ZESZYTY NAUKOWE POLITECHNIKI RZESZOWSKIEJ 289, Elektrotechnika 33
RUTJEE, z. 33, styczeń-grudzień 2013, s. 117-132

Marcin JAMRO[1]
Dariusz RZOŃCA[2]
Jan SADOLEWSKI[3]
Andrzej STEC[4]
Zbigniew ŚWIDER[5]
Bartosz TRYBUS[6]
Leszek TRYBUS[7]

ŚRODOWISKO INŻYNIERSKIE CONTROL PROGRAM DEVELOPER OBECNIE

W artykule przedstawiono przegląd obecnej funkcjonalności środowiska inżynierskiego CPDev (Control Program Developer) opracowanego w Katedrze Informatyki i Automatyki Politechniki Rzeszowskiej. Pakiet ten pozwala na programowanie sterowników PLC/PAC zgodnie z normą PN-EN 61131-3. Oparcie systemu na koncepcji dedykowanych maszyn wirtualnych będących interpreterami kodu wykonywalnego zwiększa przenośność i uniwersalność programów sterowania. W porównaniu do poprzednich wersji środowisko CPDev zostało uzupełnione o obsługę wszystkich języków normy (ST, IL, FBD, LD, SFC), projektowanie interfejsu HMI zintegrowane z tworzeniem oprogramowania sterującego, testy tablicowe i jednostkowe komponentów programowych jak również możliwość modelowania struktury i funkcji złożonych programów w formie diagramów SysML. Narzędzie do projektowania interfejsu HMI jest niezależne od platformy sprzętowej i pozwala na łączenie sterowania z wizualizacją wykorzystując języki normy PN-EN 61131-3. Testy tablicowe i jednostkowe pozwalają na zwiększenie jakości oprogramowania. Modele oparte o

[1] Marcin Jamro, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12 , 35-959 Rzeszów, tel. 17 8651685, mjamro@kia.prz.edu.pl

[2] Dariusz Rzońca, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12 , 35-959 Rzeszów, tel. 17 8651765, drzonca@prz-rzeszow.pl

[3] Jan Sadolewski, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12 , 35-959 Rzeszów, tel. 17 8651796, js@ prz-rzeszow.pl

[4] Andrzej Stec, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12 , 35-959 Rzeszów, tel. 17 8651793, astec@prz-rzeszow.pl

[5] Zbigniew Świder, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12 , 35-959 Rzeszów, tel. 17 865 1549, swiderzb@prz.edu.pl

[6] Bartosz Trybus, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12 , 35-959 Rzeszów, tel. 17 8651685, btrybus@ prz.edu.pl

[7] Autor do korespondencji: Leszek Trybus, Politechnika Rzeszowska, Katedra Informatyki i Automatyki, al. Powst. Warszawy 12 , 35-959 Rzeszów, tel. 17 8651225, ltrybus@kia.prz.edu.pl

diagramy SysML wspierają wczesne fazy projektowania programów sterowania. Nowy trzydziestodwubitowy kompilator CPDev pozwala na tworzenie większych programów. Poza procesorami ogólnego przeznaczenia (takimi jak np. AVR, ARM czy x86) skompilowane programy mogą być wykonywane także na układach FPGA. Obecne przemysłowe wdrożenia środowiska CPDev obejmują urządzenia z firm Lumel S.A. Zielona Góra (sterownik programowalny SMC), Praxis Automation Technology B.V. Leiderdorp Holandia (sterowniki systemu Mega-Guard Ship Automation and Navigation System) oraz Nauka i Technika Sp. z o.o. Zaczernie/Rzeszów (sterownik StTr-760-PLC). W artykule jako przykład wdrożenia przedstawiono krótką charakterystykę systemu Praxis Mega-Guard.

Słowa kluczowe: PLC, IEC 61131-3, FPGA, HMI, SysML.

1. Wprowadzenie

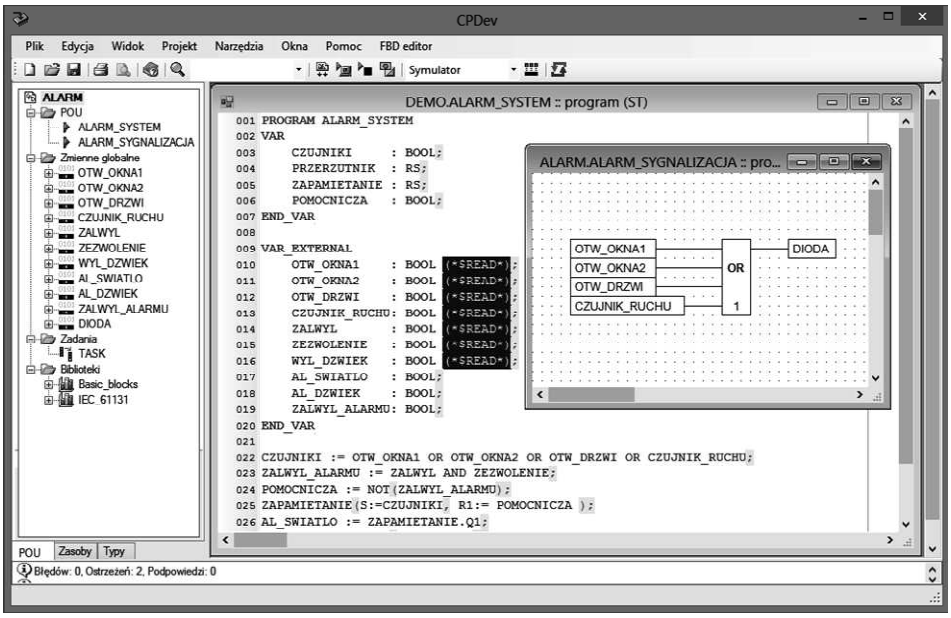
Prototyp środowiska inżynierskiego CPDev (*Control Program Developer*) przeznaczonego do programowania sterowników PLC i niewielkich rozproszonych systemów sterowania w języku ST normy PN-EN 61131-3 prezentowano już przed kilkoma laty [4, 13, 14]. Od tamtego czasu wprowadzono nowe funkcjonalności, takie jak: (1) obsługa pozostałych języków normy PN-EN 61131-3, tj. IL, FBD, LD i SFC, (2) projektowanie interfejsu HMI zintegrowane z tworzeniem oprogramowania sterującego [6], (3) narzędzie testujące dla komponentów programowych [7], (4) modelowanie problemu we wczesnej fazie projektowania za pomocą diagramów SysML [5]. Rozszerzono gamę maszyn wirtualnych. Poza maszynami dedykowanymi dla procesorów AVR, ARM i rodziny x86 opracowano także sprzętową maszynę implementowaną w układach FPGA [3]. CPDev jest obecnie wdrożony w trzech przedsiębiorstwach, dwóch polskich i holenderskim, pozwalając na programowanie sterowników poprzez łącza RS-485, Ethernet i bezprzewodowo.

W artykule przedstawiono przegląd bieżącego stanu prac nad środowiskiem CPDev, z naciskiem na nowe funkcjonalności. Dzięki nowemu trzydziestodwubitowemu kompilatorowi (poprzednio szesnastobitowy) można obsługiwać większe programy sterowania. Jak dotąd w literaturze nie opisywano także innych prototypów PLC opartych o FPGA. Narzędzie do projektowania interfejsu HMI jest niezależne od platformy sprzętowej i pozwala na łączenie sterowania z wizualizacją wykorzystując języki normy IEC. Testy tablicowe i jednostkowe stanowią zachętę dla projektanta do szczegółowej weryfikacji oprogramowania. Podobnie jak UML w aplikacjach IT, diagramy SysML wspierają we wczesnych fazach projektowanie programów sterowania.

2. Obecny stan prac nad środowiskiem CPDev

CPDev jest zintegrowanym środowiskiem inżynierskim do programowania sterowników PLC/PAC w językach normy PN-EN 61131-3. Środowisko wspiera wieloplatformowe aplikacje, tj. możliwość uruchamiania skompilowanych programów na różnych platformach sprzętowych, dotąd na AVR, ARM i x86, a od niedawna także na FPGA. Potrzeba opracowania takiego narzędzia została wskazana przez inżynierów praktyków kilka lat temu podczas jednej z Krajowych Konferencji Automatyków w Rytrze.

Obecnie CPDev obsługuje wszystkie języki normy PN-EN 61131-3, tj. tekstowe ST, IL, graficzne FBD, LD oraz mieszany SFC. Główne okno z przykładowymi krótkimi programami w ST i FBD pokazano na rys. 1.



Rys. 1. Główne okno środowiska CPDev z edytorami języków ST i FBD

Fig. 1. Main window of the CPDev environment with editors of ST and FBD languages

Użytkownik może opracowywać własne funkcje, bloki funkcyjne i programy, jak również umieszczać je w bibliotekach do ponownego wykorzystania.

Przemysłowe wdrożenia CPDev są następujące (przedsiębiorstwo, sterownik, procesor):

- Lumel – Zielona Góra, sterownik programowalny SMC, AVR ATmega 128 [8] ,
- Praxis Automation Technology – Leiderdorp Holandia, sterowniki systemu *Mega-Guard Ship Automation and Navigation System*, ARM7 LPC [11] (szerzej opisanego w p. 3),
- Nauka i Technika – Zaczernie/Rzeszów, sterownik StTr-760-PLC, ARM7 LPC [9] .

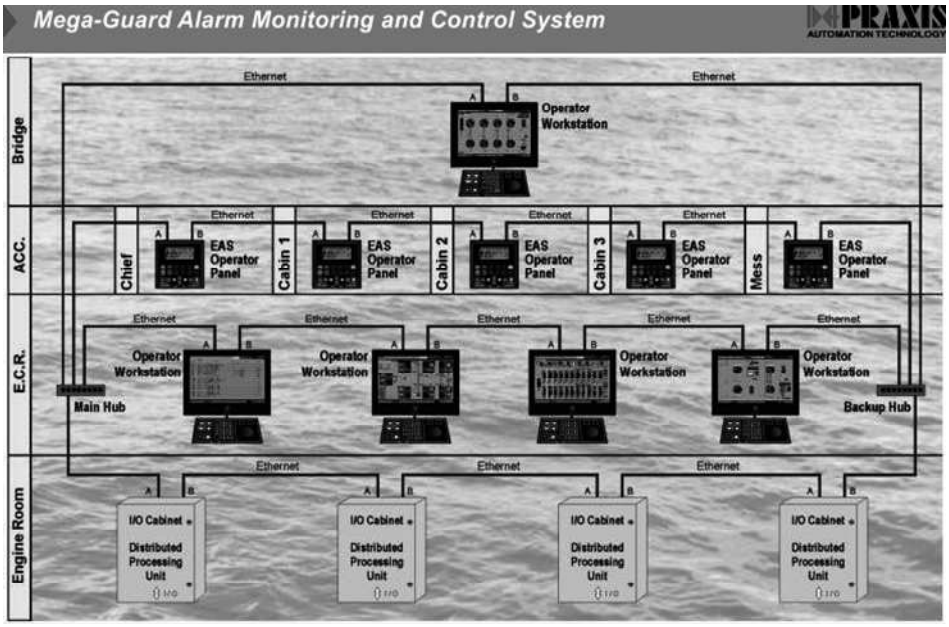
Nowe sterowniki Praxis i NiT wyposażone są w graficzny dotykowy panel HMI obsługiwany przez CPVis będący jednym ze składników środowiska CPDev (p. 5). CPDev wspiera także implementację soft-PLC przy użyciu komputera PC z modułami I/O (z National Instruments lub Inteco Kraków).

Kompilator CPDev został pierwotnie opracowany dla zastosowań w niewielkich sterownikach, o pamięciach programu i danych nie przekraczających 64 kB każda, z adresowaniem szesnastobitowym. Założenie to okazało się niewystarczające dla głównych sterowników systemu Praxis Mega-Guard bazujących na architekturze x86. Konieczne stało się więc opracowanie nowego kompilatora wykorzystującego adresowanie trzydziestodwubitowe.

3. Krótka charakterystyka systemu Praxis Mega-Guard

Jednym z wdrożeń środowiska inżynierskiego CPDev jest system do automatyzacji i nawigacji statków Mega-Guard, produkowany przez holenderską firmę Praxis Automation Technology B.V. [11]. System ten składa się z komputerów Marine PC, rozproszonych jednostek sterujących i paneli operatorskich połączonych siecią komunikacyjną [15]. Marine PC są stacjami operatorskimi bazującymi na komputerach z dyskami SSD i systemem operacyjnym Windows 7 Embedded. Jednostki sterujące mają budowę modularną. Sterownik oparty na procesorze ARM LPC i rozszerzające go moduły I/O montowane na szynie DIN połączone są magistralą CAN. Jednostka taka wykonuje programy opracowane w środowisku CPDev przystosowanym dla systemu Mega-Guard jako PAL-1131 (*Praxis Automation Language*). Panele operatorskie występują w różnych wykonaniach, dedykowanych dla poszczególnych podsystemów. Najprostsze panele zawierają zazwyczaj niewielki monochromatyczny wyświetlacz LCD i kilka do kilkunastu przycisków pełniąc rolę prostego interfejsu HMI oraz umożliwiając monitorowanie alarmów. Nowe wchodzące do produkcji panele pozwalają na tworzenie konfigurowalnych interfejsów HMI na kolorowym graficznym wyświetlaczu LCD przy pomocy narzędzia CPVis (p. 5). Poszczególne komponenty systemu Mega-Guard (Marine PC, jednostki sterujące, panele operatorskie) połączone są redundantną siecią Ethernet. Topologia sieci jest gwiaździsta, pierścieniowa lub mieszana.

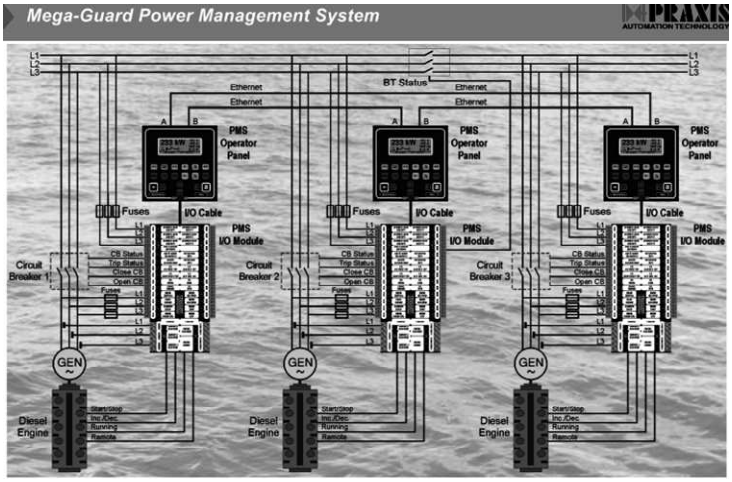
W obrębie systemu Mega-Guard można wyróżnić podsystemy mogące funkcjonować autonomicznie, bądź w kooperacji z pozostałymi. Głównym z nich jest podsystem monitorowania alarmów i sterowania AMCS (*Alarm Monitoring and Control System*), którego architekturę pokazano na rys. 2.



Rys. 2. Architektura podsystemu monitorowania alarmów i sterowania [11]

Fig. 2. Architecture of Alarm Monitoring and Control System

Podsystem AMCS może obsługiwać do 24 tys. sygnałów I/O. Stanowi on bazę umożliwiającą integrację kolejnych podsystemów. Kontrola napełniania zbiorników (*Valve Control and Monitoring System*) pozwala na zdalne sterowanie pompami i zaworami podczas załadunku, tankowania i balastowania. Wspiera ją system określający zawartość cieczy w zbiornikach (*Tank Gauging and Monitoring*). Jednym z bardziej rozbudowanych podsystemów jest zarządzanie energią (*Power Management System*). Podsystem ten odpowiada za sterowanie i ochronę generatorów zapewniając synchronizację i rozdział obciążenia pomiędzy poszczególnymi zespołami silnik – generator (max. 16 zespołów). Każdy zespół wyposażony jest w osobny sterownik i panel operatorski. Przykładową architekturę pokazano na rys. 3.



Rys. 3. Architektura podsystemu zarządzania energią [11]

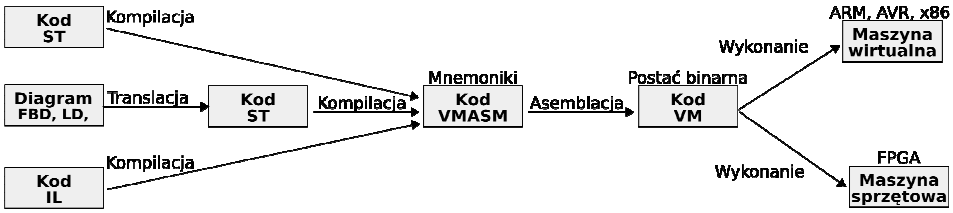
Fig. 3. Architecture of Power Management System

Obsługa napędów, przekładni, pędników kierunkowych itp. z mostka kapitańskiego jest możliwa dzięki podsystemowi sterowania napędami (*Propulsion Control System*). Zintegrowana nawigacja (*Integrated Navigation System*) obejmuje obsługę radarów ARPA X i S oraz pomocniczych urządzeń nawigacyjnych. Pozycjonowanie dynamiczne (*Dynamic Positioning System*) ułatwia manewrowanie statkiem, np. w trakcie zbliżania do platformy wiertniczej. Aktywność oficera na mostku jest monitorowana (*Bridge Navigation Watch*) pozwalając na wezwanie pomocy w sytuacjach alarmowych (przedłużająca się nieobecność, zasłabnięcie). System detekcji pożaru (*Fire Alarm System*) wykorzystuje czujniki dymu, ciepła i płomienia. W przypadku niewielkich jednostek pływających do kontroli podstawowych parametrów wystarcza mały (max. kilkaset sygnałów) podsystem alarmowania i monitorowania (*Alarm and Monitoring System*). Obsługa świateł nawigacyjnych (*Navigation Light Control System*) jest jednym z najmniejszych podsystemów, zawierającym jedynie panel operatorski i moduł I/O. Sterowanie wycieraczkami na mostku (*Wiper Control System*) umożliwia niezależny wybór prędkości, ogrzewania, spryskiwania itp. dla dziewięciu okien.

Znaczna różnorodność funkcjonalna podsystemów składowych Mega-Guard jest warta podkreślenia, jako charakterystyczna cecha systemów automatyzacji i nawigacji statków. Konieczność uwzględnienia odmiennych wymagań poszczególnych podsystemów oraz pewnych specyficznych niuansów (jak np. obsługa protokołu NMEA stosowanego w nawigacji, reprezentacja współrzędnych GPS jako zmiennych DREAL o podwójnej precyzji itp.) stymulowała rozwój środowiska CPDev.

4. Prototyp sterownika FPGA-PLC

Przedstawienie zasady funkcjonowania sterownika FPGA-PLC należy poprzedzić krótkim omówieniem istoty działania kompilatora CPDev [12, 14]. Ogólną strukturę przedstawiono na rys. 4. Programy napisane w językach ST lub IL, jak też przetłumaczone do ST z diagramów FBD, LD lub SFC, są kompilowane do uniwersalnego kodu wykonywalnego zwanego VMASM (*Virtual Machine Assembler*), wykonywanego przez maszynę wirtualną VM na docelowym procesorze. VMASM jest językiem zbliżonym do asemblera, nie związanym z konkretnym procesorem, lecz zorientowanym na programowanie sterowników zgodnie z normą IEC. Maszyna VM (procesor programowy) jest napisana w języku C, dzięki czemu może być kompilowana pod różne platformy sprzętowe (AVR, ARM, x86). Bazowa maszyna VM uzupełniana jest niskopoziomowymi procedurami producenta tworząc oprogramowanie podstawowe (*firmware*) sterownika.

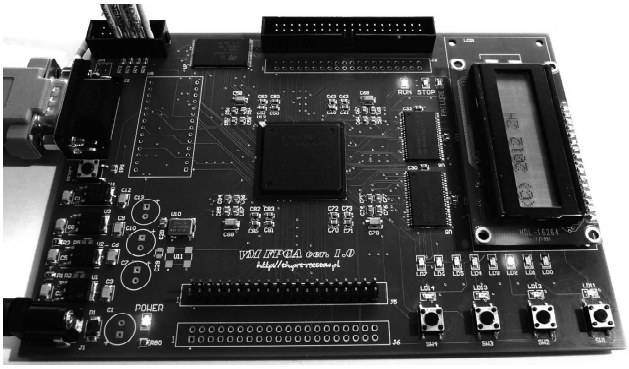


Rys. 4. Proces tworzenia oprogramowania w środowisku CPDev

Fig. 4. Software development process in CPDev environment

Wobec takiej zasady działania, sterownik PLC powstał jako struktura FPGA zastępująca programową maszynę VM maszyną sprzętową wykonującą kod VMASM tworzony przez kompilator CPDev na podstawie programów źródłowych.

Prototyp FPGA-PLC zbudowany na układzie Xilinx Spartan-3AN pokazano na rys. 5 [3]. W układzie zaimplementowano jednostkę CPU, koprocesor FPU (zmiennoprzecinkowy) oraz pamięć RAM. Płyta zawiera także pamięć NAND Flash dla programów VMASM, zegar RTC i interfejs RS-232. Moduły I/O i HMI są osobnymi płytkami (nie pokazanymi na zdjęciu).



Rys. 5. Prototyp FPGA-PLC z układem Xilinx Spartan-3AN

Fig. 5. FPGA-PLC prototype based on Xilinx Spartan-3AN chip

Architektury CPU, FPU i komponentów dodatkowych zostały wyspecyfikowane w języku opisu sprzętu Verilog i zaimplementowane w FPGA. CPU wymaga około dwóch tysięcy bloków logicznych, FPU około tysiąca, tak więc struktura FPGA-PLC może być implementowana w układach średniej wielkości.

W celu porównania prędkości wykonywania programów przez FPGA-CPU z programowymi maszynami VM przeprowadzono testy obejmujące obliczenia stałoprzecinkowe, operacje bitowe i indeksowanie tablic na sterownikach opartych o układy AVR, ARM, jak też na PC (Intel Core 2 Duo). Stosunki czasów wykonania testowego programu na tych platformach w porównaniu do FPGA, znormalizowane do tej samej prędkości zegara, przedstawiono w tab. 1. Maszyna FPGA-CPU okazała się 46 razy szybsza od AVR, 17 razy od ARM i 6 razy od PC.

Tabela 1. Wynik testów porównujących prędkość implementacji maszyny VM i FPGA

Table 1. The result of test comparing speed of VM and FPGA implementation

AVR/FPGA	ARM/FPGA	PC/FPGA
46	17	6

Ostatnio zaprojektowano także wieloprocesorowy prototyp sterownika FPGA-PLC zawierający kilka par CPU+FPU wymieniających dane przez pamięć globalną. Każda CPU realizuje osobne zadanie sterowania lub wykonuje fragment większego wspólnego programu. W takim przypadku konwencjonalny system operacyjny czasu rzeczywistego RTOS nie jest już potrzebny. Oprócz programowych bloków funkcyjnych prototyp obsługuje także tak zwane sprzętowe bloki funkcyjne HFB (*Hardware Function Blocks*) zdefiniowane w

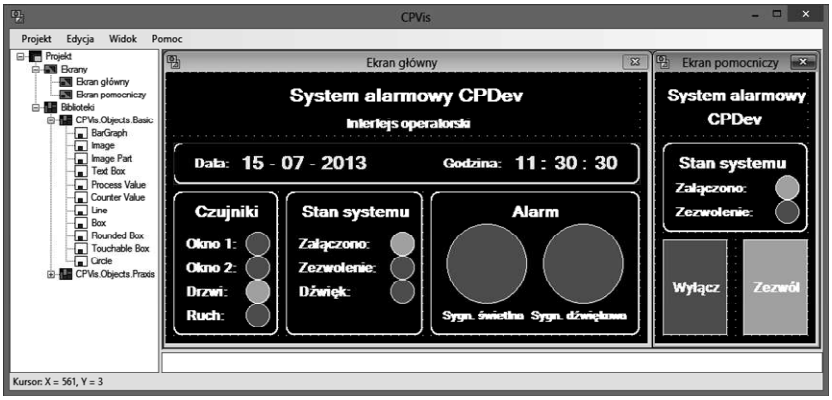
języku Verilog, dzięki czemu wykonuje się je z dużą prędkością. Kompilator CPDev został rozszerzony o instrukcje LOCK i UNLOCK, aby zapobiec jednoczesnemu wykonaniu tego samego HFB przez różne CPU.

5. Narzędzie wizualizacyjne CPVis

Jak widać na przykładzie firm Praxis i NiT (p. 2), sterowniki zintegrowane z panelem HMI są preferowane w wielu zastosowaniach. Z tego powodu środowisko CPDev zostało ostatnio rozszerzone o nowe narzędzie, nazwane CPVis, umożliwiające tworzenie konfigurowalnych interfejsów HMI [6]. CPVis zachowuje podstawowe właściwości środowiska, tj. wieloplatformowość i niezależność od sprzętu wizualizacyjnego (panele LCD/TFT, monitory itp.) jak też niezależność od bibliotek graficznych. Założono także, że zachowanie grafik HMI może być programowane w językach PN-EN 61131-3, w podobny sposób jak sterowanie.

Aby umożliwić obsługę różnych urządzeń wizualizacyjnych CPVis składa się z części zależnej oraz z części niezależnej od sprzętu. Zdefiniowano kilkanaście podstawowych funkcji graficznych, których implementacja zależy od sprzętu, takich jak `DrawRectangle`, `DrawStraightLine`, `DrawArc`, `DrawBitmap`, `FillPie` itp. Te podstawowe funkcje wykorzystywane są na wyższym poziomie przez niezależne od sprzętu obiekty graficzne, jak `Bar Graph`, `Process Value`, `Circle`, `Image` (bitmapa), `Touch Button` itp. Obiekty graficzne tworzone są poprzez deklarację w pliku bibliotecznym XML oraz implementację kodu rysującego w C przy użyciu wspomnianych podstawowych funkcji. Kształt i zachowanie obiektów graficznych wyznaczane są przez parametry dwojakiego rodzaju, podstawowe dla CPDev lub specyficzne dla CPVis. Parametry podstawowe obejmują typy danych zdefiniowane w normie PN-EN 61131-3, jak `BOOL`, `INT`, `REAL` itd., dzięki czemu program sterowania może wpływać na wizualizację poprzez przypisanie zmiennych globalnych do parametrów obiektu graficznego. Typy specyficzne dla CPVis to `COLOR`, `RANGE`, `FONT` i `COMPLEX` (złożony z innych typów).

CPVis składa się z edytora graficznego, bibliotek oraz części implementowanej w sterowniku (*runtime*). Główne okno edytora pokazano na rys. 6. Lewa część przedstawia projekt CPVis z dwoma przykładowymi ekranami, głównym i pomocniczym, oraz z drzewem bibliotek. Prawa część pozwala na tworzenie ekranów z obiektów graficznych. Tutaj jako przykład pokazano ekrany interfejsu operatorskiego dla systemu alarmowego.



Rys. 6. Główne okno programu CPVis

Fig. 6. CPVis main window

Końcowa faza projektowania obejmuje eksport danych wizualizacyjnych do pliku binarnego, wysyłanego następnie do sterownika. Na jego podstawie oprogramowanie *runtime* tworzy i aktualizuje wyświetlane grafiki. Część CPVis implementowana w sterowniku napisana jest w C/C++ i współpracuje z maszyną wirtualną VM CPDev. Zmiany wartości zmiennych podczas działania VM odzwierciedlane są na wyświetlaczu sterownika.

Większa elastyczność zarządzania obiektami graficznymi i ekranami jest możliwa gdy projektant opracowuje dodatkowe programy wizualizacyjne (po stronie CPDev) i łączy wynikowe zmienne z obiektami lub ekranami. Tego typu funkcjonalność jest dostępna w niektórych pakietach SCADA poprzez specjalizowane języki skryptowe lub Visual Basic. Tutaj jednak dla wygody projektanta te same języki normy i środowisko programistyczne wspierają zarówno sterowanie jak też wizualizację.

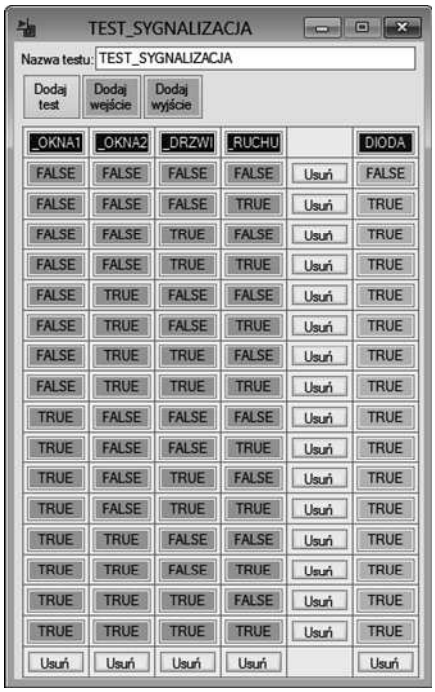
6. Testowanie jednostek POUs za pomocą CPTest

Zwiększenie jakości opracowywanego w CPDev oprogramowania sterującego osiągnięto poprzez udostępnienie nowego narzędzia CPTest pozwalającego na testowanie jednostek POU (*Program Organization Unit*), tj. funkcji, bloków funkcyjnych i programów [7]. Przypuszcza się, że dzięki niemu projektanci skorzystają z możliwości bardziej wnikliwego testowania oprogramowania (warto podkreślić, że główne zastosowania CPDev obejmują transport morski i lądowy [9, 11]).

Implementacje poszczególnych POU mogą być weryfikowane przez CPTest przy użyciu testów tabelowych i jednostkowych. Test tabelowy umożliwia zweryfikowanie w prosty sposób, czy przy danym stanie wejść

wyjścia POU mają oczekiwane wartości. Jest to szczególnie użyteczne dla bloków funkcyjnych ze zmiennymi logicznymi (BOOL), gdyż użytkownik jedynie wpisuje wartości wyjść w szablonie testu. Bardziej zaawansowane testy jednostkowe definiowane są przy użyciu dedykowanego języka.

Główne okno CPTest pokazano na rys. 7. Przedstawiono tu przykładowy test tabelowy dla systemu alarmowego. Składa się on z szesnastu przypadków testowych, które sprawdzają, czy wartość zmiennej DIODA jest równa TRUE gdy przynajmniej jedna ze zmiennych powiązanych z czujnikami (OTW_OKNA1, OTW_OKNA2, OTW_DRZWI, CZUJNIK_RUCHU) ma wartość TRUE. Rezultat przebiegu testów jest wyświetlany w osobnym oknie (nie pokazanym na rysunku).



Rys. 7. Przykładowy test tabelowy dla systemu alarmowego

Fig. 7. Sample table test for alarm system

Dedykowany język definicji testów jednostkowych zawiera instrukcje SET, RESET, ASSIGN, WAIT, LOG i ASSERT. Poza ostatnią nie wymagają one szczegółowego wyjaśnienia. ASSERT sprawdza, czy wartość wyrażenia logicznego podanego jako argument jest prawdziwa. Jeżeli nie, test kończy się niepowodzeniem. ASSERT obsługuje następujące operatory: LE, NEQ, LT, LTE,

`GT`, `GTE`. Przykładem użycia może być `ASSERT ISTRUE ENGINE` (sprawdza, czy zmienna `ENGINE` ma wartość `TRUE`), albo `ASSERT GT ALARM_LEVEL 0` (sprawdza, czy wartość zmiennej `ALARM_LEVEL` jest większa niż 0). Taki dedykowany język definicji testów wspiera podejście zbliżone do formy Arrange-Act-Assert [2]. Dzięki temu kod może być podzielony na trzy części odpowiedzialne za (1) przygotowanie bieżącego stanu, (2) wykonanie akcji i (3) sprawdzenie, czy wymagania zostały spełnione.

Gdy test zakończy się niepowodzeniem projektant może użyć wbudowanych w CPDev narzędzi debugujących w celu symulacji programu i sprawdzania pośrednich wartości zmiennych po napotkaniu pułapki. Edytory wspierają użycie pułapek także na diagramach FBD, LD i SFC.

Testy wykonywane przez CPTest mogą być definiowane bezpośrednio po utworzeniu POU, jednak kluczowe z nich powinny być zdefiniowane na początku podczas specyfikacji programu sterowania.

7. Modele SysML w CPDev

Wraz ze zwiększaniem złożoności zadań sterowania pojawia się tendencja przenoszenia początkowych faz rozwoju oprogramowania na wyższe poziomy abstrakcji. Społeczność akademicka preferuje podejścia oparte na modelowaniu, jak np. MDD (*Model Driven Development*), z graficznymi modelami reprezentującymi części systemu. Takie modele mogą być opracowane w języku SysML (*System Modeling Language*, oparty na UML), który wspiera zarówno modelowanie zorientowane obiektowo jak i modelowanie proceduralne [10]. W oparciu o [1, 5], poniżej krótko przedstawiono użycie diagramów SysML do projektowania oprogramowania w środowisku CPDev.

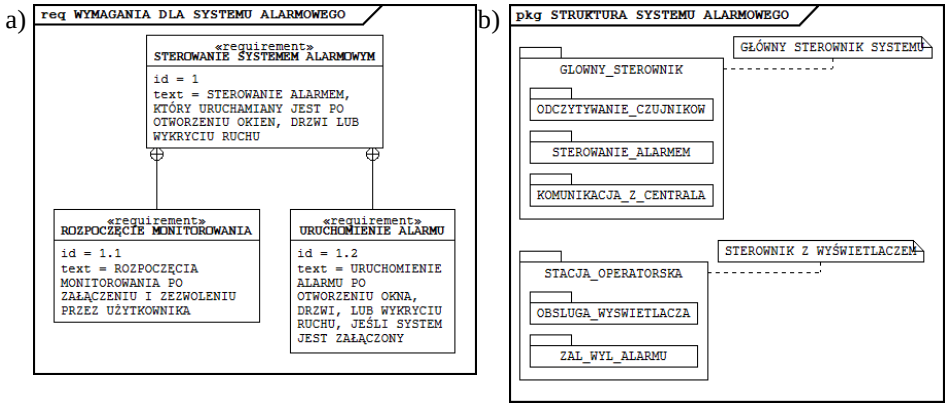
Spośród dziewięciu typów diagramów SysML wybrano cztery do modelowania poszczególnych aspektów systemu, tj. diagramy wymagań (*Requirements Diagrams*), definiowania bloków (*Block Definition Diagrams*), pakietów (*Package Diagrams*) oraz maszyn stanowych (*State Machine Diagrams*). Diagram wymagań specyfikuje zachowanie systemu zarówno w normalnych jak i sytuacjach nietypowych. W przypadku projektu złożonego z wielu bloków funkcyjnych niezbędna jest drzewiasta struktura wymagań. Węzły wyższego poziomu określają ogólne wymagania systemu, a węzły niższego poziomu wymagania dla poszczególnych programów, bloków funkcyjnych i funkcji.

Diagramy definiowania bloków specyfikują wszystkie typy jednostek POU (programy, bloki funkcyjne, funkcje) z przepływami jako wejścia i wyjścia. Na podstawie takich diagramów możliwe jest automatyczne tworzenie definicji POU w językach normy IEC. Szablony definicji stanowią bazę dla implementacji programowej uzupełnianą przez projektanta odpowiednim kodem

źródłowym. Pewne sekcje w diagramie definiowania bloków określają testy wykonywane na poszczególnych POU (poprzedni punkt).

Zgodnie z definicją zasobów diagramy pakietów reprezentują sterowniki z zadaniami. Tutaj również wymagane są dwa poziomy modelowania, wyższy dla sterowników i przyporządkowania zadań, niższy dla parametryzacji poszczególnych zadań, jak POU, czas cyklu itp. Zachowanie pewnych POU może być modelowane przez diagramy maszyn stanowych, które określają stany i warunki przejścia pomiędzy nimi. Takie diagramy mogą być automatycznie tłumaczone na kod źródłowy.

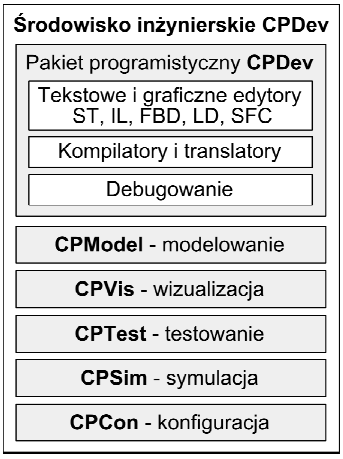
Zaproponowane podejście zostało wprowadzone do środowiska CPDev w formie prototypowego komponentu CPModel [5], który umożliwia tworzenie czterech typów diagramów SysML. Przykłady tworzonych modeli pokazano na rys. 8. Diagram na rys. 8a zawiera wymagania dla systemu alarmowego. Składa się on z trzech elementów <<*requirement*>> połączonych w taki sposób, aby powstała hierarchia wymagań. Dla każdego z nich podana jest nazwa, identyfikator oraz krótki opis. Diagram z rys. 8b modeluje strukturę systemu alarmowego korzystając z diagramu pakietów (*Package Diagram*). Wszystkie zasoby oraz zadania reprezentowane są przez pakiety.



Rys. 8. (a) Wymagania i (b) struktura dla systemu alarmowego

Fig. 8. (a) Requirements for and (b) structure of alarm system

CPModel jest ostatnim z komponentów CPDev, którego obecną strukturę pokazano na rys. 9. Środowisko CPDev IDE (*Integrated Development Environment*) zawiera edytory języków normy PN-EN 61131-3, kompilatory, translatory i debuggery. Pozostałe składniki umożliwiają modelowanie, implementację, testowanie, projektowanie wizualizacji dla HMI, symulację, konfigurację sprzętu i wsparcie w procesie uruchamiania (*commissioning*).



Rys. 9. Struktura środowiska inżynierskiego CPDev

Fig. 9. Structure of the CPDev engineering environment

8. Podsumowanie

Jakkolwiek pewne uzupełnienia i ulepszenia wciąż są potrzebne, CPDev stanowi obecnie dość kompletne środowisko inżynierskie dla programowania sterowników zgodnie z normą PN-EN 61131-3. Dostępne są edytory wszystkich języków oraz usunięto ograniczenia na rozmiar programów. Szybki prototyp PLC obejmujący jednostkę CPU z koprocesorem zmiennoprzecinkowym został zaimplementowany w FPGA. Opracowywanie interfejsu HMI w powiązaniu ze sterowaniem ułatwia projektowanie. Testy tabelowe i jednostkowe pozwalają na zwiększenie jakości oprogramowania. Diagramy SysML modelują złożone oprogramowanie przed implementacją kodu.

Bibliografia

[1] Chiron F., Kouiss, K.: Design of IEC 61131-3 function blocks using SysML. In Control Automation 2007. MED '07. Mediterranean Conference, 1-5.

[2] Grigg, J., Arrange Act Assert: http://c2.com/cgi/wiki?ArrangeActAssert, 2012.

[3] Hajduk Z., Sadolewski J., Trybus B.: FPGA-Based Execution Platform for IEC 61131-3 Control Software. Przegląd Elektrotechniczny, 2011, no. 8, 187-191.

[4] Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: Rozwój środowiska inżynierskiego CPDev do programowania systemów sterowania. W: Projektowanie, analiza i implementacja systemów czasu rzeczywistego. WKŁ, Warszawa, 2011, 151-162.

[5] Jamro M., Trybus B.: An approach to SysML Modeling of IEC 61131-3 Control Software. 18th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland, 2013, pp. 217-222.
[6] Jamro M., Trybus B.: IEC 61131-3 Programmable Human Machine Interfaces for Control Devices. Conference proceedings – 6th International Conference on Human System Interaction (HSI 2013), Sopot, Poland, 2013, 48-55.
[7] Jamro M., Trybus B.: Testing procedure for IEC 61131-3 Control Software. 12th IFAC/IEEE Conference on Programmable Devices and Embedded Systems (PDeS), Velke Karlovice, Czech Republic, 2013, pp. 192-197.
[8] LUMEL S.A.: http://www.lumel.com.pl, 2013.
[9] Nauka i Technika Sp. z o.o.: http://www.nit.pl, 2013.
[10] OMG, System Modeling Language (SysML) 1.3: http://www.sysml.org/specs, 2012.
[11] Praxis Automation Technology B.V.: http://www.praxis-automation.nl, 2013.
[12] Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L.: Programming controllers in Structured Text language of IEC 61131-3 standard. Journal of Applied Computer Science, 2008, no. 1, 49-69.
[13] Rzońca D., Sadolewski J., Trybus B.: Kompilator języka ST normy IEC 61131-3 na uniwersalny kod wykonywalny. W: Systemy Czasu Rzeczywistego (SCR). WKŁ, Warszawa, 2007, 189-198.
[14] Stec A., Świder Z., Trybus L.: Charakterystyka funkcjonalna prototypowego systemu do programowania systemów wbudowanych według normy IEC 61131-3. W: Systemy Czasu Rzeczywistego (SCR). WKŁ, Warszawa, 2007, 179-188.
[15] Trybus L., Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B.: Uzupełnienia środowiska inżynierskiego CPDev dla programowania holenderskiego systemu sterowania statków Mega-Guard, Napędy i sterowanie 6/2012, 98-103.

CONTROL PROGRAM DEVELOPER ENGINEERING ENVIRONMENT CURRENTLY

Summary

The paper presents an overview of the current functionality of the CPDev (Control Program Developer) engineering environment developed in Department of Computer and Control Engineering at Rzeszów University of Technology. The package is designed for programming PLCs/PACs according to IEC 61131-3 standard. The system is based on the concept of dedicated virtual machines being interpreters of executable code to increase the portability and versatility of control programs. The environment has been enhanced by support of all IEC languages (ST, IL, FBD, LD, SFC), HMI software design integrated with control software, unit testing of software components as well as by ability of modeling the structure and operation of complex programs in SysML. Tool for designing HMI interface is independent of the hardware platform, and allows to combine control with visualization using IEC languages. Table and unit tests allow to increase software quality. Models based on SysML diagrams support the early design stages of control software. New CPDev compiler allows to handle larger programs. In addition to virtual machines run on general-purpose processors, compiled programs can also be executed by FPGA-PLC prototype. Current industrial implementations of the CPDev environment include devices from Lumel S.A. Zielona Gora, Poland (SMC programmable controller), Praxis Automation

Technology B.V. Leiden, The Netherlands (Mega-Guard Ship Automation and Navigation System) and Nauka i Technika Sp. z o.o. Zaczernie/Rzeszów, Poland (StTr-760-PLC controller). Brief description of the Praxis Mega-Guard system has been presented as an example of the implementation.

Keywords: PLC, IEC 61131-3, FPGA, HMI, SysML.

DOI: 10.7862/re.2013.8

Tekst złożono w redakcji: sierpień 2013
Przyjęto do druku: grudzień 2013