Jan CISEK, Leszek TRYBUS
POLITECHNIKA RZESZOWSKA, KATEDRA INFORMATYKI I AUTOMATYKI

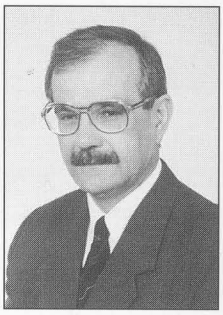
Automatyczna konfiguracja sekwencyjnego układu sterowania

Dr inż. Jan CISEK

Absolwent Wydziału Elektrycznego Politechniki Rzeszowskiej (1990). Pracę doktorską obronił z wyróżnieniem na Wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki AGH Kraków w 2004 r. Od 1994 jest asystentem, a obecnie adiunktem w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Jego zainteresowania skupiają się na problematyce weryfikacji poprawności oprogramowania sterowników. Jest autorem lub współautorem 8 publikacji.

e-mail: cisek@prz-rzeszow.pl

Prof. dr hab. inż. Leszek TRYBUS



Kierownik Katedry Informatyki i Automatyki Politechniki Rzeszowskiej. Ukończył AGH Kraków (1970), gdzie również uzyskał doktorat, habilitację i tytuł. Od ukończenia studiów pracuje w Politechnice Rzeszowskiej. Autor ok. 130 publikacji. Zajmuje się regulatorami mikroprocesorowymi, rozproszonymi systemami sterowania i zastosowaniami informatyki.

e-mail: ltrybus@prz-rzeszow.pl

Streszczenie

Przedstawiono metodę automatycznego tworzenia schematu bloków funkcyjnych dla sterownika PLC, przy użyciu języka funkcyjnego ML. Opisano podstawowe elementy metody na przykładzie sekwencyjnego układu sterowania. Rozważono przykład generacji układu sterowania przejazdem kolejowym ostatecznie zaimplementowany w sterowniku wielofunkcyjnym PSW-166.

Abstract

The method of automatic generation of function block diagram for a PLC using ML functional language is considered. A sequential control problem has been used to present fundamental elements of the method. Railway crossing control and PSW-166 multifunction controller are applied in the example.

Słowa kluczowe: metoda bloków funkcyjnych FBD, automatyczna konfiguracja, sterownik programowalny PLC, język funkcyjny ML.

Keywords: function block diagram (FBD), automatic configuration, PLC controller, ML functional language.

1. Wstęp

Sterowniki mikroprocesorowe, w tym programowalne sterowniki logiczne PLC (*Programmable Logic Controllers*) stały się obecnie powszechne. Znajdują zastosowania w energetyce, komunikacji i ochronie środowiska. W ostatnich latach opracowano normę IEC 61131 [1,2] standaryzującą programowanie sterowników. Jednym ze zdefiniowanych w niej języków programowania są schematy bloków funkcyjnych FBD (*Function Block Diagram*), w których algorytm sterowania przedstawia się graficznie w formie połączonych bloków.

W praktycznych aplikacjach dany jest słowny lub analityczny opis funkcjonowania układu stanowiący specyfikację. Zadaniem projektanta jest wygenerowanie konfiguracji w postaci schematu FBD. Jeżeli schemat FBD jest generowany automatycznie na podstawie wzoru analitycznego, to mamy do czynienia z tzw. automatyczną konfiguracją sterownika.

Jako platforma narzędziowa do rozwiązania postawionego problemu posłużył język funkcyjny ML [3]. Znajduje on zastosowanie do weryfikacji programów i modelowania sieci Petriego [4]. Powstał w nim także model sterownika PLC konfigurowanego metodą FBD [5,6].

Poniżej przedstawiono sposób realizacji układu sekwencyjnego wyspecyfikowanego jako automat Mealy'ego. Jako przykład posłużył układ sterowania zaporami na dwutorowym przejeździe kolejowym. Otrzymany automatycznie schemat bloków funkcyjnych zrealizowano w przemysłowym sterowniku mikroprocesorowym PSW-166 produkowanym przez ZPDA Ostrów Wlkp.

2. Specyfikacja analityczna

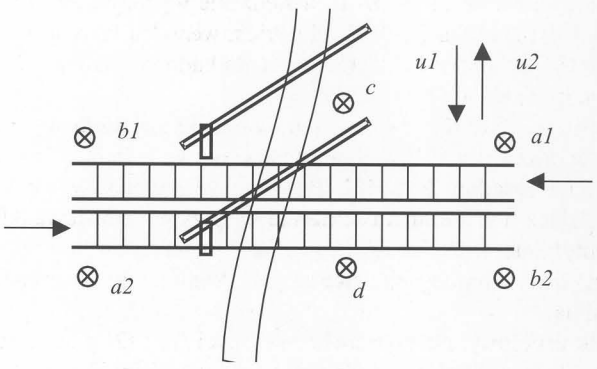
Specyfikacja. Opis funkcjonowania układu sterowania podaje się w języku naturalnym lub formalnym. Praca jest ograniczona do logicznych układów sekwencyjnych. Przyjęto, że specyfikacją układu tego typu jest zbiór formuł logicznych, które należy spełnić.

Automat Mealy'ego. Dalsze rozważania dotyczą realizacji asynchronicznego automatu Mealy'ego uważanego za bardziej ogólny od automatu Moore'a. Jest on specyfikowany przez podanie funkcji przejść i wyjść:

$$S_i = \delta(X_i, S_{i-1}) \quad - \text{ funkcja przejść},$$
$$Y_i = \lambda(S_i, X_i) \quad - \text{ funkcja wyjść},$$

gdzie X i Y są wektorami wejść i wyjść, a S oznacza zbiór stanów wewnętrznych. Funkcje δ i λ są kombinacyjne. W funkcji przejść argumentem oprócz wektora wejść jest wektor poprzednich stanów S_{i-1}. Jest on zwykle zapamiętywany w przerzutnikach, które będą tutaj jedynymi sekwencyjnymi elementami układu.

Przykład. W układzie sterowania zaporami na przejeździe kolejowym pokazanym na rys. 1 znajdują się czujniki sygnalizujące:
a1, a2 - zbliżanie pociągu jadącego w lewo lub w prawo,
b1, b2 - oddalanie pociągu,
c, d - stan otwarcia i zamknięcia zapór (wył. krańcowe).



Rys. 1. Dwutorowy przejazd kolejowy
Fig. 1. Dual track railway cross

Sygnały u1 i u2 sterują odpowiednio opuszczaniem i podnoszeniem zapór. Przyjęto, że na każdym torze ruch jest jednokierunkowy.

Analiza. Do rozwiązania zadania pomocne jest wprowadzenie dwóch dodatkowych sygnałów p1 i p2 informujących o tym, że w strefie przejazdu pomiędzy czujnikami a, b znajdują się pociągi jadące w lewo lub w prawo. Zakłada się, że p1 przyjmuje wartość true, gdy nadjeżdża pociąg z prawej (czujnik a1). Po opuszczeniu strefy przejazdu (czujnik b1) sygnał p1 wynosi false. Analogicznie definiuje się p2 dla drugiego toru. Sygnały p1 i p2 określają stan układu.

Na podstawie powyższej analizy można zdefiniować następujące funkcje przejść:

$$p1_i \Leftrightarrow a1_i \vee p1_{i-1} \wedge \neg b1_i, \quad p2_i \Leftrightarrow a2_i \vee p2_{i-1} \wedge \neg b2_i \quad (1a,b)$$

Jak widać, p1 określono jako alternatywę sygnału a1 oraz koniunkcji poprzedniej wartości p1 i negacji aktualnej wartości b1. Podobnie oblicza się wartość p2.

Sygnał sterowania opuszczaniem zapór u1 powinien się pojawić, gdy w strefie przejazdu znajduje się co najmniej jeden pociąg, a zapory nie są zamknięte. Zatem

$$u1_i \Leftrightarrow (p1_i \vee p2_i) \wedge \neg d_i \qquad (2)$$

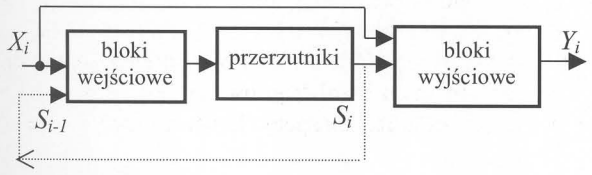
Jak zapisano w (2) aktualna wartość sterowania u1 zależy od bieżących wartości p1 i p2 reprezentujących stany automatu i wejścia d.

Podnoszenie następuje, gdy w strefie nie ma żadnego pociągu i zapory nie są w pozycji górnej, czyli

$$u2_i \Leftrightarrow \neg(p1_i \vee p2_i) \wedge \neg c_i \qquad (3)$$

3. Realizacja automatu sekwencyjnego

Budowa automatu. Ogólny schemat blokowy realizacji automatu Mealy'ego przez sterownik pokazano na rys. 2.



Rys. 2. Automat Mealy'ego
Fig. 2. Mealy automat

Bloki wejściowe wraz z przerzutnikami realizują funkcje przejść. Na wyjściach przerzutników są wartości odpowiadające stanom S_i. Bloki wyjściowe wyznaczają wartości wyjść Y_i.

Język ML. Narzędziem do automatycznej generacji konfiguracji będzie język funkcyjny ML posiadający następujące cechy:
- pełna kontrola zgodności typów,
- sprawdzanie kompletności wyrażeń i funkcji,
- wbudowany typ listowy,
- mechanizm generowania i obsługi wyjątków,
- uniezależnienie od platformy sprzętowej.

W pakiecie MOSML [7] oprócz kompilatora dostępny jest interaktywny interpreter. Ułatwia on śledzenie wykonywania programów oraz modyfikacje. W skład pakietu wchodzą także programy MOSMLLEX i MOSMLYAC służące do budowy analizatorów leksykalnych i składniowych.

Notacja. Specyfikację automatu zapisuje się tekstowo. Składnia wyrażeń jest zbliżona do stosowanej w systemach programowania układów PLD [8]. Przyjęto, że sygnały wewnętrzne określające stan automatu deklaruje się przy użyciu słowa STAN i identyfikatora. W funkcjach przejść i wyjść symbole /, +, * i ' (prim) oznaczają negację, alternatywę, koniunkcję i wartość poprzednią.

Plik źródłowy. Na podstawie zależności (1) i (2) można utworzyć plik specyfikujący zgodnie z regułami podanymi wyżej. Najpierw deklaruje się identyfikatory oznaczające stany wewnętrzne p1, p2, a następnie funkcje przejść i wyjść. Wzory (1) i (2) wyglądają teraz następująco

```
STAN p1                 (* definicja stanu          *)
STAN p2
p1 = a1 + p1' * /b1 ;(* funkcje przejść:  (1a)  *)
p2 = a2 + p2' * /b2 ;(*                   (1b)  *)
u1 = p1 + p2 * /d ;   (* funkcja wyjścia  (2)   *)
```

Specyfikacja u2 wygląda podobnie Powyższy plik specyfikujący zostanie dalej przetworzony na konfigurację.

Realizacja w sterowniku. Zakłada się, że obliczenia są wykonywane sekwencyjnie w pętli nieskończonej w następujący sposób:
- pobranie wartości wejść obiektowych,
- obliczenie kolejnych bloków schematu FBD,
- ustawienie wyjść obiektowych.

W realizacji przyjęto, że podstawowym elementem sekwencyjnym układu jest przerzutnik typu D, który wraz ze specyfikacją pokazano na rys. 3.

$$q_i \Leftrightarrow (c_i \wedge \neg c_{i-1}) \wedge d_i \vee \neg(c_i \wedge \neg c_{i-1}) \wedge q_{i-1}$$

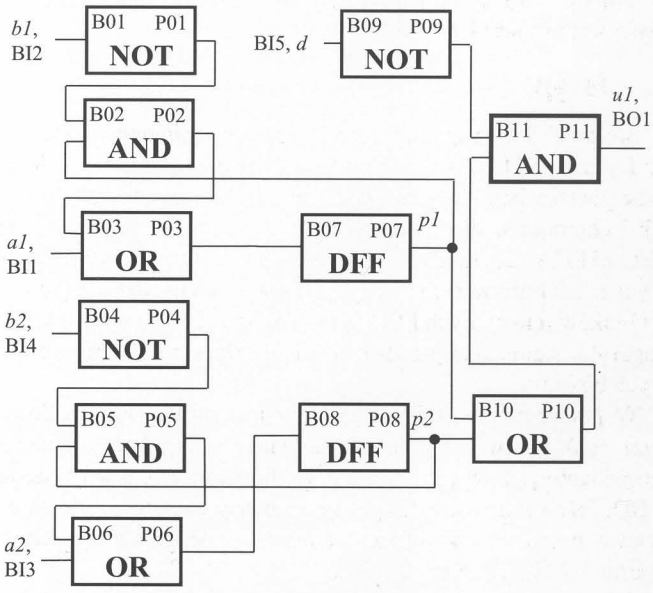
Rys. 3. Przerzutnik D i jego specyfikacja
Fig. 3. D flip flop and its specification

Wartość wejścia d jest przepisywana na wyjście q przy narastającym zboczu zegara c. W przeciwnym razie q zachowuje wartość poprzednią. W układzie asynchronicznym przy pominięciu sygnału zegarowego c ($c_i \wedge \neg c_i \Leftrightarrow$ true) przerzutnik D w każdym kolejnym cyklu sterownika przepisuje sygnał wejściowy na wyjście ($c_i \Leftrightarrow d_i$).

Przekształcenia. Do przetwarzania pliku specyfikującego i tworzenia konfiguracji powstał program utworzony z pomocą narzędzi MOSMLLEX i MOSMLYAC. Program ten wykonuje następujące zadania:
- analiza leksykalna i składniowa deklaracji automatu, wyodrębnienie symboli i analiza funkcji,
- przypisanie symboli wejść i wyjść (zapisywanych w listach) do wejść i wyjść obiektowych sterownika,
- realizacja kombinacyjnych funkcji przejść (wzbudzeń przerzutników D) i funkcji wyjść za pomocą bloków funkcyjnych,
- generacja schematu FBD układu sterowania.

Do przetwarzania użyto listowych typów danych, funkcji rekurencyjnych oraz własnych typów danych. Otrzymany program dzięki cechom języka ML jest względnie krótki i nie ma ograniczeń na rozmiar realizowanej konfiguracji.

Konfiguracja. Po uruchomieniu programu otrzymuje się plik z konfiguracją sterownika przedstawioną graficznie na rys. 4.



Rys. 4. Układ sterowania zaporami na przejeździe kolejowym
Fig. 4. Control system of railway cross

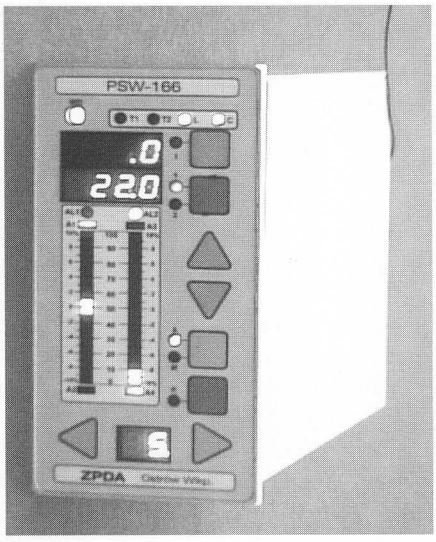
Postać schematu jest podobna jak dla sterowników SIPART Siemensa [9] i zgodna z metodą bloków funkcyjnych FBD. W lewym górnym rogu bloku znajduje się jego nazwa, np. B01, a w prawym tzw. "pozycja", np. P01, określająca kolejność obliczeń.

Bloki B01-B03 wyznaczają funkcję wzbudzenia przerzutnika D (B07) zdefiniowaną w (1a). Na wyjściu B07 otrzymuje się sygnał p1. Do wejścia bloku B02 przyłączono wyjście bloku B07, który w bieżącym cyklu nie był jeszcze wykonywany (sprzężenie zwrotne), gdyż pozycja P07 następuje po P02. Zatem wyjście to reprezentuje wartość p1 z poprzedniego cyklu. Analogiczną funkcję pełnią bloki B04-B06 i B08 dla sygnału p2 realizując zależność (1b). Funkcję wyjścia (2) realizują bloki B09, B10 i B11.

4. Implementacja w sterowniku PSW

Opisywane algorytmy i programy zostały przystosowane do sterownika wielofunkcyjnego PSW [9].

Cechy ogólne i zastosowania. Wytwarzane od 11 lat sterowniki PSW (rys. 5) są urządzeniami aparatowymi konfigurowanymi z bloków funkcyjnych. Liczba sygnałów obiektowych w PSW-166 (3-ciej generacji) wynosi 56. Łącząc kilka sterowników magistralą CAN można realizować systemy rozproszone liczące 300-500 sygnałów.



Rys. 5. Sterownik wielofunkcyjny PSW-166
Fig. 5 PSW-166 multifunction controller

Charakterystyka konfiguracji. Konfiguracja PSW jest podawana tekstowo w postaci listy. Jej elementy są następujące:
- nazwy etapów konfiguracji: DEFN, CONN, POSN (definiowanie, łączenie i kolejność)
- oznaczenia bloków B01, B02...,
- wejścia, np. B01.1 (pierwsze wejście bloku pierwszego)
- wyjścia, np. B01.b (boolowskie wyjście bloku pierwszego)
- funkcje: NOT, AND, OR,...
- kolejność obliczeń (pozycja): P01, P02, ...
- wejścia BI1, BI2,... wyjścia obiektowe BO1,B02,... .

Część z tych oznaczeń występuje na rys. 5. Dla układu sterującego przejazdem kolejowym lista konfiguracyjna ma postać podaną w tabl. 1. Transformacja schematu graficznego może być wykonana przez graficzny program konfiguracyjny [10].

Lista konfiguracyjna. Na rys. 4 podano informacje o przyporządkowaniu symboli ze specyfikacji, tzn. a1, b1, a2,...,u1 do wejść i wyjść obiektowych sterownika - BI1, BI2,..., BO1. W tabl. 1, etapie konfiguracji DEFN zdefiniowano 11 bloków funkcyjnych B01-B11 przypisując im funkcje NOT, AND, OR lub DFF. W etapie CONN podano połączenia wejść zdefiniowanych bloków. Ostatni etap POSN określa kolejność obliczeń (pozycję bloku).

Tabl. 1. Lista konfiguracyjna PSW
Tab. 1. PSW configuration list

```
DEFN
        B01     NOT
        B02     AND
        B03     OR
        B04     NOT
        B05     AND
        B06     OR
        B07     DFF
        B08     DFF
        B09     NOT
        B10     OR
        B11     AND
CONN
        B01.1   BI2
        B02.1   B01.b
        B02.2   B07.b
        B03.1   B02.b
        B03.2   BI1
        B04.1   BI4
        B05.1   B04.b
        B05.2   B08.b
        B06.1   B05.b
        B06.2   BI3
        B07.1   B03.b
        B08.1   B06.b
        B09.1   BI5
        B10.1   B07.b
        B10.2   B08.b
        B11.1   B09.b
        B11.2   B10.b
        BO1     B11.b
POSN
        P1      B01
        P2      B02
        P3      B03
        P4      B04
        P5      B05
        P6      B06
        P7      B07
        P8      B08
        P9      B09
        P10     B10
        P11     B11
```

5. Podsumowanie

Przedstawiono metodę automatycznej generacji konfiguracji sekwencyjnego układu sterowania metodą bloków funkcyjnych FBD. Specyfikacja jest podana w formie automatu Mealy'ego. Pokazano sposób realizacji automatu w sterowniku. Omówiono przyjęty sposób zapisu symboli stanów, funkcji przejść i wyjść. Podano krótki opis narzędzi służących do wykonania analizy specyfikacji i przekształcenia jej na konfigurację. Otrzymany program jest względnie krótki i nie ma ograniczeń na rozmiar realizowanej konfiguracji. Jako przykład posłużył układ sterowania zaporami na przejeździe kolejowym. Zrealizowano go w przemysłowym programowalnym sterowniku wielofunkcyjnym PSW-166. Podano zarówno schemat FBD jak i listę konfiguracyjną otrzymanego automatycznie rozwiązania. Jako narzędzie zastosowano język funkcyjny ML.

Literatura:

[1] R. Lewis: Programming Industrial Control Systems Using IEC 1131. IEE Contr. Eng. Series 50, London 1995.
[2] PN-EN 61131-3: Sterowniki programowalne - Języki programowania. Wyd. Normalizacyjne ALFA-WERO, Warszawa 1998.
[3] L. C. Paulson: ML for the Working Programmer 2nd ed. Cambridge University Press, Cambridge 2000.
[4] P. D. Varhol: ML and Colored Petri Nets for Modeling and Simulation, Dr. Dobbs Journal Vol. 16, No 9, Sept. 1991.
[5] J. Cisek: Zarys metody tworzenia oprogramowania modelu sterownika w języku funkcjonalnym ML. VI Konferencja Systemy Czasu Rzeczywistego'99, Zakopane 1999, 222-229.
[6] J. Cisek: Weryfikacja poprawności oprogramowania układu sterowania złożonego z bloków funkcyjnych. Rozprawa doktorska, AGH Kraków 2004.
[7] S. Romanenko, P. Sestoft, Moscow ML Language Overview, Na stronie http://www.dina.dk/~sestoft/mosml.html (format pdf).
[8] C. Zieliński: Podstawy projektowania układów cyfrowych, PWN, Warszawa 2003.
[9] L. Trybus: Sterowniki wielofunkcyjne, WNT, Warszawa 1992.
[10] B. Trybus, Z. Świder, L. Trybus: Integrated environment for graphical configuration of programmable controllers. 4th Inst. Sci. Colloq. CAX TECHNIQUES, Bielefeld 1999, 475-482.

Title: Automatic configuration of sequential control system

Artykuł recenzowany