Diagramy FBD w środowisku programistycznym CPDev

Bartosz Trybus
Agnieszka Ziętek

W artykule omówiono rozszerzenie środowiska programistyczno-uruchomieniowego CPDev umożliwiające tworzenie aplikacji sterujących za pomocą schematu bloków funkcji FBD. Edytor graficzny służy do tworzenia diagramów, zaś odpowiedni konwerter przekształca je do kodu w języku ST, który następnie jest kompilowany do postaci wykonywalnej. Do zapisu danych opisujących diagram FBD oraz bibliotekę bloków zastosowano format XML, w tym dokumenty o strukturze zgodnej ze standardem PLCOpen. Dzięki opisanemu rozszerzeniu środowisko CPDev pozwala obecnie na tworzenie aplikacji sterujących za pomocą trzech języków wg PN-EN 61131-3, tj. ST, IL i FBD.

Środowisko programistyczno-uruchomieniowe CPDev (*Control Program Developer*) [1] opracowane w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej umożliwia programowanie systemów sterowania zgodnie z PN-EN 61131-3:2004 (idt. IEC 61131-3) [1]. Głównym celem normy jest poprawa jakości oprogramowania sterującego poprzez zdefiniowanie języków programowania i procedur implementacyjnych. Uwolnienie projektantów od konieczności używania języków ogólnego przeznaczenia oraz ujednolicenie i usystematyzowanie procesu tworzenia i konfiguracji oprogramowania wpływa na jego poprawną implementację w sterownikach.

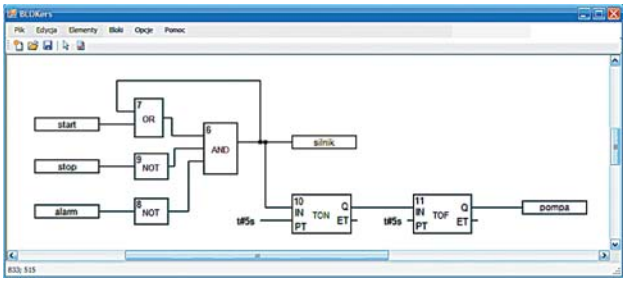
W środowisku CPDev programy kompilowane są do uniwersalnej postaci, niezależnej od docelowej platformy sprzętowej. Programy są wykonywane przez maszynę wirtualną, która może być implementowana na różnych platformach [5]. Pierwszym zastosowaniem środowiska CPDev jest oprogramowanie sterownika SMC firmy Lumel, będącego podstawą niewielkich rozproszonych systemów kontrolno-pomiarowych [3].

Do tej pory programiści korzystający ze środowiska CPDev mogli stosować dwa tekstowe języki programowania, tzn. tekst strukturalny ST (podstawowy język środowiska) oraz lista instrukcji IL [4]. W artykule opisana jest implementacja języka graficznego FBD (*Function Block Diagram*) w środowisku CPDev. FBD, czyli schemat bloków funkcji, jest odpowiednikiem schematu przepływu sygnałów dla obwodów logicznych przedstawianych w formie połączonych bramek logicznych oraz funkcji i bloków funkcji [9]. FBD pozwala na budowanie programów przy użyciu gotowych bloków lub funkcji. Polega to na wyborze odpowiednich bloków z biblioteki oraz umieszczenie ich w przestrzeni roboczej edytora graficznego, a następnie ustaleniu zależności poprzez ich połączenie. Podstawowymi zaletami języka FBD decydującymi o jego popularności są intuicyjność, łatwość tworzenia programów i czytelność diagramów.

dr inż. Bartosz Trybus – adiunkt w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej
mgr inż. Agnieszka Ziętek – absolwentka Politechniki Rzeszowskiej (2008), programistka

Edytor diagramów FBD

Edytor FBD dla środowiska CPDev (rys. 1) umożliwia tworzenie diagramów na komputerze pracującym pod kontrolą systemu MS Windows [10]. Udostępnia typowe funkcje edycyjne, tj. wstawianie bloków do diagramu i rysowanie połączeń między nimi, usuwanie oraz zaznaczanie obiektów, skalowanie diagramu. Oprócz bloków funkcyjnych obiektami na diagramie mogą być także zmienne wejściowe i wyjściowe oraz wartości stałe.



Rys. 1. Diagram FBD algorytmu START-STOP sterującego silnikiem i pompą

Ważnym elementem edytora jest wbudowany mechanizm kontroli składniowej sprawdzający poprawność diagramu (na żądanie lub przy zapisywaniu). Zgłaszane błędy dotyczą m.in. niezgodności typów połączonych zmiennych, braku połączenia lub jego niedokończenia.

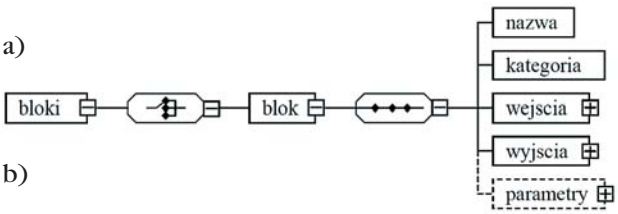
Diagram FBD dla programu START-STOP przedstawiono na rys. 1. Zmienna wyjściowa *silnik* (logiczna, typu BOOL) jest ustawiana, gdy aktywne jest wejście *start* a nieaktywne wejścia *stop* lub *alarm*. Po włączeniu silnika, po 5 s włączana jest pompa. Gdy jedna ze zmiennych stop lub alarm zostanie ustawiona, silnik jest natychmiast wyłączany, a pompa dopiero po 5 s. W przykładzie użyto funkcji logicznych AND, OR, NOT oraz bloków funkcyjnych TON i TOF. Bloki te należą

do standardowych bloków zdefiniowanych w PN-EN 61131-3:2004 i reprezentują czasomierze (*timer on/off delay*). Tutaj służą do opóźnionego włączania i wyłączania pompy, dlatego na wejście opóźniające (PT) podawana jest stała wartość 5 s.

Biblioteka bloków i funkcji

W PN-EN 61131-3:2004 zdefiniowano pewną liczbę funkcji i bloków standardowych. Są one wspólne dla wszystkich języków programowania (IL, ST, FBD) i udostępniane w środowisku CPDev. Bibliotekę bloków funkcyjnych zaimplementowano w edytorze FBD (rys. 2a) w formie pliku formatu XML (*eXtensible Markup Language*). Każdy z bloków ma nazwę i należy do określonej kategorii. Ma również określoną liczbę wejść i wyjść. Blok TON, który należy do kategorii *dwustanowe* (rys. 2b), ma wejścia IN typu BOOL, PT typu TIME oraz wyjścia Q (BOOL) i ET (TIME).

a)

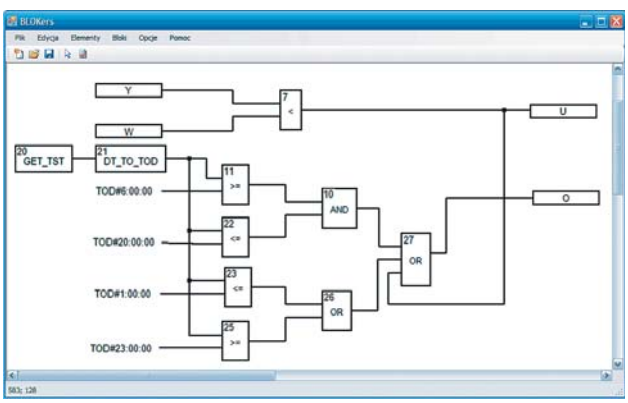


b)

```
<blok >
    <nazwa>TON</nazwa>
    <kategoria>dwustanowe</kategoria>
    <wejscia>
        <we name="IN">
            <type>
                <BOOL />
            </type>
        </we>
        <we name="PT">
            <type>
                <TIME />
            </type>
        </we>
    </wejscia>
    <wyjscia>
        <wy name="Q">
            <type>
                <BOOL />
            </type>
        </wy>
        <wy name="ET">
            <type>
                <TIME />
            </type>
        </wy>
    </wyjscia>
    <parametry />
</blok>
```

Rys. 2. a) Struktura biblioteki bloków edytora FBD; b) reprezentacja bloku TON w bibliotece

Oprócz funkcji i bloków wymaganych wg PN-EN 61131-3:2004, w bibliotece edytora FBD zawarto także funkcje specyficzne dla środowiska CPDev. Są one związane z obsługą sprzętu, dostępem do zmiennych systemowych, konwersją typów itp. Przykładem może być funkcja systemowa GET_TST zwracająca stan zegara czasu rzeczywistego (data i czas). Za pomocą funkcji DT_TO_TOD dokonywana jest konwersja typu (*obcięcie*) do godziny, minuty i sekundy. Zaprezentowany algorytm (rys. 3) steruje piecem c.o. i pompą obiegową. Jeżeli temperatura zadana jest większa niż zmierzona (W>Y), piec jest włączany sygnałem U, a jeżeli W<Y, piec jest wyłączany. Pompa obiegowa O jest włączona stale w ciągu dnia (od 6:00 do 20:00), a w nocy tylko wtedy, gdy piec pracuje oraz między godzinami 23:00 a 1:00.



Rys. 3. Diagram FBD algorytmu sterującego piecem c.o. i pompą w zależności od pory dnia

Konwersja FBD do postaci tekstowej

Edytor FBD zapisuje utworzone diagramy również jako pliki w formacie XML. W tym przypadku struktura dokumentu bazuje na założeniach standardu PLCOpen [7]. W standardzie tym zaproponowano formaty dokumentów XML dla różnych języków wg PN-EN 61131-3, w tym FBD.

Edytor FBD tworzy dokumenty o nieco uproszczonej w stosunku do PLCOpen strukturze, która jest jednak wystarczająca dla potrzeb środowiska CPDev. Na rys. 4 przedstawiono fragment pliku XML dla diagramu START-STOP z rys. 1. Blok AND o identyfikatorze 6 ma trzy wejścia (*inputVariables*) typu BOOL, które łączą się (*connection*) z blokami o identyfikatorach odpowiednio 7, 9 i 8 (*refLocalId*) oraz jedno wyjście (*outputVariables*) typu BOOL.

Kluczowym modułem środowiska CPDev jest kompilator języka ST, dlatego zde-

```
<block typeName="AND" localId="6">
  <inputVariables>
    <variable name="IN1">
      <type>
        <BOOL />
      </type>
      <connection refLocalId="7"/>
    </variable>
    <variable name="IN2">
      <type>
        <BOOL />
      </type>
      <connection refLocalId="9"/>
    </variable>
    <variable name="IN3">
        <type>
          <BOOL />
        </type>
        <connection refLocalId="8"/>
      </variable>
    </inputVariables>
    <outputVariables>
      <variable name="OUT1">
        <type>
          <BOOL />
        </type>
      </variable>
    </outputVariables>
    <parameters />
  </block>
```

Rys. 4. Fragment dokumentu XML opisującego diagram z rys. 1

cydowano się na opracowanie translatora FBD2CPDev, który konwertuje plik XML opisujący diagram FBD na odpowiadający mu kod w języku ST. Dokonywane jest to według następujących reguł:

- diagram FBD jest konwertowany do jednostki organizacyjnej oprogramowania (*Program Organization Unit* – POU) w języku ST
- wejściowe i wyjściowe zmienne diagramu są reprezentowane jako zmienne globalne
- dla każdej instancji bloku tworzona jest odpowiednia zmienna
- połączenia pomiędzy wyjściami i wyjściami bloków reprezentowane są poprzez zmienne.

```
(*$LIBRARY start-stop*)
(* GLOBAL VARIABLES *)
VAR_GLOBAL
    start  : BOOL;
    stop   : BOOL;
    alarm  : BOOL;
    silnik : BOOL;
    pompa  : BOOL;
END_VAR

(* USER POUs *)
PROGRAM start-stop

VAR
    var_OR7_0  : BOOL;
    var_NOT9_0 : BOOL;
    var_NOT8_0 : BOOL;
    var_AND6_0 : BOOL;
    TON10 : TON;
    TOF11 : TOF;
END_VAR

VAR_EXTERNAL
    start  : BOOL    (*$READ*);
    stop   : BOOL    (*$READ*);
    alarm  : BOOL    (*$READ*);
    silnik : BOOL    (*$WRITE*);
    pompa  : BOOL    (*$WRITE*);
END_VAR

    var_AND6_0 := AND(var_OR7_0,var_NOT9_0,var_NOT8_0);
    var_OR7_0  := OR(var_AND6_0,start);
    var_NOT8_0 := NOT(alarm);
    var_NOT9_0 := NOT(stop);
    TON10(IN := var_AND6_0, PT := t#5s);
    TOF11(IN := TON10.Q, PT := t#5s);
    silnik := var_AND6_0;
    pompa  := TOF11.Q;
END_PROGRAM
```

Rys. 5. Program w języku ST odpowiadający diagramowi z rys. 1.

Na rys. 5 pokazano wynik takiej konwersji dla diagramu z rys. 1. Zmienne *start*, *stop*, *alarm*, *silnik* i *pompa* są zmiennymi globalnymi (*VAR_GLOBAL*). Pozostałe zmienne w deklaracjach *VAR...END_VAR* zostały automatycznie utworzone podczas konwersji i reprezentują bloki funkcyjne (*TON10* i *TOF11*) oraz połączenia między nimi (np. *var_OR7_0* odpowiada linii wychodzącej z funkcji OR). Algorytm programu (zapisany jest po prawej stronie) składa się z linii przypisujących odpowiednim zmiennym wyniki działania funkcji logicznych AND, OR i NOT, wywołujących bloki funkcjonalne TON i TOF oraz uaktualniających zmienne wyjściowe *silnik* i *pompa*.

Podsumowanie

Język tekstowy ST jest zaawansowanym narzędziem umożliwiającym tworzenie skomplikowanych algorytmów sterowania. Jest on jednak zbliżony do typowych języków programowania, co może zniechęcać praktyków, którzy nie są programistami. Znajomość ST, w porównaniu z językami graficznymi, deklaruje stosunkowo niewielki procent respondentów fachowych pism [8].

Naturalnym staje się więc uzupełnienie środowiska programistycznego CPDev o możliwość tworzenia programów sterujących w języku FBD. Wykorzystano tu otwarty standard PLCOpen, co umożliwia współpracę z innymi, zewnętrznymi aplikacjami za pośrednictwem dokumentów XML. Wewnętrznie diagram FBD jest przekształcany do postaci tekstowej (ST), a następnie kompilowany i ładowany do platformy docelowej, gdzie maszyna wirtualna rozpoczyna jego wykonywanie (rys. 6). Dzięki temu rozszerzeniu pakiet CPDev może być obecnie stosowany do tworzenia i uruchamiania aplikacji sterujących w trzech językach: ST, IL i FBD, przy czym wykonywane zadanie (*control task*) może być złożone z kilku programów zapisanych w dowolnym z tych języków.

Rys. 6. Etapy tworzenia aplikacji w języku FBD w środowisku CPDev

Bibliografia

1. PN-EN 61131-3:2004, *Sterowniki programowalne*. Część 3: *Języki programowania* (idt. IEC 61131-3: 2003 Standard: Programmable Controllers Part 3, Programming Languages).
2. Rzońca D., Sadolewski J., Trybus B.: *Prototype environment for controller programming in the IEC 61131-3 ST language*. Computer Science and Information Systems, December 2007.
3. Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: *Mini-DCS system programming in IEC 61131-3 Structured Text*. Journal of Automation, Mobile Robotics & Intelligent Systems, Vol. 2, No.3, 2008.
4. Sadolewski J., Szmyd E.: *Polski kompilator oraz sterowniki programowane za pomocą języka IL normy IEC 61131-3*. XIII Konferencja Automation 2009. Pomiary Automatyka Robotyka. 2009, nr 2, (CD) s. 615-622.
5. Sadolewski J., Trybus B.: *Wieloplatformowa maszyna wirtualna dla systemów sterowania.* [w:] *Modele i zastosowania systemów czasu rzeczywistego*. (red. Z. Mazur, Z. Huzar), WKŁ, Warszawa 2008, s. 293–302.
6. Microsoft .NET Framework Developer's Guide – http://msdn2.microsoft.com/en-US/library/aa720433.aspx
7. XML Formats for IEC 61131-3 ver. 1.01 – Official Release – www.plcopen.org
8. Pietrusiewicz K.: *PLC: opinie, oczekiwania, prognozy*. Control Engineering Polska, http://www.designnews.pl/raport_200709.php4?num=575
9. Kasprzyk J.: *Programowanie sterowników przemysłowych*. WNT 2006.
10. Ziętek A.: *Graficzny interfejs w technologii .NET do tworzenia schematów FBD*. Praca magisterska, Wydział Elektrotechniki i Informatyki Politechniki Rzeszowskiej, 2008. ■