Edytory graficzne języków LD i FBD w pakiecie CPDev

Zbigniew Świder
Politechnika Rzeszowska, Katedra Informatyki i Automatyki, ul. W. Pola 2, 35-959 Rzeszów

Streszczenie: W artykule opisano edytory graficzne języków LD i FBD wykorzystywane w autorskim środowisku inżynierskim CPDev. Schematy stworzone za pomocą tych edytorów są wstępnie konwertowane do kodu źródłowego w ST, a następnie tłumaczone do autorskiego kodu pośredniego. Unikalną cechą edytorów jest mechanizm automatycznego wyznaczania połączeń między elementami schematu, wykorzystujący algorytm A*. Poszczególne elementy schematu nie mają sztywno określonego położenia w polu roboczym, co wpływa na przejrzystość schematu. W trybie śledzenia on-line, bezpośrednio na schemacie graficznym wyświetlane są wartości wyjść poszczególnych elementów. W tym celu schemat graficzny jest konwertowany do postaci grafu AOV, a następnie upraszczany i bezpośrednio z niego generowany jest kod źródłowy w języku ST. Opracowane edytory graficzne umożliwiają tworzenie zarówno prostych jak i złożonych programów sterowania, jak również zapewniają łatwą edycję, klonowanie fragmentów schematu, zapis, odczyt oraz drukowanie stworzonego diagramu.

Słowa kluczowe: środowisko inżynierskie, IEC 61131-3, maszyna wirtualna, translatory języków LD i FBD, edytory graficzne, grafy AOV

1. Wprowadzenie

Środowiska inżynierskie implementujące języki normy IEC 61131-3 [1] (przyjętej w Polsce jako PN-EN 61131-3) są powszechnie stosowane w przemyśle do tworzenia programów sterowania. Standard ten definiuje pięć języków programowania, w tym dwa tekstowe IL, ST, dwa graficzne LD, FBD oraz mieszany język SFC. Języki IL, LD i SFC są przeznaczone głównie do zastosowań w procesach produkcyjnych opartych na sterownikach PLC, natomiast języki ST, FBD są stosowane w obszarach wykorzystujących sterowniki PAC.

Prace nad środowiskiem inżynierskim zgodnym z normą IEC 61131-3 rozpoczęły się już ponad 10 lat temu [2]. Opracowane wtedy środowisko nazwane zostało CPDev (ang. *Control Program Developer*) i kompilowało jedynie programy z języka ST do autorskiego kodu VMASM (ang. *Virtual Machine Assembler*) interpretowanego następnie przez stworzoną maszynę wirtualną, zaimplementowaną początkowo na ośmiobitowych mikrokontrolerach MCS 51 i AVR. Od początku środowisko wspierało możliwość samodzielnej implementacji maszyny wirtualnej na docelowym sprzęcie, zarówno przez inżynierów w przemyśle, jak i studentów. W kolejnych latach do pakietu dodano rozszerzenia dla 32- i 64-bitowych procesorów, uzupełniono je o translatory języków graficznych LD, FBD, SFC, narzędzie do projektowania interfejsów HMI i wieloprojektowe środowisko uruchomieniowe. Dwa kolejne rozszerzenia wsparły badania nad podejściem MDD (ang. *Model Driven Development*) [3] i testami jednostkowymi [4]. Do tej pory środowisko CPDev zostało wdrożone przez kilku znaczących producentów urządzeń automatyki w kraju i za granicą.

Translatory języków graficznych (LD, FBD) na język ST, wbudowane w pakiety inżynierskie, często bazują na grafach AOV (ang. *Activity on Vertex*) [5, 6]. W pakiecie CPDev moduł translatora również korzysta z grafu AOV, z którego to bezpośrednio (po optymalizacji) generowany jest program w języku ST. Szczegóły zostaną przedstawione w rozdziale 5.

W artykule zaprezentowano dwa elementy środowiska inżynierskiego CPDev – edytory graficzne języków LD i FBD, zgodnych z normą IEC 61131-3. W kolejnym rozdziale przedstawiono budowę środowiska, główne okno programu oraz interfejs użytkownika. W rozdziale 3 opisano możliwości edytorów LD i FBD, a w rozdziale 4 translator schematów graficznych na kod źródłowy w języku ST (wraz z przykładami konwersji). W podsumowaniu przedstawiono wybrane wdrożenia przemysłowe pakietu CPDev.

Autor korespondujący:
Zbigniew Świder, swiderzb@prz.edu.pl

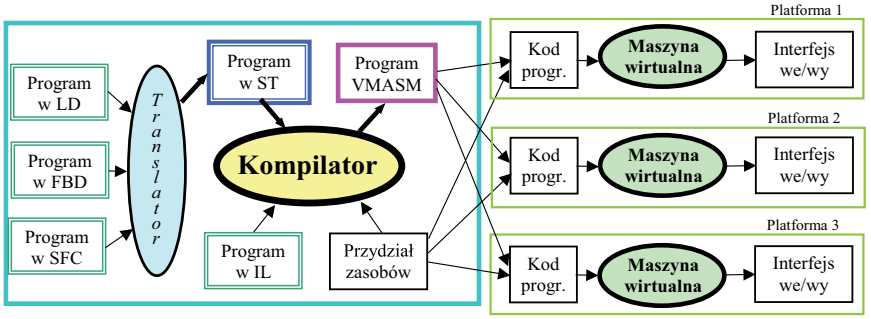
Artykuł recenzowany
nadesłany 12.09.2019 r., przyjęty do druku 05.12.2019 r.

2. Środowisko inżynierskie CPDev

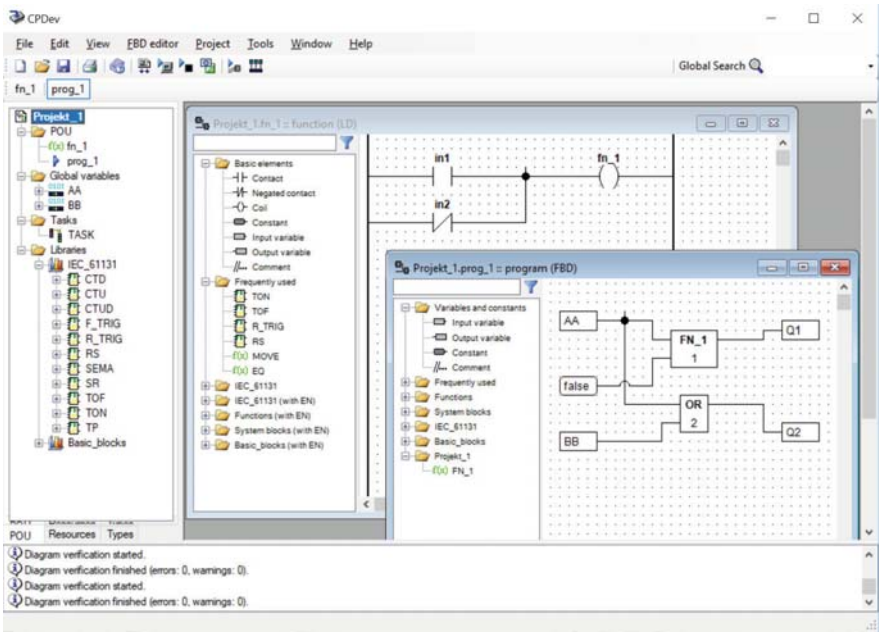
Środowisko inżynierskie, takie jak przedstawiany tu pakiet CPDev, można uznać za: (1) *uniwersalne*, jeśli umożliwia gene-

rowanie kodu wykonywalnego na różne platformy sprzętowe, (2) *otwarte* pod względem programowym i sprzętowym, jeżeli dostępne są dla użytkownika końcowego narzędzia do tworzenia własnych bloków funkcjonalnych oraz specyfikacje interfejsów wejścia/wyjścia, (3) *elastyczne*, jeżeli spośród wszystkich typów danych zdefiniowanych w normie IEC można swobodnie wybrać te, które najlepiej odpowiadają danej aplikacji.

Pierwsza wersja CPDev, wspierająca początkowo jedynie język ST, została w kolejnych latach rozszerzona o obsługę pozostałych języków normy IEC 61131-3, graficzne projektowanie interfejsów HMI, automatyczny generator dokumentacji oraz dodatkowe moduły o zastosowaniu badawczym. Na rysunku 1 przedstawiono kolejne etapy przetwarzania programów sterowania napisanych w różnych językach normy. Języki tekstowe (ST, IL) są tu bezpośrednio kompilowane do mnemoników opracowanego na potrzeby środowiska specjalizowanego języka VMASM. Diagramy graficzne LD, FBD i SFC są najpierw konwertowane na język ST, a następnie kompilowane do VMASM. Język ST jest więc bazowym językiem środowiska CPDev. Wykonywalny kod binarny programu jest generowany z mnemoników VMASM i ładowany do maszyny wirtualnej. Źródła maszyny wirtualnej napisano w całości w języku C, co znacząco ułatwia przenośność na różne platformy sprzętowe.



Rys. 1. Etapy tworzenia oprogramowania dla różnych platform sprzętowych
Fig. 1. Stages of software development for various hardware platforms



Rys. 2. Główne okno programu i interfejs użytkownika w systemie CPDev
Fig. 2. The main program window and the user interface in the CPDev system

Podstawowe funkcjonalności CPDev są podobne do innych tego typu środowisk inżynierskich [7, 8]. Nowością jest autorski mechanizm automatycznego wyznaczania połączeń między elementami graficznymi na schematach FBD i LD [9] bazujący na algorytmie A* [10]. Również tworzony przez użytkownika schemat graficzny jest konwertowany do postaci grafu AOV [5], który następnie jest upraszczany (eliminowane są nadmiarowe węzły) i bezpośrednio z niego generowany jest program w języku ST.

Kompilator wykorzystuje zbiór reguł języka ST, biblioteki bloków funkcjonalnych oraz wykaz instrukcji elementarnych realizowanych przez maszynę wirtualną. Oprócz kodu wykonywalnego tworzone są informacje służące do śledzenia, uruchamiania i symulacji programu. Konfiguracja zasobów sprzętowych dotyczy pamięci, interfejsu wejścia/wyjścia oraz interfejsu komunikacyjnego, a specyfikacja obejmuje typy i obszary pamięci, liczby oraz typy wejść i wyjść oraz kanały komunikacyjne.

Podstawowe składniki maszyny wirtualnej to: pamięci programu i danych, osobne stosy programu i danych, moduł wykonywania instrukcji oraz interfejs dla docelowej platformy. Maszyna wirtualna działa jako interpreter wykonując kolejne linie kodu programu. Interfejs dla docelowej platformy składa się z niskopoziomowych funkcji zależnych od sprzętu i systemu operacyjnego, jak aktualizacja wejść i wyjść czy też zarządzanie czasem cyklu.

Poza maszyną wirtualną wykonującą główne zadania sterowania, środowisko uruchomieniowe może zawierać również moduły HMI obsługujące poszczególne interfejsy graficzne do komunikacji z operatorem. Maszyna wirtualna i moduł HMI działają wtedy równolegle i niezależnie, wymieniając kanałem komunikacyjnym potrzebne dane za pomocą zmiennych globalnych.

Na rysunku 2 pokazano główne okno interfejsu CPDev, zawierające:
- (po lewej stronie) drzewa struktury projektu, zmiennych globalnych, wykonywanych zadań, zastosowanych bibliotek i dostępnych zasobów programowych,
- (na górze) menu główne i paski narzędziowe (ikony do szybkiego wyboru),
- (na dole) okno komunikatów, raportów oraz informacji statusowych,
- (centralnie) obszar roboczy z oknami zawierającymi programy użytkownika w językach normy IEC 61131-3 (ST, IL, LD, FBD, SFC).

W obszarze roboczym (rys. 2) otwarte są dwa okna: jedno programu (*prog_1*) w języku FBD oraz drugie funkcji użytkownika (*fn_1*) w języku LD – funkcja ta użyta jest w programie *prog_1*. Interfejsy obu edytorów graficznych (FBD i LD) są identyczne, co znacząco ułatwia ich obsługę przez użytkownika końcowego tworzącego programy sterowania. W programie są to jednak dwie osobne klasy *FBDEditor* oraz *LDEditor* dziedziczące od wspólnej klasy bazowej *GraphicEditor*.

Poszczególne programy lub funkcje (napisane w dowolnych językach normy) mogą także wymieniać dane przez zmienne globalne. Do dyspozycji użytkownika są też gotowe biblioteki funkcji i bloków funkcjonalnych zgodnych z normą IEC 61131-3, zarówno z wejściem EN jak i bez niego. Na bieżąco dołączane są także własne biblioteki.

3. Edytory graficzne języków FBD i LD

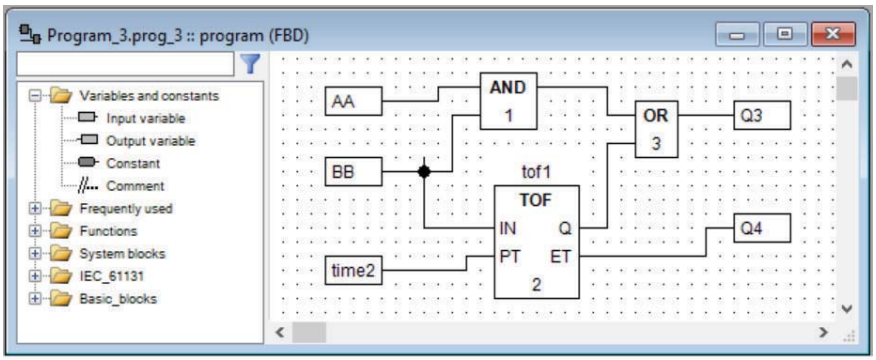
Wspólną cechą zaimplementowanych w pakiecie CPDev edytorów diagramów FBD i LD jest mechanizm automatycznego wyznaczania połączeń między elementami za pomocą zmodyfikowanego algorytmu A* [10], łączącego podejście konwencjonalne z heurystycznym. Modyfikacja ta polega na dodaniu współczynników wagowych [9], których odpowiedni dobór wpływa znacząco na liczbę przecięć z innymi połączeniami, jak też na liczbę zmian kierunków (w górę, w dół, w lewo, w prawo), aby możliwie zmniejszyć długość połączeń. Dzięki temu elementy schematu, takie jak bloki, funkcje, styki czy cewki, nie mają ściśle przywiązanego położenia w polu roboczym (np. poziomo w drabince LD) i mogą być rozmieszczone dowolnie w polu roboczym. Wpływa to znacząco na przejrzystość schematu i pozwala w łatwy sposób go modyfikować i dodawać nowe elementy w razie potrzeby.

Przykładowy program (diagram) w języku FBD jest pokazany na rys. 3. Użyto tu instancji bloku bibliotecznego (*TOF*) oraz bramek *AND* i *OR*. Zwróćmy uwagę, że w odróżnieniu od innych tego typu pakietów inżynierskich (np. [7, 8]), położenie elementów schematu (bloków, funkcji itp.) jest tu dowolne, a kolejność ich przetwarzania jest zdefiniowana przez użytkownika (określa to liczba w dolnej części prostokąta, np. bramka AND będzie przetwarzana jako pierwsza). Oczywiście użytkownik może w każdej chwili zmienić tę kolejność, wstawić lub usunąć jakiś element, czy też sklonować fragment schematu (kolejność elementów zostanie wtedy automatycznie przenumerowana).

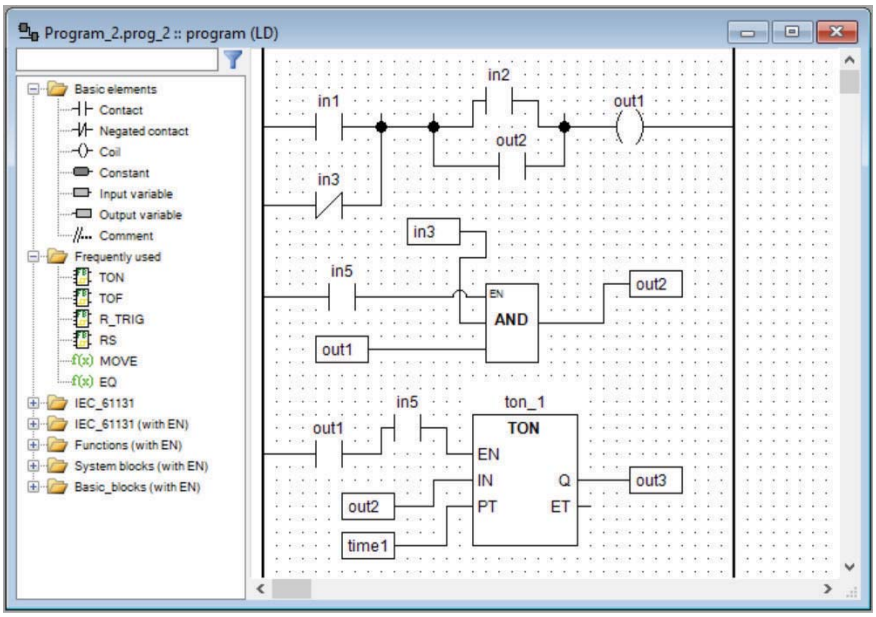
Tak utworzone schematy są przekształcane do programów źródłowych w języku ST, gdzie mogą być dowolnie modyfikowane. Podobnie bloki funkcjonalne lub funkcje, napisane przez użytkownika w innych językach normy, są automatycznie obrazowane w edytorze graficznym i mogą być tam użyte do tworzenia kolejnych schematów.

Przykładowy schemat w języku drabinkowym (LD) pokazano na rys. 4. Składa się on z trzech szczebli, gdzie w pierwszym zastosowano zestyki a na wyjściu cewkę (*out1*). Tutaj również, w zależności od preferencji użytkownika, można trzymać się „klasycznego" sposobu tworzenia schematu LD (elementy w jednej linii) lub zastosować format „swobodny" (np. wysunięty w górę zestyk *in2*). W drugim szczeblu użyto funkcji bibliotecznej AND z wejściem EN oraz celowo wysuniętą w górę zmienną wejściową *in3*. W trzecim szczeblu zastosowano instancję bloku funkcjonalnego TON (także z wejściem EN).

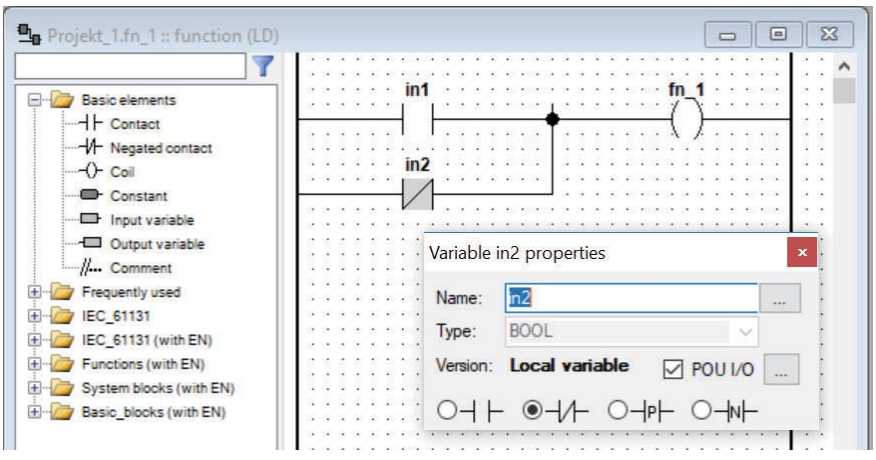
Użytkownik może również tworzyć własne funkcje i bloki funkcjonalne gromadząc je w swoich bibliotekach – można tu korzystać ze wszystkich dołączonych bibliotek i dostępnych języków (ST, IL, LD, FBD, SFC). Na rys. 5 pokazano przykład funkcji użytkownika (o nazwie *fn_1*) opracowanej w języku LD. Funkcja ma dwa wejścia (*in1*, *in2*) oraz wyjście (o takiej samej nazwie jak nazwa funkcji). W oknie właściwości elementu można modyfikować jego nazwę, typ, działanie oraz określić, czy jest to wejście lub wyjście funkcji czy też zmienna lokalna (pomocnicza) lub globalna.



Rys. 3. Program w FBD utworzony w edytorze graficznym pakietu CPDev
Fig. 3. The program in FBD created in the graphic editor of the CPDev package



Rys. 4. Program w LD utworzony w edytorze graficznym pakietu CPDev
Fig. 4. The program in LD created in the graphic editor of the CPDev package



Rys. 5. Przykład tworzenia funkcji użytkownika w edytorze LD
Fig. 5. An example of creating a user function in the LD editor

4. Translator języków LD i FBD

Jednym z założeń przy tworzeniu pakietu CPDev było to, aby w trybie debugowania możliwe było wyświetlanie bezpośrednio na schemacie (diagramie) wartości wyjść wszystkich użytych elementów (styków, cewek, bloków funkcjonalnych i funkcji). Przedstawiono to na rys. 6. Z lewej strony okna, w miejscu drzewa projektu, prezentowane są wtedy wartości wybranych zmiennych (globalnych i lokalnych) oraz wejść i wyjść użytych bloków i funkcji. Dodatkowo na schemacie w oknie roboczym w trybie *on-line*, po najechaniu kursorem na połączenie, wyświetlane są aktualne wartości zmiennych (np. FALSE na wyjściu zestyku ALARM) oraz poszczególne połączenia między elementami schematu zmieniają kolor lub atrybut (np. TRUE – linia ciągła niebieska, FALSE – linia przerywana czarna).

Dlatego też stworzony przez użytkownika schemat w języku LD jest najpierw konwertowany do postaci grafu AOV, którego to wierzchołkami (*vertices*) są te właśnie elementy schematu (reprezentujące zmienne, zadania lub czynności), a krawędzie grafu (*edges*) definiują połączenia (relacje) między nimi (rys. 7). Tak stworzony graf AOV jest kolejno upraszczany (eliminowane są nadmiarowe węzły), a następnie bezpośrednio z niego generowany jest program w języku ST. W przeciwieństwie do translatorów opisywanych m.in. w pracach [5, 6], tutaj graf AOV nie jest przekształcany do drzewa binarnego, gdyż (zgodnie z pierwotnym założeniem) potrzebne są wartości na wyjściach każdego elementu.

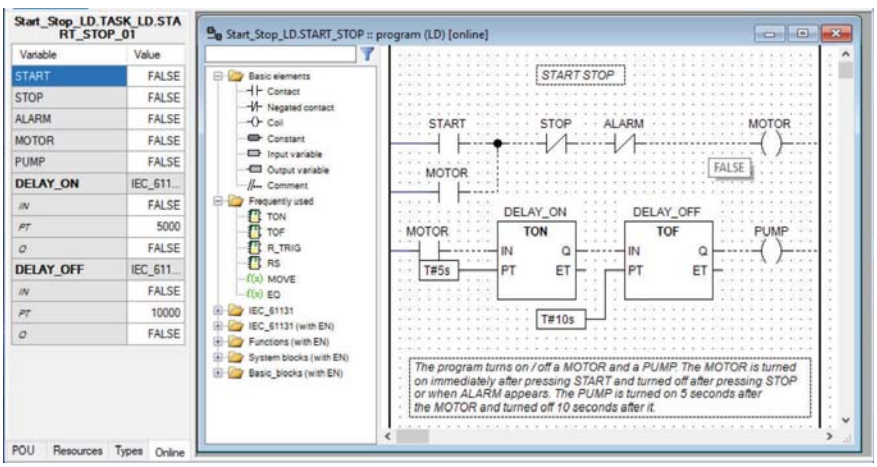
Można to zaobserwować we fragmencie programu LD (rys. 7), gdzie jeden wiersz diagramu tłumaczony jest na kilka linii programu w ST oraz wprowadzane są dodatkowe zmienne lokalne reprezentujące wyjścia elementów schematu (z prefiksem, np. *out_contact_* czy też *out_bp_*, oraz postfiksem, np. *_80_80* oznaczającym współrzędne elementu na schemacie). Zatem zmienna *out_contact_input1_80_80* oznacza wyjście elementu *input1* (zestyku) o współrzędnych (*80*, *80*) – na grafie AOV odpowiada jej węzeł wirtualny *vn1* (*virtual node*). Podobnie, wyjściu elementu *input3* odpowiada węzeł *vn2*.

Połączenie szeregowe węzła wirtualnego i rzeczywistego tłumaczone jest jako operacja AND (np. *vn4* := *vn3* AND *input2*), natomiast równoległe dwóch węzłów wirtualnych jako OR (np. *vn3* := *vn1* OR *vn2*). W podobny sposób konwertowane są złożone schematy LD.

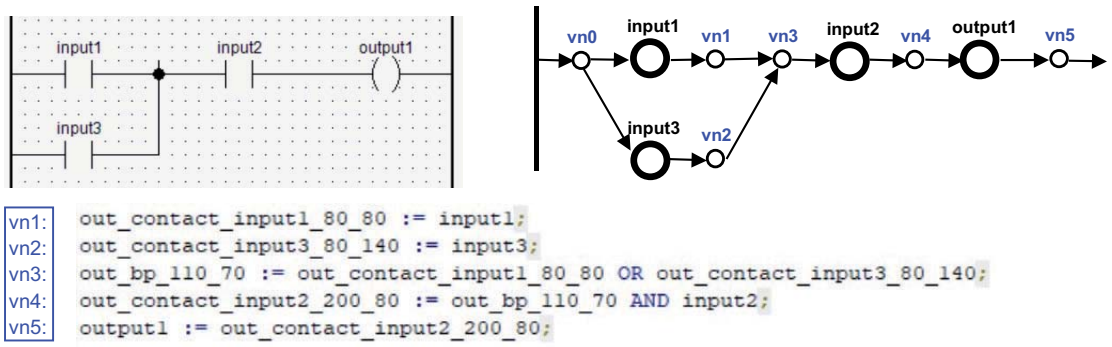
Ze względu na swobodny format diagramu LD (elementy szczebla nie muszą leżeć w jednej linii), kolejność przetwarzania elementów na schemacie wyznaczana jest na podstawie grafu AOV, a nie ich fizycznego położenia (współrzędnych) na diagramie. Dlatego też wysunięcie zestyków *in2* oraz *out2* na schemacie (rys. 8) nie ma żadnego wpływu na generowany kod (analizowany jest graf AOV, a nie położenie elementów na diagramie).

Funkcje i bloki funkcjonalne w języku LD mogą mieć dodatkowe wejście EN (*enable*). Norma IEC 61131-3 nie traktuje go jako zwykłego wejścia, dlatego też programy używające funkcji lub bloków z EN należy w sposób specjalny przekonwertować na ST (rys. 9).

Konwertowanie programów w FBD jest prostsze (rys. 10), gdyż kolejność przetwarzania elementów schematu jest definiowana przez użytkownika (w oknie właściwości).

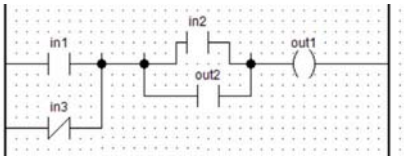


Rys. 6. Wygląd okienek pakietu CPDev w trybie debugowania on-line
Fig. 6. The appearance of CPDev package windows in on-line debug mode



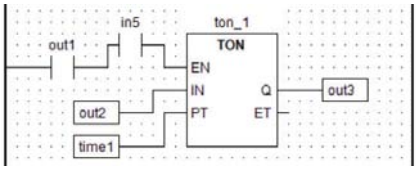

```
vn1:  out_contact_input1_80_80 := input1;
vn2:  out_contact_input3_80_140 := input3;
vn3:  out_bp_110_70 := out_contact_input1_80_80 OR out_contact_input3_80_140;
vn4:  out_contact_input2_200_80 := out_bp_110_70 AND input2;
vn5:  output1 := out_contact_input2_200_80;
```

Rys. 7. Program w LD, odpowiadający mu graf AOV oraz tłumaczenie na język ST
Fig. 7. The LD program, the corresponding AOV graph and translation into ST



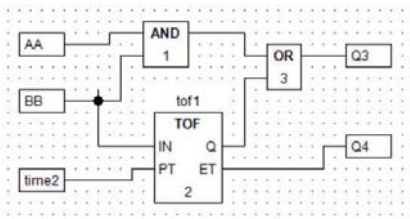

```
out_contact_in1_80_60 := in1;
out_contact_in3_80_120 := NOT in3;
out_bp_90_50 := out_contact_in1_80_60 OR out_contact_in3_80_120;
out_bp_130_50 := out_bp_90_50;
out_contact_in2_210_40 := out_bp_130_50 AND in2;
out_contact_out2_220_90 := out_bp_130_50 AND out2;
out_bp_230_50 := out_contact_in2_210_40 OR out_contact_out2_220_90;
out1 := out_bp_230_50;
```

Rys. 8. Przykład tłumaczenia szczebla drabinki w LD na język ST
Fig. 8. Example of ladder rung translation into ST language




```
out_contact_out1_80_310 := out1;
out_contact_in5_140_290 := out_contact_out1_80_310 AND in5;
IF out_contact_in5_140_290 THEN
ton_1(IN:=out2,PT:=time1,Q=>out_ton_1_Q);
out3 := out_ton_1_Q;
END_IF
```

Rys. 9. Program z blokiem z wejściem EN i jego tłumaczenie na język ST
Fig. 9. Program with a block with EN input and its translation into ST




```
(1) out_and1 := AND(AA, BB);
(2) tof1(IN:=BB,PT:=time2,Q=>out_tof1_Q,ET=>out_tof1_ET);
    Q4 := out_tof1_ET;
(3) out_or1 := OR(out_and1, out_tof1_Q);
    Q3 := out_or1;
```

Rys. 10. Program w języku FBD i jego tłumaczenie na język ST
Fig. 10. The program in FBD and its translation into ST

6. Podsumowanie

W artykule zaprezentowano edytory języków LD i FBD stosowane w środowisku inżynierskim CPDev. Programy w językach graficznych normy IEC 61131-3, jak LD, FBD i SFC, są wstępnie konwertowane do kodu źródłowego w ST, natomiast programy w językach tekstowych ST i IL są już bezpośrednio tłumaczone do kodu pośredniego VMASM.

Unikalną cechą opracowanych edytorów graficznych diagramów FBD i LD jest mechanizm automatycznego wyznaczania połączeń między elementami schematu z wykorzystaniem zmodyfikowanego algorytmu A*. Dzięki temu poszczególne elementy schematu nie mają z góry narzuconego położenia w polu roboczym (np. w drabince LD), a więc mogą być rozmieszczone dowolnie, co znacząco wpływa na przejrzystość schematu i jednocześnie pozwala w łatwy sposób go modyfikować.

Przy projektowaniu pakietu CPDev przyjęto, że w trybie debugowania możliwe będzie wyświetlanie bezpośrednio na schemacie graficznym (diagramie) wartości wyjść jego poszczególnych elementów. Dlatego schemat w języku LD jest najpierw konwertowany do postaci grafu AOV, który to jest kolejno upraszczany (eliminowane są nadmiarowe węzły), a bezpośrednio z niego generowany jest kod źródłowy w języku ST (przykłady zamieszczono w rozdziale 4).

Sterowniki podsystemów automatyki okrętowej, należące do systemu Mega-Guard z firmy Praxis [11], są najistotniejszym wdrożeniem pakietu CPDev. Każdy moduł składa się z procesora sterującego, jednostek wejścia/wyjścia i panelu dotykowego TFT, komunikujących się przez CAN lub Ethernet. Natomiast wdrożenia firmy Nauka i Technika [12] bazują na sterowniku StTr-PLC, będącym składnikiem zintegrowanej platformy wykorzystującej moduły komunikacyjne GSM, Wi-Fi oraz ZigBee.

Podziękowania

Autor dziękuje współtwórcom środowiska CPDev, w szczególności Leszkowi Trybusowi, Bartoszowi Trybusowi, Janowi Sadolewskiemu, Dariuszowi Rzońca oraz Marcinowi Jamro, za wszelką pomoc przy tworzeniu i modyfikacjach opisywanych tu edytorów graficznych.

Bibliografia

1. IEC 61131-3 – Programmable controllers – Part 3: Programming languages, 2003, 2013.
2. 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 and Intelligent Systems", Vol. 2, No. 3, 2008, 48–54.
3. Jamro M., Rzońca D., Rząsa W., *Testing communication tasks in distributed control systems with SysML and Timed Colored Petri Nets model*, "Computers in Industry", Vol. 71, 2015, 77–87, DOI: 10.1016/j.compind.2015.03.007.
4. Jamro M., *POU-Oriented Unit Testing of IEC 61131-3 Control Software*, "IEEE Transactions on Industrial Informatics", Vol. 11, No. 5, 2015, 1119–1129, DOI: 10.1109/TII.2015.2469257.
5. Fen G., Ning W., *A Transformation Algorithm of Ladder Diagram into Instruction List Based on AOV Digraph and Binary Tree* [in:] TENCON – IEEE Region 10 Conference, 2006, DOI: 10.1109/TENCON.2006.343937.
6. Huang L., Liu W., Liu Z., *Algorithm of transformation from PLC ladder diagram to structured text* [in:] 9th International Conference on Electronic Measurement Instruments, 2009, 4-778–4-782, DOI: 10.1109/ICEMI.2009.5274701.
7. 3S-Smart Software Solutions GmbH, CODESYS, [www.codesys.com].

Graphic Editors of LD and FBD Languages in CPDev Package

Abstract: The article describes graphic editors of LD and FBD languages used in the CPDev proprietary engineering community. The diagrams created with the help of these editors are pre-converted to source code in ST, and then translated into the author's intermediate code. A unique feature of these editors is the mechanism of automatically determining connections between diagram elements using the A* algorithm. Individual elements of the diagram do not have a rigid position in the working field (e.g. LD ladder), which significantly affects the transparency of the diagram. In on-line tracking mode, the output values of its individual elements are displayed directly on the graphic scheme. To ensure this, the graphic schema is first converted to an AOV graph, and then simplified and the ST source code is directly generated from it. Developed graphic editors allow creating both simple and complex control programs, as well as ensure easy editing, cloning of fragments of the scheme, saving, loading and printing of the created diagram.

Keywords: engineering environment, IEC 61131-3, virtual machine, LD and FBD language translators, graphic editors, AOV graphs

dr hab. inż. Zbigniew Świder
swiderzb@prz.edu.pl
ORCID: 0000-0003-3504-5340

Absolwent Wydziału Elektrotechniki i Informatyki Politechniki Rzeszowskiej (1984). Na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej w Gliwicach uzyskał stopień doktora nauk technicznych (1992) oraz doktora habilitowanego nauk technicznych (2004). Od początku pracy zawodowej zajmuje się sterownikami mikroprocesorowymi, a ostatnio metodami samostrojenia i adaptacji nastaw regulatorów, komputerowymi systemami automatyki oraz środowiskami inżynierskimi do programowania przemysłowych układów sterowania.