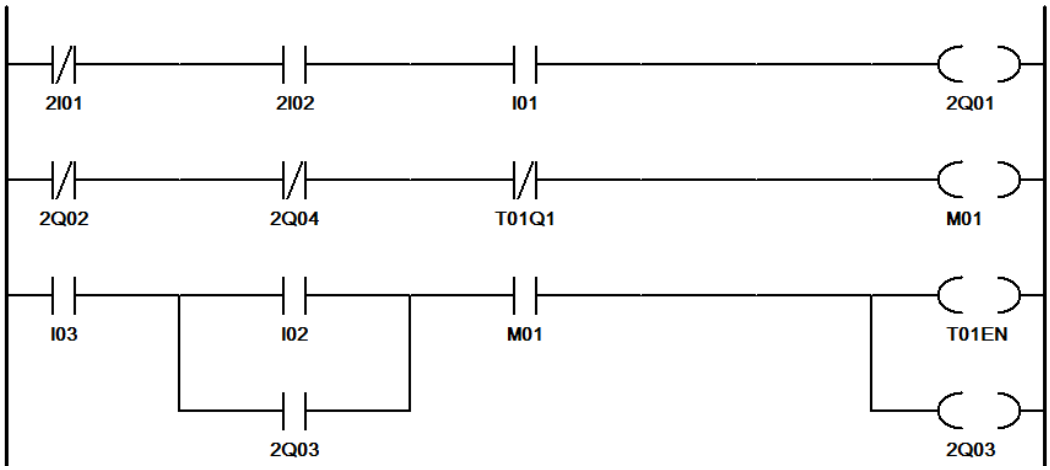
```
IF value <2 THEN
  WHILE wartosc <4 DO
    wartosc: = wartosc +1
  END WHILE;
END IF;
```

Podobnie jak inne języki wysokiego poziomu Instruction List posiada składnię i słowa kluczowe pozwalające na stosunkowo łatwe zrozumienie kodu. Istnieje jednak konieczność kompilacji takiego programu, co powoduje, że program staje się zwykle dłuższy i wolniejszy niż przykładowy odpowiednik w Instruction List. Programista nie ma bezpośredniego wpływu na przekształcanie kodu programu na postać maszynową [4, 8].

1.3. Ladder Diagram

Ladder Diagram należy do grupy języków graficznych. Jego podstawową zaletą jest intuicyjny sposób tworzenia i analizy programów. Ten rodzaj języka wybierają najczęściej osoby, którzy zaczynają programowanie sterowników PLC oraz osoby projektujące dotąd układ stycznikowo-przekaźnikowe. Diagram drabinkowy bazuje na symbolach zapożyczonych ze schematów elektrycznych zawierających przekaźniki i styczniki.

Program zawiera symbole przedstawiające styki, cewki lub bloki funkcyjne. Symbole te układane są w programie w sposób przypominający kolejne stopnie drabiny i stąd pochodzi nazwa [4]. Diagram ograniczony jest z lewej i prawej poprzez szyny zasilające. Prawa szyna zasilająca nie musi występować jawnie. Idea wykonywania programu w języku LD polega na „przepływie prądu" pomiędzy szynami – od lewej do prawej i od góry do dołu. Wszystkie wejścia lub wyjścia instrukcji programu posiadają własny adres, określający lokalizację w pamięci sterownika. W pamięci odwzorowuje się także stan wejść i wyjść kontrolera.



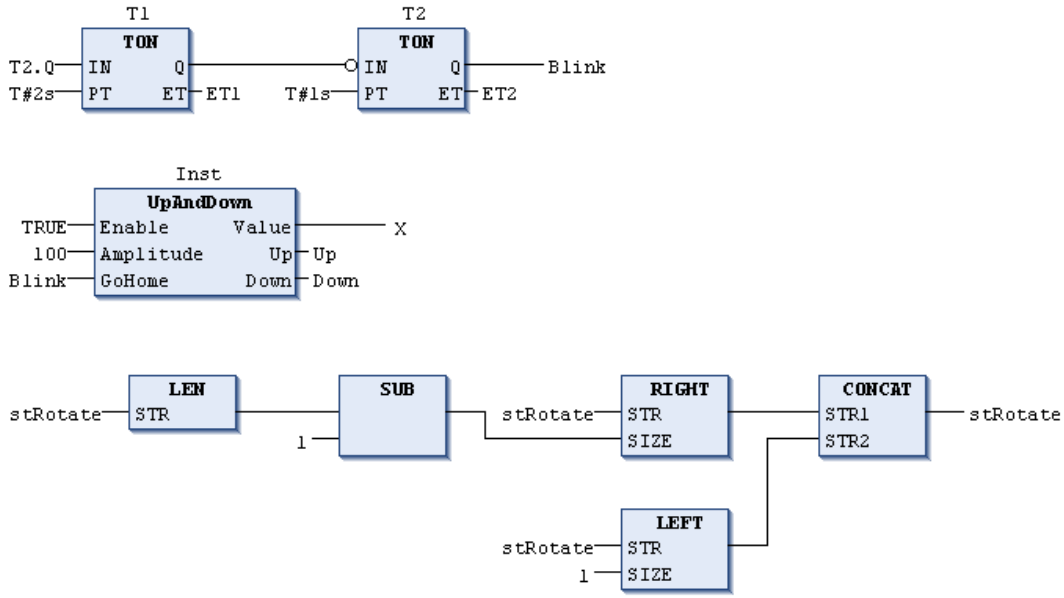
Rys. 1. Fragment kodu programu zrealizowanego w języku Ladder Diagram

Ladder Diagram pozwala na używanie gotowych funkcji oraz bloków funkcyjnych (timer, licznik). Wykorzystuje je się wtedy, gdy nie można zrealizować programu za pomocą standardowych funkcji logicznych. Mogą to być również bloki określone przez użytkownika. W takim przypadku funkcje i bloki na schemacie przedstawiane są w formie prostokątów. Każdy blok musi posiadać jedno wejście i wyjście binarne, aby możliwy był przepływ sygnału („przepływ prądu").

1.4. Function Block Diagram

Wykonywanie programu utworzonego w języku graficznym Function Block Diagram polega na „przepływie sygnału". Przepływ sygnału realizowany jest w taki sposób, że sygnał przepływa z wyjścia jednego bloku do wejścia kolejnego bloku funkcjonalnego lub funkcji.

Bloki symbolizuje się obiektami prostokątów oraz elementów sterujących. Łączone są one liniami poziomymi i pionowymi.



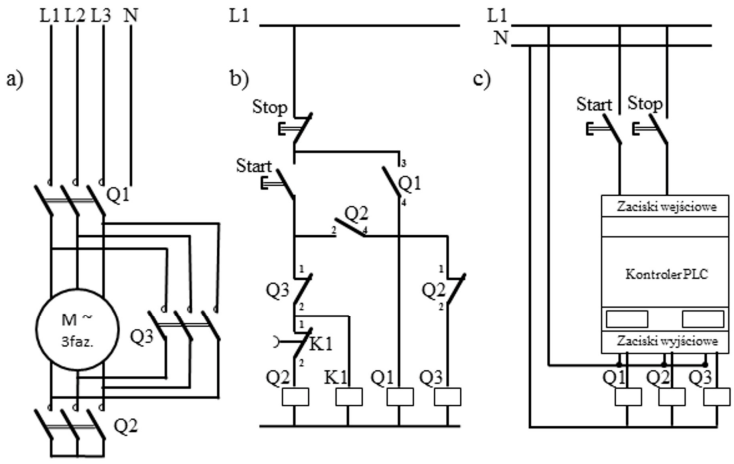
Rys. 2. Fragment kodu programu zrealizowanego w języku Function Block Diagram [2]
Źródło: [2]

Programy w FBD są zbliżone do zapisanych za pomocą języka LD. W języku FBD nie występują symbole styków i cewek. Odpowiadającym im elementem są zmienne. Tworzenie kodu programu z użyciem elementów graficznych jest metodą, która pozwala w sposób intuicyjny przełożyć schemat elektryczny układu sterowania na kod programu.

2. REALIZACJA UKŁADU STEROWANIA

W tekście przedstawiono najpierw stycznikowo-przekaźnikowy układ rozruchu silnika trójfazowego asynchronicznego z przełącznikiem gwiazda-trójkąt, zaś po nim zamieszczono układy sterujące pracą kontrolera PLC wykonane w czterech opisanych wcześniej językach. W celu ograniczenia wartości natężenia prądu rozruchowego, układ elektryczny najpierw kojarzony jest w gwiazdę, a po upływie zadeklarowanego czasu, przełączany jest w konfigurację trójkąta. Od strony bezpieczeństwa ważne jest, aby wykluczyć jednoczesne przyłączenie obu układów (rys. 3a). W celu uproszenia analizy, ze schematu zostały usunięte elementy zabezpieczenia przeciwporażeniowego, przeciwzwarciowego oraz przeciążeniowego. Główne elementy układu sterowania stanowią 3 styczniki z cewkami oznaczonymi Q1, Q2 i Q3 oraz przekaźnik czasowy oznaczony K1.

W układzie przedstawionym na rys. 3b, po naciśnięciu przycisku Start pod napięciem cewka znajdą się przekaźnika czasowego K1 i stycznika Q2. Styk 3-4 stycznika podadzą napięcie na cewkę stycznika Q1. Styk rozwierny 1-2 stycznika Q2 zablokuje możliwość załączenia stycznika Q3. Styk 3-4 stycznika Q1 podtrzyma napięcie na cewkach styczników Q2, Q1 i przekaźnika K1. W momencie zamknięcia styków głównych stycznika Q1 rozpoczyna się rozruch silnika z uzwojeniami silnika połączonymi w gwiazdę zasilonymi przez styki główne stycznika Q2.



Rys. 3. Schemat silnika trójfazowego uruchamianego w układzie przełącznika gwiazda-trójkąt: a) obwód główny zrealizowany w technice stycznikowo-przekaźnikowej, b) obwód sterowania, c) obwód sterowania zrealizowany dzięki kontrolerowi PLC [14]

Źródło: [14]

Po zadeklarowanym na przekaźniku czasie rozruchu, nastąpi otwarcie styku zwłocznego 1-2 stycznika K1, co spowoduje wyłączenie stycznika Q2. Zamykający się styk bierny 1-2 stycznika Q2 załączy napięcie na cewkę stycznika Q3. Styki główne stycznika Q3 połączą uzwojenia silnika w trójkąt. Zatrzymanie silnika w dowolnej chwili rozruchu lub w czasie pracy może nastąpić przez celowe naciśnięcie przycisku wyłączającego Stop lub po zaniku napięcia.

Układ stycznikowo-przekaźnikowy może zostać zastąpiony prostszym układem z kontrolerem PLC (rys. 3c). W takim układzie wszystkie cewki napędowe styczników podłączane są w taki sam sposób. Styki sterujące (tutaj Start i Stop) podłączane są wejść binarnych. Pod kątem połączeń elektrycznych jest to układ znacznie prostszy.

2.1. Instruction List

W przedstawionych dalej realizacjach programów dla kontrolera PLC sterowanie wykonywane jest za pomocą dwóch zmiennych binarnych Start i Stop. Kontroler załącza i wyłącza 3 cewki styczników Q1, Q2 i Q3 z rys. 3c. Sekwencję układu sterowania przedstawiono poniżej:

- Sprawdzenie stanu przycisków (zmiennych) Start oraz Stop.
- Załączenie stycznika Q1 w przypadku naciśnięcia przycisku Start i otwartego przycisku Stop (podanie napięcia na układ).
- Zainicjowanie odmierzenia czasu impulsu uruchamiania silnika w układzie gwiazdy (blok funkcyjny TP).
- Zainicjowanie odmierzenia czasu opóźnienia uruchamiania silnika w układzie trójkąta (blok funkcyjny TON).
- Załączenie stycznika Q2 - układu gwiazdy (jeśli spełniony jest warunek odmierzania czasu impulsu TP wraz z wyłączonym stycznikiem Q3).
- Zainicjowanie odmierzenia czasu opóźnienia uruchamiania silnika w układzie trójkąta (blok funkcyjny TON).
- Odłączenie zasilania cewki stycznika Q2 po upływie czasu impulsu z bloku TP.
- Załączenie stycznika Q3 po upływie czasu zadeklarowanego w TON – praca silnika w układzie trójkąta.

Działanie silnika można zatrzymać w dowolnym momencie deklarując wartość binarną „1” zmiennej Start (zamykając styki przycisku). Dwa czasy występujące w algorytmie sterowania oznaczano jako „opoz1”, „opoz2” lub jawnie deklarowano ich wartości na 3 i 4

sekundy. Jednocześnie należy pamiętać, że brak pętli w programach wynika z cyklicznej pracy właściwej sterownikom PLC.

Poniżej zamieszczono kod programu w Instruction List. Początek programu to deklaracja binarnych zmiennych wejściowych i wyjściowych oraz wartości timerów. Pierwsza sekcja programu (3 wiersze) składa się z trzech instrukcji LDN, AND oraz ST. Iloczyn logiczny zanegowanego styku Stop i styku Start powoduje wysłanie wartości 1 na wyjście Q1 (ST Q1), co w praktyce oznacza załączenie styków głównych stycznika Q1 i podanie napięcia na układ.

```
VAR                (* deklaracja zmiennych I0.0 *)
   opoz1: TP;
   opoz2: TON;
END_VAR
VAR_INPUT
   STOP:  BOOL;
   START: BOOL;
   czas1: TIME;
   czas2: TIME;
END_VAR
VAR_OUTPUT
   Q1: BOOL;
   Q2: BOOL;
   Q3: BOOL;
END_VAR

LDN  STOP            (* program *)
AND  START
ST   Q1
LD   TRUE
AND  Q1
ST   opoz1.IN
CAL  opoz1(PT:=czas1)
LD   opoz1.Q
ANDN Q3
ST   Q2
LD   TRUE
AND  Q1
ST   opoz2.IN
CAL  opoz2(PT:=czas2)
LD   opoz2. Q
ANDN Q2
ST   Q3
```

Celem opisywanej dalej części programu jest załączenie wyjścia Q2 (cewki stycznika Q2) odpowiedzialnego za czasowe załączenie układu skonfigurowanego w gwiazdę. Uzyskanie przez iloczyn logiczny wartości TRUE i zmiennej Q1 powoduje zainicjowanie generowania impulsu przez pierwszy z timerów. Instrukcja CAL służy do wywoływania bloków funkcyjnych. W tym przypadku bloku TP zadeklarowanego jako „opoz1". PT oznacza wejście timera TP służące do ustawienia czasu działania „czas1".

W czasie kiedy iloczyn stanu timera (opoz1.Q) i zanegowanego stanu wyjścia Q3 wynosi 1 (true) zasilana jest cewka stycznika Q2 (ST Q2, praca w konfiguracji w gwiazdę).

Od tego momentu zaczyna się deklaracja czasu opóźnienia załączenia w trójkąt. Czas opóźnienia jest odliczany. Jeśli wystąpi sytuacja upłynięcia czasu opóźnienia oraz wyłączenia

wyjścia Q2, następuje załączenie wyjścia Q3(LD opoz1.Q ANDN Q2), co prowadzi do pracy silnika w układzie trójkąta.

2.2. Structured Text

W zamieszczonym dalej kodzie programu w języku Structured Text pominięto procedury deklarowania zmiennych i skupiono się na samym modzie programu. Został on podzielny na trzy sekcje.

W pierwszej sekcji występuje zmienna Stop (zmienna przedstawiająca stan wyłącznika awaryjnego) oraz zmienna Start wskazująca na naciśnięcie przycisku uruchamiającego. Wartość logiczna TRUE dla koniunkcji zmiennych Start i Stop załącza wyjście Q1. W sekcji wykorzystano instrukcję warunkową If…Then..Else.

W drugiej sekcji uruchamiany jest pierwszy timer, który przez czas trwania impulsu uruchamia wyjście Q2 (silnik pracuje w układzie gwiazdy). Praca kontrolera sterowana jest taką samą instrukcją warunkową jak w pierwszej sekcji.

Instrukcja „opoz1(IN := Q1, PT:= T#3s)" wywołuje blok funkcji TP zadeklarowany jako „opoz1" wraz z parametrem wejściowym IN równe Q1. Oznacza to, że uruchomienie bloku następuje po zmianie stanu wyjścia Q1 po PT czyli po 3 s.

Instrukcja „opoz2(IN:=Q1, PT:=T#4s)" wywołuje blok funkcji TON zadeklarowany jako „opoz2" wraz z parametrem wejściowym IN równym Q1, co oznacza uruchomienie bloku po zmianie stanu wyjścia Q1. Nastawa czasu w tym przypadku to 4 s.

Trzecia sekcja jest skonstruowana identycznie jak druga. Różnica tkwi w innym sposobie pracy timera oraz zadeklarowaniu dla niego większej wartości czasu niż w sekcji drugiej. Po upłynięciu tego czasu podawany jest sygnał na wyjście Q3 odpowiedzialne za pracę silnika przy konfiguracji w trójkąt.

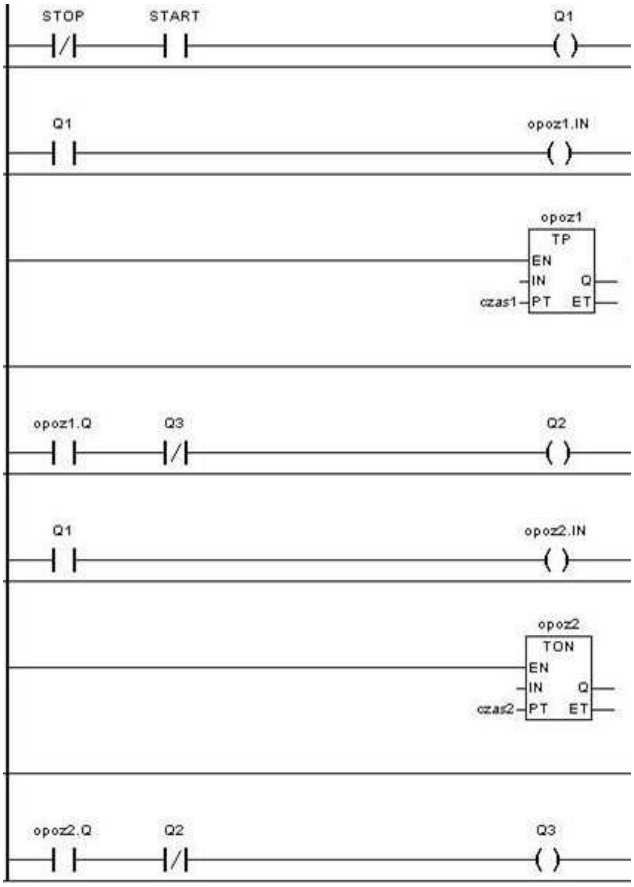
```
IF STOP =FALSE
AND START=TRUE
THEN Q1:=TRUE;
ELSE  Q1:=FALSE;
END_IF;

opoz1(IN := Q1, PT:= T#3s);
IF opoz1.Q=TRUE
THEN Q2:=TRUE;
ELSE Q2:=FALSE;
END_IF;

opoz2(IN:=Q1, PT:=T#4s);
IF opoz2.Q=TRUE
THEN Q3:=TRUE;
ELSE Q3:=FALSE;
END_IF;
```

2.3. Ladder Diagram

Początkowa faza funkcjonowaniu programu w Ladder Diagram polega na kontrolowaniu przycisków Start oraz Stop. Konfiguracja, w której Start pozostaje zwarty, zaś Stop rozwarty wygenerowanie wartości 1 na wyjściu oznaczonym Q1 (zasilany jest sytcznik Q1 i podawana na układ napięcie).

Funkcję opóźnionego wyłączenia stycznika gwiazdy (Q2) otrzymuje się przez użycie bloku funkcyjnego TP. Konieczne jest użycie bloku funkcyjnego gdyż nie ma możliwości realizacji funkcji czasowych za pomocą styków dostępnych w języku LD. Program składa się ze styku zwiernego odzwierciedlającego stan wyjścia Q1, który jednocześnie jest wejściem

IN timera TP. W trakcie generowania impulsu przez blok TP i przy jednocześnie rozwartym styku prezentującym wyjście Q3 (brak zasilania stycznika Q3), na wyjście Q2 odpowiadające załączeniu stycznika gwiazdy podawany jest stan 1.



Rys. 4. Kod oprogramowania układu sterowania rozruchem silnika zrealizowany w języku Ladder Diagram [5]

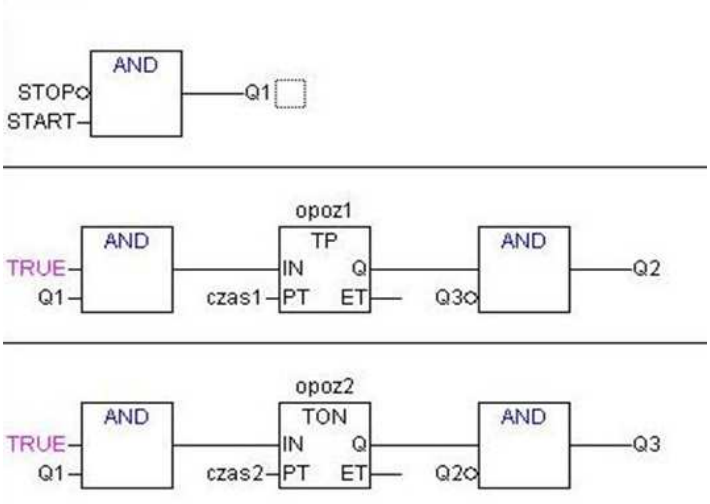
Źródło: [5]

Ostatnia sekcja programu składa się ze styku zwiernego odzwierciedlającego stan wyjścia Q1, które jednocześnie inicjuje działanie timera TON. Po odmierzeniu czasu opóźnienia przez blok TON, przy jednocześnie rozwartym styku prezentującym wyjście Q2, na wyjście Q3 podawany jest stan 1. Odpowiada to stanowi pracy silnika w konfiguracji uzwojeń w trójkąt.

2.4. Function Block Diagram

Budowa kodu w Function Block Diagram jest zbliżona do realizacji diagramu w LD. Początkowy blok AND realizuje koniunkcję stanu styków Stop oraz Start. W przypadku osiągnięcia przez koniunkcję wartości TRUE, na wyjściu Q1 pojawia się binarna 1. W praktyce załączany jest stycznik Q1.

Na początku drugiej sekcji sprawdzany jest stan wyjścia Q1 (czy załączono zasilanie układu). W przypadku pozytywnej odpowiedzi uruchamiany jest timer TP z czasem oznaczonym jako „opoz1". Przez czas „opoz1" timer TP generuje stan True. Jeśli równocześnie wyłączone jest wyjście Q3, to zostaje załączone wyjście Q2 (prowadzony jest rozruch w układzie gwiazdy).



Rys. 5. Kod oprogramowania układu sterowania rozruchem silnika zrealizowany w Function Block Diagram [5]

Źródło: [5]

Trzecia część programu optycznie przypomina drugą. Na początku sekcji sprawdzany jest stan wyjścia Q1 (czy załączono zasilanie układu). W przypadku pozytywnej odpowiedzi uruchamiany jest timer TON z czasem opóźnienia w stosunku do załączenia Q1 oznaczonym jako „opoz2". Po czasie „opoz2" timer TON generuje stan True. Jeśli równocześnie wyłączone jest wyjście Q2, to zostaje załączone wyjście Q3 (silnik pracuje w układzie trójkąta).

PODSUMOWANIE

Każdy z języków programowania posiada unikalne, specyficzne dla danego języka cechy. Przykładowo w przemyśle motoryzacyjnym dominuje programowanie za pomocą schematów drabinkowych. Mimo, że wielu programistów rozumie, że możliwe jest uzyskanie pożądanych efektów wydajniejszą metodą, to wybiera dotychczasowy sposób, traktując go jako metodę minimalizacji kosztów [6].

Na podstawie zgromadzonych przez autorów doświadczeń oraz w oparciu o przedstawioną prezentację czterech metod programowania kontrolera PLC można twierdzić, że:

- Tam gdzie ważne jest szybkie przedstawienie innym osobom sposobu oprogramowania urządzenia dominują języki graficzne tj. Ladder Diagram oraz Function Block Diagram.
- Języki graficzne sprawdzają się w oprogramowaniu mniejszych urządzeń czyli w przypadkach, gdy programista może łatwo kontrolować cały graficzny obraz oprogramowania.
- Osobom spoza kręgu zawodowych programistów najmniej czasu zajmie przyswojenie sobie programowania w języku drabinkowym. Kolejnym językiem pod kątem łatwości przyswajania jest Function Block Diagram, Structured Text. Instruction List uznawany jest za najmniej przyjazny dla użytkownika.
- Z punku widzenia tworzenia zaawansowanych algorytmów najwydajniejszym językiem jest Structured Text przypominający swą składnią po części języki C i Pascal, ponieważ umożliwia korzystanie z pętli, skoków warunkowych czy instrukcji wyboru.
- Zaletą języka Structured Text jest możliwość wykonywania w kodzie złożonych operacji matematycznych oraz łatwość przyswojenia przez użytkowników znających języki programowania używane w informatyce.

– Do zalet kodowania w Ladder Diagram zalicza się łatwość dokonywania zmian w algorytmie sterowania.
– Zaletą języków Instruction List i Structured jest szybkość wykonywania programu przez kontroler PLC. Instruction List jest podobny do assemblera, co pozwala na uzyskanie dużej wydajności obliczeń. Pamiętać należy, że możliwe do realizacji są tylko podstawowe operacje na danych (z wykorzystaniem akumulatora i stosu).
– Mimo, że w większości przypadków możliwa jest realizacja zadania w każdym z języków programowania, a nawet niekiedy edytory dają możliwość automatycznej konwersji jednego kodu na drugi, to niekiedy taka konwersja nie jest możliwa.
– Platforma sprzętowa może wymuszać na użytkowniku wybór konkretnego języka z powodu braku alternatywnych języków programowania. Zdarza się, że producenci nie oferują w swych produktach pełnej funkcjonalności oraz mają ograniczone możliwości programowania np. do dwóch języków.

Opisane powyżej zalety przeważają nad potencjalnymi problemami wynikającymi z programowania i użytkowania z kontrolerów PLC, o czym świadczy popularność kontrolerów w praktyce przemysłowej oraz domowej. Opracowane są rozwiązania polegające na automatycznym przeprogramowywaniu kontrolerów PLC traktowanym jako część procesu zintegrowanego procesu produkcyjnego. Zmiana ustawień urządzeń przemysłowych w locie ma za zadanie poprawić efektywność produkcji [1].

BIBLIOGRAFIA

1. Anosike A.I., Zhang D.Z., *Dynamic reconfiguration and simulation of manufacturing systems using agents*. Journal of Manufacturing Technology Management, 2006 nr 4 (vol. 17).
2. Beckhoff Information System, *Function Block Diagram - FBD*. http://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/html/Function_Block_Diagram_FBD.htm&id=, V 2013.
3. Broel-Plater B., *Sterowniki programowalne. Właściwości i zasady stosowania*. Wydawnictwo Uczelniane Politechniki Szczecińskiej, Szczecin 2003.
4. Flaga S., *Programowanie sterowników PLC w języku drabinkowym*. BTC, Legionowo 2010.
5. Gawron K., *Kontrolery PLC – problem wyboru metody programowania*. Rozprawa dyplomowa. Politechnika Lubelska, Lublin 2013.
6. Hajarnavis V., Young K., *An investigation into programmable logic controller software design techniques in the automotive industry*. Assembly Automation 2008, nr 1 (vol. 28).
7. Hoske M.T., *Choose the right programming language*. Control Engineering, 2003 nr 7 (vol. 50).
8. Kacprzak S., *Programowanie sterowników PLC zgodnie z normą IEC61131-3 w praktyce*. Wydawnictwo BTC, Legionowo 2011.
9. Kasprzyk J., *Programowanie sterowników przemysłowych*. WNT, Warszawa 2013.
10. Król A., Moczko-Król J., S5/S7 *Windows. Programowanie i symulacja sterowników PLC firmy Siemes*. Nakom, Poznań 2000.
11. Legierski T., Wyrwał J., Kasprzyk J., Hajda J., *Programowanie sterowników PLC*. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice 1998.
12. Łukasik Z., Seta Z., *Programowalne sterowniki PLC w systemach sterowania przemysłowego*. Wydawnictwo Politechniki Radomskiej, Radom 2004.
13. Pawlak M., *Sterowniki programowalne*. Politechnika Wrocławska, Wrocław 2010.
14. Pietrzyk W. (red.), *Laboratorium z elektrotechniki*. Wydawnictwa Uczelniane Politechniki Lubelskiej, Lublin 2003.

15. Sałat R., Korpysz K., Obstawski P., *Wstęp do programowania sterowników PLC*. Warszawa 2010.

THE SELECTION OF PLC PROGRAMMING LANGUAGE

Abstract

If the task of creating software for the PLC controller relates to a new, original control system, programmers have the problem of selecting of an adequate controller programming language. In many cases, the language and programming environment are enforced by the manufacturer of the controller family. If the decision is up to the engineer, he is responsible for the selection of the most effective methods of programming.

There is a characteristic of all common programming methods in the paper. The description is illustrated by a practical example in which the device control system is implemented using methods previously described.

Autorzy*:*
dr inż. **Andrzej Sumorek,** dr inż. **Marcin Buczaj,** dr inż. **Marek Horyński,** dr inż. **Artur Boguta,** mgr inż. **Sebastian Styła** – Politechnika Lubelska, Wydział Elektrotechniki i Informatyki, a.sumorek@pollub.pl