Configurable Operator Interface for CPDev Environment

Marcin Jamro, Bartosz Trybus

Rzeszów University of Technology, Department of Computer and Control Engineering

Abstract: The paper presents a graphical extension to the IEC 61131-3 CPDev programming environment. The extension called CPVis provides development tools and runtime components to create an operator interface for control software. CPVis editor is used to design display pages. Graphic objects are selected from libraries and represent visual controls on the display. The target operator panel runs CPVis graphics runtime to process the display configuration. Update of the display is done by reflecting changes of variable values processed by CPDev virtual machine.

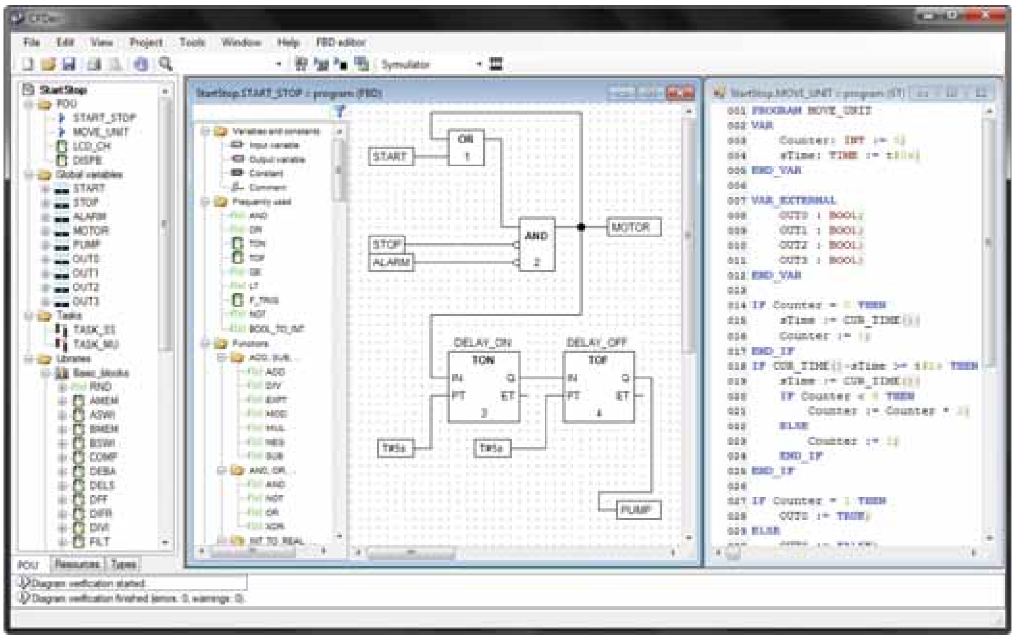
Keywords: operator interface, HMI, control systems, visualization

1. Introduction

Operator panels are an important part of control and monitoring systems. They are often used for examining values of process variables or setting parameters by providing Human-Machine Interface (HMI) [7, 11–12]. The former solutions using simple alphanumeric LCD displays or keypads are currently frequently replaced with high-resolution graphic panels with touch screen capabilities.

The paper presents the concept and implementation of a configurable graphical operator panel for CPDev engineering environment (Control Program Developer) [3]. CPDev, developed at Rzeszów University of Technology, contains programming tools and multi-platform runtime (virtual machine) for IEC 61131-3 [1] control software. During development of CPDev industrial applications, a need arose to extend it with a mechanism to create and maintain graphical HMI interfaces. The main assumptions of the extension, called CPVis (Control Program Visualizer), were: easy composition of visualization displays by using pre-defined graphical objects, low resource usage, coexistence and exchanging data with CPDev control software to access real-time variable values.

The paper is organized as follows: Section 2 presents briefly the CPDev engineering environment and its applications for industrial control; Section 3 describes concepts of the configurable operator panel and its software architecture; the visual editor used to design and configure HMI displays is described in Section 4; the section also presents graphical objects that are placed on the displays; Section 5 gives some details on the runtime part of the presented solution. The graphics runtime is executed on the target HMI panel, processes a binary configuration with display data and exchanges process information with CPDev virtual machine.



```
PROGRAM MOVE_UNIT
VAR
    Counter: INT := [illegible];
    sTime: TIME := t#[illegible];
END_VAR

VAR_EXTERNAL
    OUT0 : BOOL;
    OUT1 : BOOL;
    OUT2 : BOOL;
    OUT3 : BOOL;
END_VAR

IF Counter = [illegible] THEN
    sTime := CUR_TIME();
    Counter := [illegible];
END_IF
IF CUR_TIME()-sTime >= t#[illegible] THEN
    sTime := CUR_TIME();
    IF Counter < [illegible] THEN
        Counter := Counter + [illegible];
    ELSE
        Counter := [illegible];
    END_IF
END_IF

IF Counter = [illegible] THEN
    OUT0 := TRUE;
ELSE
```

```
Diagram verification started.
Diagram verification finished (errors: 0, warnings: 0).
```

Fig. 1. Main window of CPDev engineering environment
Rys. 1. Główne okno środowiska inżynierskiego CPDev

2. CPDev engineering environment

CPDev environment integrates tools for creating, compiling and running control software [4]. The programmer creates a new project in CPDev IDE (Integrated Development Environment), which main window is shown in fig. 1. Control program is composed of POUs (Program Organizational Units) which can be written in all five languages defined in IEC 61131-3 standard, both textual and graphical. These languages are: ST (Structured Text), IL (Instruction List), FBD (Function Block Diagram), LD (Ladder Diagram) and SFC (Sequential Function Chart) [2–3].

The source program is processed by the CPDev compiler which generates universal executable code. CPDev Virtual Machine must be implemented in particular target platform to execute the VMASM code [10]. The machine can be applied for Programmable Logic Controllers (PLCs), Programmable Automation Controllers (PACs) or distributed control systems (DCSs). Other components of the CPDev environment include hardware and communication configuration tool, simulator, debugger and a testing platform for POU unit tests [6].

CPDev has been implemented in SMC industrial controller from LUMEL S.A., Zielona Góra, Poland [9]. SMC operates as a central unit in small DCS systems and has been used in several applications involving measurements, control, monitoring and diagnostics. Mini-Guard Ship Control and Positioning System from Praxis Automation Technology B.V. [8], Leiderdorp, The Netherlands, is another application [5]. Mini-Guard consists of several types of dedicated controllers communicating over Ethernet. Another CPDev application is a soft-controller, i.e. PC equipped with I/O boards used for lab and teaching purposes. Current works involve applying CPDev to pumping stations and transportation.

3. Operator Interface architecture

Software architecture of the operator interface is presented in fig. 2. The upper part shows modules used during design stage. The main component here is a visualization project maintained by CPVis editor. The project contains information about one or more visualization displays (pages) that will be presented to the operator. Each display contains instances of graphical objects, i.e. visual controls like bar graphs, pie charts, indicators, numeric values, buttons, or bitmaps. Information about these objects (their definitions) is taken from graphic object libraries, either predefined or specific to the application.

As can be seen, the software structure of the CPDev operator interface is oriented towards graphical objects. The HMI designer's role is to compose the visualization displays by selecting objects from libraries and configure their parameters (position, size, colors, etc.). This is achieved with CPVis editor (see the next section) and does not require a designer to have programming skills.

The visualization project can be stored in XML file for later use or exported to a binary form which is used by the runtime part of CPVis.

The lower part of fig. 2 shows runtime components of CPDev operator interface. The components are sometimes referred as CPDev graphic subsystem and can be executed on a HMI panel, a controller equipped with a graphic LCD display or a PC acting as a monitoring station. CPVis graphics runtime interprets the binary file generated from the design project. The binary contains all visualization data, including displays, configuration, graphical objects and their parameters.

To implement dynamic update of the presented display a data exchange link is established between the graphics runtime and CPDev virtual machine (fig. 2). This link is platform specific. If these two components are run on the same hardware (the most common scenario) it can be implemented as a shared memory. In another case a communication protocol can be used (e.g. Modbus).

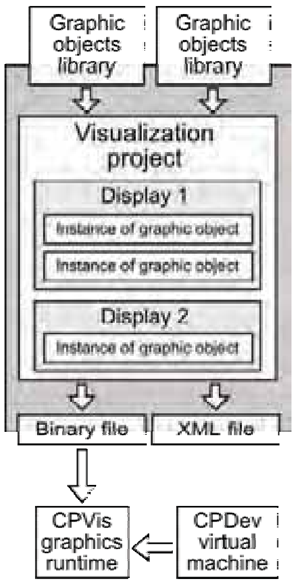


Fig. 2. Software architecture of operator interface for CPDev environment
Rys. 2. Architektura interfejsu operatorskiego dla środowiska CPDev

Each graphic object has a set of parameters (fig. 3). Instances of the objects are configurable during the design stage. Some parameters have default values, predefined in the graphic objects library. The defaults can be overridden if required. The user can specify values for the parameters, such as background color, numeric value, maximum value or text to be displayed. The libraries contain also information about type of the parameter, its name or description. The parameter values may not stay constant (default or fixed) during execution. Instead, the designer can bind a parameter to CPDev program variables (global variables only). In such a case, the actual parameter value will be fetched from the CPDev virtual machine and the graphical object state will automatically reflect the value of the variable. As seen, real-time update of the screen is also oriented towards objects.

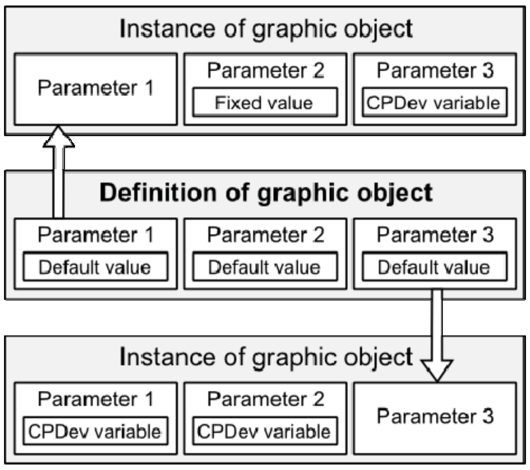


Fig. 3. Choosing values of parameters for graphic objects
Rys. 3. Wybór wartości parametrów obiektów graficznych

Each of the graphical object parameters has a specified type. The operator panel for the CPDev environment supports nine data types for this purpose. Some of them are simple IEC 61131-3 types: BOOL (logical), BYTE and INT (integers), REAL (floating point number). The other types are CPVis-specific: COLOR (index from a color palette), RANGE (with given minimum and maximum value), TEXT (character string), FONT (index of a font style), IMAGE (bitmap) or COMPLEX (can contain other types, excluding COMPLEX). Declaration of the CPDev variables that will be bound to the graphical object parameters should match their types such as BOOL, BYTE, INT, REAL. However, COLOR and FONT parameters can also be mapped to integer variables holding index values.

As stated above, graphic object definitions are stored in libraries. The concept of the operator interface allows to extend the predefined set of objects with a specific ones by creating a custom library. It will contain definitions of the custom objects. The object definition includes its parameters and visual characteristics. The visual representation of an object is defined in C language using universal drawing primitives like line, rectangle or circle. This way, the same drawing code is used by the development software (CPVis editor) and the graphics runtime.

It is worth mentioning that custom objects can reduce time needed by graphic processing, especially when limited resources are concerned. While applying CPVis to a particular application, it is a good practice to create a custom object instead of using the same pattern of graphic shapes multiple times. Such an approach can also positively influence efficiency of the design stage and simplifies project maintenance.

4. Editor of visualization displays

CPVis editor is a tool of CPDev engineering environment used to create and edit visualization displays. Design of the operator interface is an integral part of CPDev project, together with definition of POUs, control tasks, hardware configuration etc.

The editor works in WYSIWYG (What You See Is What You Get) mode. Left part of the main window (fig. 4) contains project tree with visualization displays. Two displays are defined in the sample project (fig. 4), namely *Display #1*, *Display #2*. The tree also shows graphic object libraries used by the project. Here, only one library is used (*CPVis.Object.Basic*). The project tree expands the library branch showing a list of graphic objects defined in the library. There are simple graphic objects like *Image* or *Image Part*, *Box* and *Rounded Box*, *Text Box*, *Line*. The more sophisticated objects are *Bar Graph*, *Slider*, *Pie Graph*, or *Dial Graph*. The main part

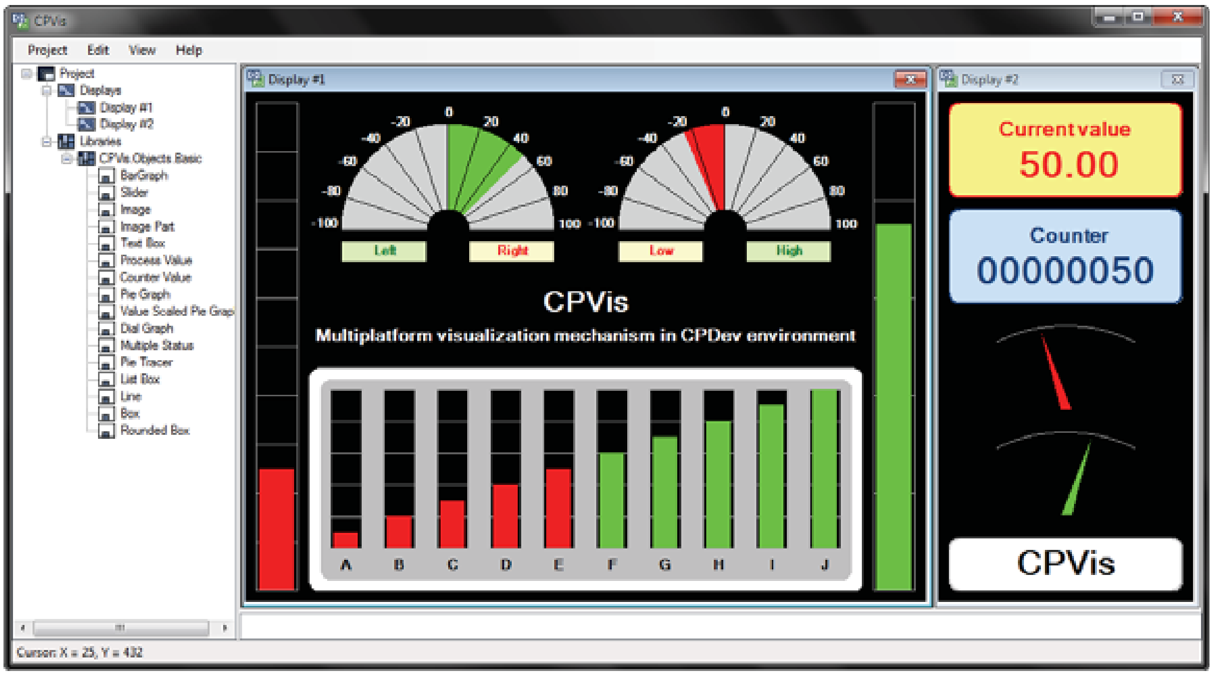


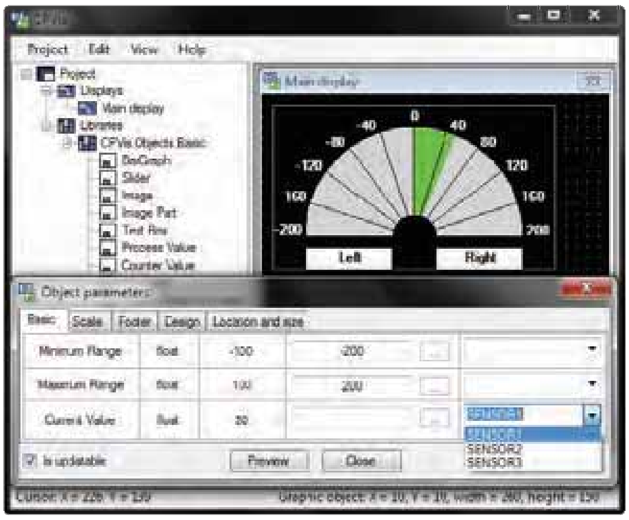
Fig. 4. Main window of CPVis editor
Rys. 4. Główne okno edytora CPVis

of the CPVis editor's window is used as a work area. Two displays are shown in fig. 4 – *Display #1* in the center and *Display #2* on the right. They contain objects from the library such as bar graphs, pie graphs, process values and others.

CPVis editor provides a set of functionalities that are used by the designer to prepare a configuration for the operator panel. The following features are available:

- visualization displays management,
- support of graphic object libraries,
- adding graphic objects to the display,
- moving or deleting graphic objects,
- setting parameters of graphic objects,
- undoing or redoing operations,
- saving and loading a project,
- adjusting display settings to user preferences (e.g. changing a size of editing grid).

The display is composed by dragging an object from the library tree and placing it on the display. Parameters of graphic objects can be set at design time, such as changing the color of a box (filled rectangle). As previously said, some parameters can be bound to CPDev variables, meaning that they will update their values during runtime. The editor allows to select global variables defined in CPDev project as a source for the parameter value, as shown in fig. 5, where the variable SENSOR1 is bound to the Current Value parameter of pie graph.



Minimum Range	float	-100	-200	
Maximum Range	float	100	200	
Current Value	float	20		SENSOR1

SENSOR1
SENSOR2
SENSOR3

Cursor: X = 226, Y = 130   Graphic object: X = 10, Y = 10, width = 280, height = 150

Fig. 5. Binding parameters to CPDev variables
Rys. 5. Przypisywanie parametrów do zmiennych CPDev

It can be seen in fig. 4 that the designer can adjust the graphic objects sizes (e.g. bar graphs). They can also be drawn with a custom aspect ratio. In addition, bar graphs can be placed either vertically or horizontally (this feature is not shown). Such capabilities allow flexible arrangement of the objects on the display and indicate the more important elements. The editing grid can be used for consistent object alignment.

Graphic objects can overlap on the display. The designer can change the Z-order in which the objects appear on the screen by moving them up, down, placing them directly at the top or at the bottom of the screen. However, to reduce CPU load during display refreshment, this feature is limited to a situation when the covered objects are not updatable (i.e. not bound to CPDev variables). CPVis editor will warn the designer if this limitation is not met. Usually, the covered objects are static parts of the display (images or constant texts) used to create a background for the top, updatable ones. For example, in the display presented in fig. 4 the row of bar graphs at the bottom is emphasized by the grey background bitmap with inscriptions.

Visualization display editing was implemented with the assumption that it will be intuitive and will not require special programming skills. Setting parameter values can be done via build-in editors for different value types. For example, color picker can be used instead of entering an index value in a palette, numeric range can be easily set with the specialized dialog etc. Upon user entry of a value, the editor performs basic correctness checking. For example, if the type of entered value is different from the parameter type, the value is rejected. Similarly, the editor will allow to bind only those CPDev variables which types match the parameter's type.

The visualization design is stored in XML file alongside the associated CPDev project. It can be later open for modification or extension. The editor automatically creates backups of the file. When the design is ready it can be stored in a binary file (CPV file) used at runtime.

5. Graphics runtime components

As presented in fig. 2, the CPV binary file created as a result of the design stage is used by the graphics runtime to produce display output. The CPV file contains information about defined displays (pages), graphic objects, that are used, and parameter settings. The structure of the file is shown in fig. 6.

CPV file begins with the header. It has fixed length and stores general information of the visualization file, such as version identifier and number of defined displays.

Next part of the CPV file contains definitions of visualization displays. Data of each display is stored in a separate section, starting with a header, which stores information about the display (e.g. its width and height). Instances of graphic objects being a part of the display are also stored in the CPV file. Each of them is identified by an unique identifier stored in the object's header, together with X, Y coordinates and its size. Object parameters are represented by the parameter header with information about its size (in bytes) and data type. Value of the parameter can be stored directly in the CPV file as fixed (i.e. set at design time or used default) or can be bound to CPDev variable. In the latter case, the CPV file holds address of the CPDev global variable.

The contents of the CPV file is copied to the memory of the target HMI device. If a communication mechanism between the development PC and the device is available, it can be used to transmit the visualization data. This is similar to the way in which a compiled control program is transmitted to the CPDev virtual machine [3].

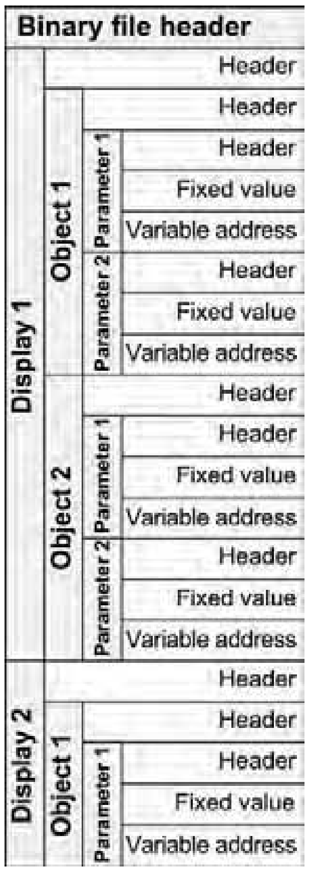


Fig. 6. Structure of CPVis binary file
Rys. 6. Struktura pliku binarnego CPVis

The firmware of the target HMI device should contain the CPVis graphics runtime that interprets the CPV binary. The runtime is written in C++ and can be ported to different platforms. Until now, ARM-based HMI panel and Windows PC have been used as targets.

To update the display, the function *drawScreen* () of the runtime should be invoked periodically. If the refresh cycle is synchronized with the cycle of the CPDev virtual machine, the updates will immediately reflect changes of the CPDev variables that are bound to graphic objects' parameters. The runtime will automatically exchange data with the CPDev virtual machine to fix current values of the parameters.

Since the designer can define multiple displays in the CPVis editor, the graphics runtime has to support switching the visualization pages. A global CPDev variable (integer) is bound to the visualization project and represents the currently visible display. If the value of the variable changes (for example as a result of pressing a button), the visualization display will automatically be changed to the one identified by the new value.

To improve drawing speed, a detection mechanism is used that determines whether an objects requires a redraw. For example, if the value of the CPDev variable bound to the object have not changed since the last call of *drawScreen()*, the object will not be redrawn. This significantly speeds up processing, because only a portion of the display is refreshed. Another implemented optimization causes skipping an interpretation of the parts of the CPV binary that will not be used (such as data of inactive objects or displays).

The CPDev program running on the virtual machine can interact with the graphic subsystem in a few ways. As already mentioned, it provides data for graphic objects that are used to show values of program variables. For example, a bar graph may represent a REAL variable in a graphical manner, while a text box can be used to display the value as a number. Another option is to bind a CPDev variable to a parameter that represents some visual characteristics of an object. This is especially useful with COLOR and FONT parameter types, since the appearance of an object (color, font style, size, etc.) can be changed by a CPDev program during runtime.

If enough computing resources are available, the CPDev program can not only be used to handle the display and panel elements (such as key buttons, switches, touch panel) but also to perform real-time control functions. For example, the runtime components (CPDev virtual machine with CPVis graphics runtime) can be used on a PLC/PAC controller with integrated HMI panel, thus both control and visualization are performed on the same device.

6. Summary

Including a graphical operator interface is nowadays an important part of the design of control and monitoring systems. Software can be developed accordingly to IEC 61131-3 standard by using CPDev engineering environment, mostly used for PLC, PAC, softPLC controllers, and distributed control systems applications. Some of CPDev users from the industry insisted on an extension that will allow to design and configure graphical operator panels. The visualization should interact with the CPDev virtual machine executing control software.

The solution proposed in the paper consist of a few cooperating software modules. The most important is the editor of visualization displays used on a development machine. The designer prepares the display layout and appearance by arranging graphical objects such as bar graphs, process values, bitmap images etc. The objects are taken from libraries, either predefined by the tool or application-specific.

The target HMI device executes CPVis runtime interpreter which processes a binary generated by the editor. It also runs the CPDev virtual machine with control programs. Drawing optimization reduces CPU usage during refresh of the display, hence the proposed solution can be applied for devices with limited resources.

References

1. IEC 61131-3 Standard: Programmable Controllers. Part 3. Programming Languages, IEC, 2003.

2. Jamro M., *Graphics editors in CPDev environment*, "Journal of Theoretical and Applied Computer Science", Vol. 6, No. 1, 2012, 13–24.
3. Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Extension of CPDev engineering environment for control system programming* (in Polish), [in:] Trybus L, Samolej S. (ed.), "Projektowanie, Analiza i Implementacja Systemów Czasu Rzeczywistego", WKŁ, Warszawa 2011, 151–162.
4. Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Running a distributed control and measurement system* (in Polish), [in:] Malinowski K., Dindorf R. (ed.), „Postępy automatyki i robotyki", Part 1, Vol. 16, Komitet Automatyki i Robotyki Polskiej Akademii Nauk, Wydawnictwo Politechniki Świętokrzyskiej, Kielce 2011, 168–181.
5. Jamro M., Rzońca D., Sadolewski J., Stec A., Świder Z., Trybus B., Trybus L., *Enhancements of CPDev engineering environment for programming Mega-Guard ship control system* (in Polish), "Napędy i sterowanie", 6/2012, 98–103.
6. Jamro M., Trybus B., *Executing and testing of programs written in IEC 61131-3 languages* (in Polish), Konferencja Systemy Czasu Rzeczywistego 2012, Kraków, 10-13.09.2012.
7. Nachreiner F., Nickel P., Meyer I., *Human factors in process control systems: The design of human-machine interfaces*, "Safety Science", Vol. 44, Iss. 1, 2005, 5–26.
8. [http://www.praxis-automation.nl] – PRAXIS Automation Technology 2012.
9. [http://www.lumel.com.pl/en/] – Lumel S.A., 2012.
10. Trybus B., *Development and Implementation of IEC 61131-3 Virtual Machine*, "Theoretical and Applied Informatics", Vol. 23, No. 1, 2011, 21–35.
11. Wittenberg C., *A pictorial human-computer interface concept for supervisory control*, "Control Engineering Practice", Vol. 12, Iss. 7, 2004, 865–878.
12. Zhang P., *Human-machine interfaces*, "Advanced Industrial Control Technology", Oxford, 2010, 527–555. ■

Konfigurowalny interfejs operatorski w środowisku CPDev

Streszczenie: W artykule przedstawiono rozszerzenie środowiska programistycznego CPDev o możliwość tworzenia graficznych interfejsów operatorskich. Rozszerzenie obejmuje narzędzia projektowe oraz oprogramowanie uruchomieniowe (runtime). Projektant interfejsu używa edytora CPVis do skomponowania ekranów wizualizacyjnych wybierając z bibliotek obiekty graficzne reprezentujące kontrolki. Na docelowym urządzeniu HMI uruchamiany jest moduł CPVis runtime, którego zadaniem jest interpretowanie danych wizualizacyjnych. Odświeżenie obiektów na ekranie graficznym odbywa się na podstawie informacji z maszyny wirtualnej CPDev, która dostarcza aktualne wartości zmiennych.

Słowa kluczowe: panel operatorski, HMI, systemy sterowania, wizualizacja

Marcin Jamro, MSc

Research assistant in the Department of Computer and Control Engineering at Rzeszow University of Technology. He received his MSc degree at Rzeszow University of Technology in 2012. His research focuses on software engineering on real-time systems.

e-mail: mjamro@kia.prz.edu.pl

Bartosz Trybus, PhD

Assistant professor in the Department of Computer and Control Engineering at Rzeszow University of Technology. He graduated from Faculty of Electrical Engineering, Automatics, Informatics and Electronics, AGH – University of Science and Technology in Cracow, Poland. He received his PhD degree in Computer Science in 2004. His main research concern real-time systems and runtime environments for control software.

e-mail: btrybus@kia.prz.edu.pl