User Manual

DCU CONTROL SYSTEM PLATFORM – CONTROL DESIGN AND PROGRAMMING

PROSYSTEMY, s.r.o.
Infoline@prosystemy.sk
www.prosystemy.sk

User Manual

DCU Control System Platform – Control Design and Programming

04/10/24

Contents

1 Control system structure.....5
2 Control design step by step.....7
2.1 Step 1 - Define control design specifications, propose control system and instrumentation structure.....8
2.2 Step 1-B - Build control system and instrumentation.....8
2.3 Step 2 - Create new control design project.....8
2.4 Step 3 - Draw a functional diagram of control application.....9
2.5 Step 4 - Control application simulation.....11
2.6 Step 5 - Translate control application functional diagram.....12
2.7 Step 6 - Set DCU communication settings.....12
2.8 Step 7 - Set DCU network and connect control design PC.....14
2.8.1 Single DCU.....14
2.8.2 Network of DCU units.....14
2.9 Step 8 - Load translated control application.....15
2.10 Step 9 - (discontiued) On-chip simulation control - HIL validation.....15
2.11 Step 10 - On-line control application validation.....16
2.11.1 Display control application diagram.....16
2.11.2 Display control variables real-time values.....16
2.11.3 Override (force) any control variable.....17
2.11.4 Modify parameters of any control function.....18
2.11.5 Record history values of control variables.....18
2.12 Step 11 - Build user visualization and batch servers.....19
2.13 Step 12 - (discontinued) Control application tuning and optimization.....19
2.14 Setting remote connection with User Command Center.....20
3 DCU Control Design Tool.....22
3.1 Introduction.....22
3.2 How to start DCU design tool.....23
3.3 DCU Design Toolbox main window menu.....24
3.4 DCU Design Toolbox main window.....25
3.5 DCU Design Toolbox – Process experiment tools.....30
3.6 DCU Design Toolbox visualization setup.....31
3.6.1 Data recording window.....32
3.6.2 Visualization window.....34

3.6.3 Alarm watch guard window *39*
3.6.4 Server virtual I/O window *39*
3.6.5 Operator Panels *41*
3.6.6 Batch servers window *44*
4 DCU control application **47**
4.1.1 Overview *47*
4.2 General parameters 49
4.2.1 Unit identification *49*
4.2.2 Sampling times *49*
4.2.3 Connection settings *50*
4.2.4 Encryption *50*
4.2.5 Users *51*
4.2.6 Modbus RTU *51*
4.2.7 RS485 serial link *51*
4.2.8 Hardware-in-the-loop modes settings (discontinued) *52*
4.2.9 Low-level parameters *52*
4.3 Control variable 53
4.4 Control functions 54
4.4.1 Physical analog input *55*
4.4.2 Physical analog output *55*
4.4.3 Physical digital input *56*
4.4.4 Physical digital output *56*
4.4.5 Virtual Input *57*
4.4.6 Virtual output *57*
4.4.7 RS485 Modbus RTU master analog output *58*
4.4.8 RS485 Modbus RTU slave analog output *58*
4.4.9 RS485 Modbus RTU master analog input *59*
4.4.10 RS485 Modbus RTU slave analog input *60*
4.4.11 RS485 Modbus RTU master digital output *61*
4.4.12 RS485 Modbus RTU slave digital output *61*
4.4.13 RS485 Modbus RTU master digital input *61*
4.4.14 RS485 Modbus RTU slave digital input *62*
4.4.15 Constant *62*
4.4.16 Nth order discrete transfer function (IIR filter) *62*
4.4.17 Math function with one argument f(x) *63*
4.4.18 Math function with two arguments f(x,y) *63*
4.4.19 Comparison of two analog values *64*
4.4.20 Analog input converter *64*
4.4.21 Dynamic switch *66*
4.4.22 Signal transformation d/dt, Gain, Offset, Saturation, Deadband, Polynom, Look-up, , Delay, format conversion *66*
4.4.23 Multi-signal real-value operation *68*
4.4.24 Multi-signal binary-value operation *68*
4.4.25 Binary edge processing, binary NOT *69*
4.4.26 Memory *70*
4.4.27 Counter / timer *70*
4.4.28 Schedule (time program) *71*
4.4.29 Variable state *72*
4.4.30 State space *73*
4.4.31 RST / PID controller *73*
4.4.32 Binary signal generator *76*
4.5 New special control functions 76
5 DCU Visualization **77**
5.1 Introduction 77
5.2 Web-based visualization development step by step 77
5.3 Building process view page 78
5.3.1 Process view page structure *78*
5.4 Web-based visualization deployment 79
6 DCU Batch service **80**

7 DCU hardware.....82
7.1 Electrical specifications.....82
7.1.1 DCU controller electrical wiring.....82
7.1.2 Analog inputs.....84
7.1.3 Analog outputs.....84
7.1.4 Digital (binary) inputs and outputs.....84
7.1.5 Communication connectors.....85
7.1.6 Auxiliary circuits.....85
7.1.7 Power supply.....85
7.2 Physical specifications.....85
7.3 Reset to default communication settings.....86
7.4 Loading of embedded real-time system.....88
7.4.1 Load using design tool.....88
8 Reference.....89
9 Annex A: Precision of AI module NTC10 for different thermistors.....90
10 Annex B: Precision of AI module PT1000 for different thermistors.....91
11 Annex C: List of DCU events and alarms.....92
11.1 DCU events and alarms.....92
11.2 Events of visualization.....95

1 Control system structure

General structure of a DCU control system is shown in 1. Two main elements of the control system are Dynamic Control Unit (DCU) controller that executes real-time process control, and visualization server, that records and displays process data and provides access to process. These two elements are interconnected by local network.

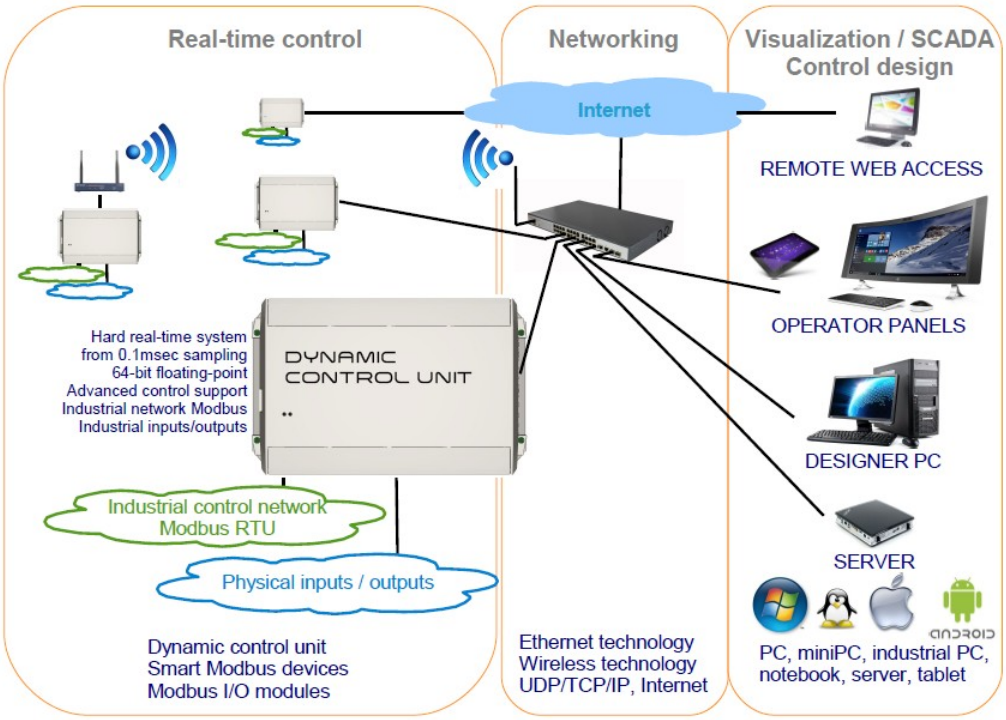


Figure 1: General structure of a DCU control system.

Software structure of a server or a designer PC or an operator panel is shown in 2. User Command Service (UCC) is communicating with DCUs and records data to MySQL database. UCC also executes requests from design platform and from visualization platform.

Device that uses remote web access to the process needs only an internet browser of any kind.

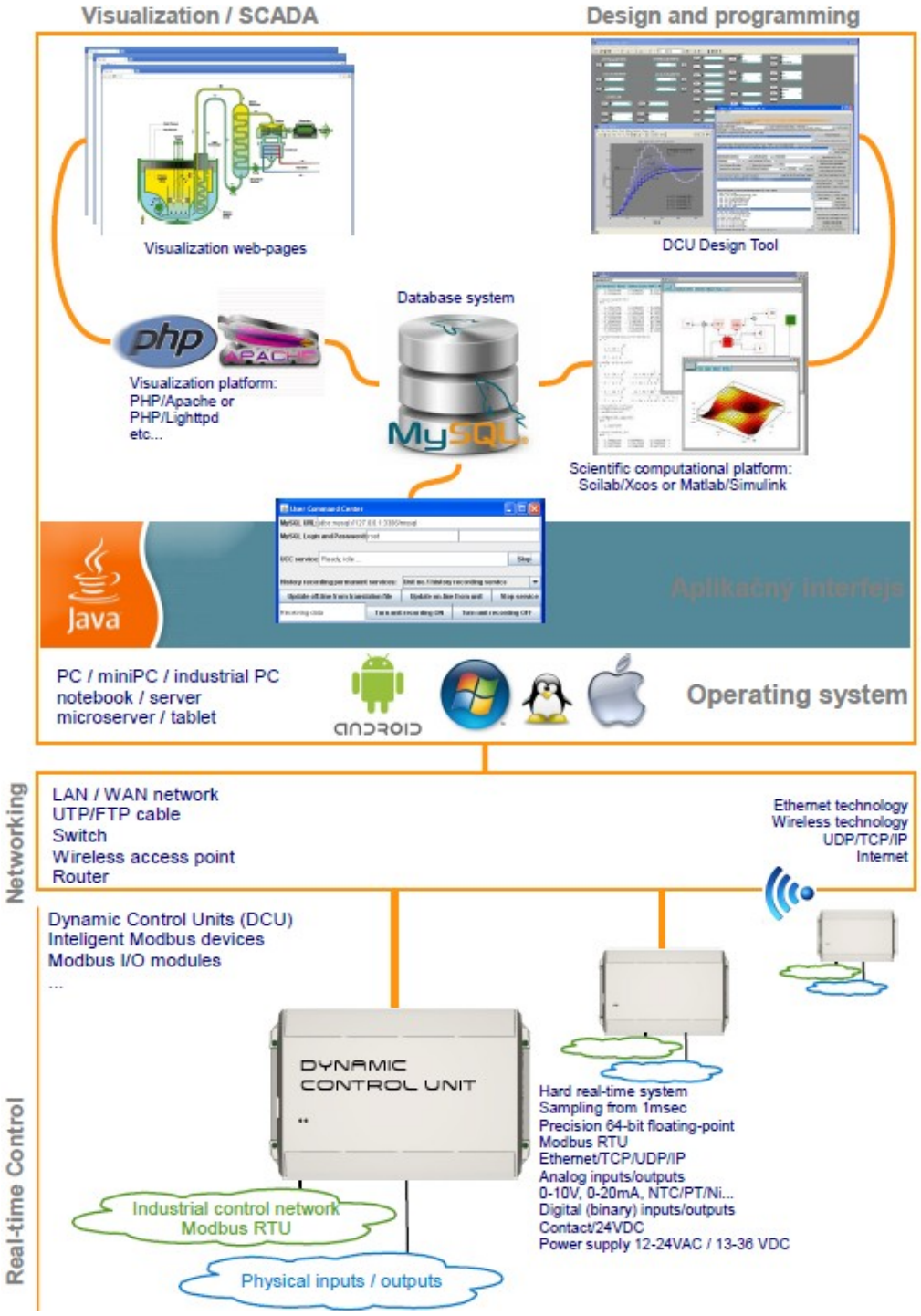


Figure 2: Software structure of DCU control System Platform.

2 Control design step by step

In this section, procedure of control design is described in detail. For quick and simple start of control system design see [7]. The entire procedure of designing and implementing control system of a technological process can be described by the following procedure:

Step 1 - Define control design specifications, propose control system and instrumentation structure. Result of this step should be instrumentation and electrical diagrams, operational requirements, such as operational panels, batch servers, remote access.

Step 1-B - Build control system and instrumentation. Connect DCU to power supply, sensors, detectors, valves, other action devices. This step may be done in parallel with next steps, but it must be finished before steps 10 and 11.

Step 2 - Create new control design project inside DCU design tool and import into the project initial functional diagram (diagram with defined empty DCU control unit with only inputs and outputs) or an existing diagram with similar control strategies as those supposed in the new project. Design tool for Scilab/Xcos or for Matlab/Simulink can be used for this task.

Step 3 - Draw control application into the functional diagram of the new project. The control application must guarantee all required process control performance specifications. Drawing of the control application is in our case performed within dynamic simulation tool (Xcos of Scilab or Simulink of Matlab).

Step 4 - Simulate the functional diagram control loops if necessary in order to validate control algorithms before using them to control real process. Since the functional diagram is drawn in dynamic simulation tool, dynamic simulations are possible. Dynamic simulations makes sense if a dynamic process model is at hand. The model must be drawn inside functional diagram and it must be properly interconnected with the control application.

Step 5 - Translate control application functional diagram into DCU-form diagram suitable for loading into DCU unit.

Step 6 - Set communication settings for each DCU.

Step 7 - Set DCU network and connect control design PC. LAN Ethernet connection is used. Control unit and PC are either in the same local network connected via router or they can be connected peer-to-peer with crossed Ethernet cable.

Step 8 - Load translated control application into control units.

Step 9 - On-line control application validation. Control application loaded into DCU is started and its real-time behavior is monitored and analyzed by control design engineer.

Step 10 - (optional) Control application tuning and optimization. Coefficients of PI/PID/RST controllers, digital filters, gains, etc. may be tuned or say optimized to improve robustness, speed, operational costs according to priority of technology engineers.

Step 11 - Build user visualization of process control system. If a user (operator, manager) is supposed to monitor and control in some way process control behavior a web-based monitoring and visualization may be developed on-measure according to

its requests. Alarms may be defined in alarms_definition_server,xml file placed in "Visualization/process" folder. Batch servers with appropriated tasks may be defined.

In the next sections control design and implementation is described in detail step by step.

2.1 Step 1 - Define control design specifications, propose control system and instrumentation structure

This step should produce at least the following results:

- Control system and Instrumentation Drawing - diagram displaying placement of sensors, detectors, valves and other control and instrumentation elements with respect to controlled technology (process or plant).
- Electrical circuit diagram of control system and instrumentation - defines electrical connections for instrumentation and control devices.
- Process control performance specifications - defined by technology engineer in cooperation with control design engineer, these specifications define desired behavior of the process under all possible operational conditions, alarm events to be indicated and recorded, desired user interface of the process, etc.
- DCU networking diagram. If a single DCU is used the diagram is not needed, only a connection to local area network may be specified. If several DCU units are used, it is preferable to create specific local area network only for DCUs using a separate router. Inside this network a data server may be placed for process data recording. DCUs may easily exchange process data without being disturbed by other LAN devices. Another alternative for DCU networking is to use Modbus RTU via RS485 serial link. This bus is strictly for process data exchange among control units and/or 3rd party process control devices. See section DCU communication capabilities for more details.

2.2 Step 1-B - Build control system and instrumentation.

In this step, control system is physically built., Sensors, valves, detectors, etc. are wired and connected to new DCU or prepared to be connected to DCU once programmed. DCU control network is built as well, power supply for technology and control systems are prepared and wired.

2.3 Step 2 - Create new control design project

Open design tool (see section Installation - Start control design tools) first. Then create new project select in design tool menu: Project → New. Tool will ask for placement of project so create a new directory of the project and create new project in this directory. Design tool will then ask to attach to project some initial functional diagram. Select either the functional diagram of basic example or any other functional diagram of a control application made previously.

The functional diagram of the basic example contains defined 1 DCU with pre-defined all physical inputs and outputs. Select an existing control application functional diagram if the new application to be created is similar to the existing one.

When creation done, version 1.0 of the project is automatically created and the selected diagram is copied into the new project directory.

2.4 Step 3 - Draw a functional diagram of control application

DCU controllers are programmed using functional diagram of desired control application. A control application functional diagram represents :

Desired behavior of control unit outputs with respect to unit inputs.

Instead of classical programming language, diagram is used to represent all control algorithms. A functional diagram was proved to be the most convenient method for designing simple as well as the most sophisticated control algorithms or even entire control applications. In automotive, aerospace, or control theory fields control algorithms are developed and validated primarily in form of control diagrams. The diagram brings not only clarity, but also possibility to represent the algorithm mathematically. Thanks to these features advanced optimization and analysis tools may be used to optimize and analyze control algorithms.

A functional diagram is a network of interconnected functional blocks. A function block can have input ports and output ports. Different block inputs are connected to different block outputs. Each functional block represents a function which is evaluated over inputs and function result is sent to block's outputs. Example of functional diagram is shown in Figure 3 at the top.

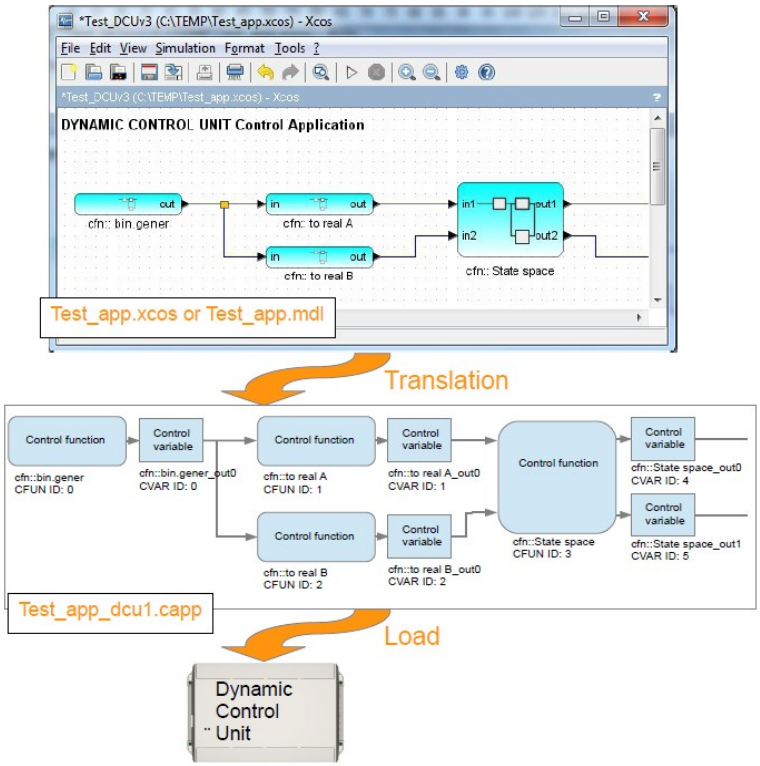


Figure 3: Control application diagram structure.

A functional diagram is drawn in an appropriate software tool. In case of DCU Control System Platform, it is drawn within the most efficient dynamic simulation tools available: Scilab Xcos, or Matlab Simulink. These tools are especially

developed for drawing control diagrams, as well as for designing and simulation of most advanced control algorithms and process dynamic models.

Functional diagram drawn in Xcos or Simulink consists of functional blocks with input/output ports and connection links connecting ports as shows Figure 3 at the top. Structure of functional diagram as represented inside DCU controller is shown in at the bottom of the Figure 3. Apart from links the internal structure contains:

- **Control functions.** These functions correspond exactly to diagram functional blocks. For identification purposes, each control function has a name defined by user (obligatory starting with 'dcu::') and unique ID number assigned automatically during translation of diagram.
- **Control variables.** For each control function output one control variable is created by translation process. The variable holds actual and if needed previous results of the corresponding control function output. The variables and its actual values are used to monitor and even adjust behavior of control function. As in the case of control functions, a unique control variable ID number is assigned for identification. Name of control variable is generated as well. It is the name of control function that produces the output with '_outx' string added at the end, where x is index number of input starting from 0, see Figure 3. Different parameters need to be set including initial value, sampling time for recording value to database, or alarm values for activating alarm events. Note that alarm values may be defined inside visualization platform instead.

Notice that ID number of a control function is not correlated with ID numbers of corresponding output control variables. Control function or control variable is uniquely identified by its ID number and non-uniquely (in case of Xcos diagram) by its name.

As mentioned above, control application diagram represents behavior of DCU controller outputs with respect to DCU inputs. DCU supports the following types of inputs and outputs.

DCU Inputs

- Physical analog input. Real value depending on input configuration. Electrical signal between 0-10V or 0-20mA is measured and converted into desired range of real values. NTC thermistor, PT thermistor or other type of thermistor can be used as well, but conversion functional block must be used to obtain correct temperature values.
- Physical digital input. Binary value 0/1 is sensed as opened/closed contact.
- Virtual input. Real or integer or digital (binary) value received via real-time UDP/IP Ethernet connection from other DCUs or PCs.
- Modbus RTU analog input. Real value received via Modbus RTU industrial bus. RS485 serial bus is used for this Modbus communication. DCU can act as master or slave. Master actively requests needed input value from Modbus slave (in Modbus protocol read Holding registers (03) or read Input registers (04) commands). Slave passively waits for input value from Modbus master (master must send command write Holding register(s) (06/16)).
- Modbus RTU digital (binary) input. DCU Master actively requests needed

input value from Modbus slave device (in Modbus protocol read coil (01) or read Discrete Input (02) commands). DCU in Slave mode passively waits for input value from Modbus master (master must send command write coil (05)).

DCU Outputs

- Physical analog output. Real value depending on output configuration. Electrical signal between 0-10V or 0-20mA is sent to output terminal.
- Physical digital output. Binary value 0/1 is interpreted either as opened/closed contact or 24VDC/0VDC on output's terminals.
- Virtual output. Real or integer or digital (binary) value is sent via real-time UDP/IP Ethernet connection to other DCUs or PCs.
- Modbus RTU analog output. Real value sent via Modbus RTU industrial bus. RS485 serial bus is used for this Modbus communication. DCU can act as master or slave. Master actively transmits value to a Modbus slave device (in Modbus protocol write holding register(s) command (06/16)). Slave passively waits for output value request from a Modbus master (in Modbus protocol read Holding registers (03) or read Input registers (04) commands).
- Modbus RTU digital (binary) output. DCU Master actively transmits value to a Modbus slave device (in Modbus protocol write coil command (05)). DCU in Slave mode passively waits for output value request from a Modbus master device (in Modbus protocol read coils (01) or read discrete inputs (02) commands).

Behavior of DCU outputs with respect to its inputs is then defined using different mathematical, logical and signal processing control function blocks. Notice that DCU inputs and outputs are considered to be control function blocks as well. For complete description of all control function blocks see section Control functions and variables.

2.5 Step 4 - Control application simulation

The control application is defined inside functional diagram of dynamic simulation tool (Xcos, Simulink) and so to execute simulation standard procedure may be employed (see [1], [2]):

- Obtain process model
- Place process model into the functional diagram.
- Connect DCU's inputs and outputs to process mode.
- Add scopes, values displays and other blocks for monitoring simulation results into the diagram.
- Set simulation settings
- Start simulation.
- Observe simulation results

If a process model is not at hand constant values, steps, special signals may be connected to DCU inputs to simulate a particular operational conditions while

observing behavior of DCU outputs.

2.6 Step 5 - Translate control application functional diagram

To translate the functional diagram containing control application, close diagram and click “Translate diagram” button inside DCU design tool. Diagram must be closed because it is updated with control function ID numbers obtained from translation. The translated control application file is saved into appropriate project directory and translation directory. The translated control application file is stored in XML format with extension .capp. The file name is projectfilename_dcuX.capp, where projectfilename is the file name of the project, and X is ID number of the control unit, see Figure 3. XML format has advantage of descriptive and understandable data storage. To check translated application any XML browser can be used.

Notice that no compilation, linking into C/C++ and assembler is needed. Thus no C/C++/Assembler programming skills are needed to design and implement hard real-time embedded control application. Embedded hard real-time system inside DCU contains a functional diagram interpret (parser) which interprets and executes uploaded control application diagram.

An on-line monitoring/HIL (hardware-in-the-loop) simulation diagram is generated at the end of translation process. This diagram can be opened by selecting menu “On-line monitor/HIL simulation” → “Open on-line/HIL diagram”. On-line monitor/HIL simulation diagram is a copy of translated design diagram but control functions contains actual function ID numbers from translation. The on-line diagram has the same name as design diagram plus at the end of the name is appendix “_trsl”. The diagram is used to visualize actual real-time process control values directly in control application diagram or to start and visualize hardware-in-the-loop validation procedure.

Note: *This procedure is RAM memory and time demanding and in Scilab/Xcos it is feasible only for smaller diagrams.*

The translation process generates also web-page definitions (php files) for displaying on-line control application diagram in web-based visualization. These php files are automatically placed into control system project visualization folder. Each control unit has its control application file. The file name is capp_X.php where X is ID number of control unit. Super-blocks (subsystems) inside control application have its own php file. File name is capp_X_sb_Y_superblockname.php, where Y is an index of super-block generated in translation and superblockname is the label of the super-block.

When translation done, it appears in list of translations with specified index, translation date and time, and directory. Selecting particular translation in the list of translations will display all translated DCU control applications in list below. This is because functional diagram may contain several DCU units and each is translated and saved in a control application file separately.

2.7 Step 6 - Set DCU communication settings

For each translated DCU control application, communication settings are read from translation and placed into design tool along with DCU. For DCU selected in list of

translated DCUs, its corresponding communication settings are displayed below the list.

On the other side any new DCU has its communication settings set to default, namely:

- **DCU user access.** Only one user is defined
  - User ID number = 8
  - User security level=15
  - User password=”root”
- **TCP/IP access.**
  - DCU IP address = 10.0.0.10
  - User command TCP port = 1033
- **Encryption.**
  - User command encryption - disabled.
  - History recording encryption - disabled.
  - Real-time UDP encryption - disabled.

First access to a new DCU controller must be done not with communication settings from translated application but with default settings. For this purpose the button “Load Unit With Default Comm. Settings” is present in the design tool. By clicking the button selected DCU is loaded with translated control application using default (factory) communication settings. Up from that moment DCU has no more default communication settings but those defined in control application.

The communication settings of a DCU in design tool defines how the design tool will access directly DCU and its recorded historical process data.

The communication settings must be set according to actual placement of designer PC and process data history recording service. If the designer PC is connected directly to control network (see next section) and process history recording will be executed directly from designer PC, settings read from translated control applications will be used, no changes are needed.

However, if designer PC is not connected to local network (communication via Internet is considered) or process data history recording is actually running on a dedicated server, settings must be adjusted. There are these sets of communication settings:

- **DCU user access.** User ID number, user security level, user password. This set define DCU user connecting to DCU. Users are defined inside DCU. Designer should connect as a user with highest security level (15).
- **TCP/IP access.** DCU IP address, user command TCP port.
- **Encryption.** Enabled/disabled encryption for communication with DCU, enabled/disabled history recording encryption, encryption key.
- **User Command Center (UCC) remote access.** Enable/disable remote connection to, IP address of UCC database, TCP port of UCC database. If enabled, these settings specify, that for communication with DCU and its

history process data, other than local (designer PC) UCC will be used. This situation may occur when a process control system was already installed and launched together with process monitoring and visualization. The visualization system obviously include running UCC which performs tasks from visualization as well as history process data recording. Interruption of monitoring and visualization capabilities may be undesirable. In such case, designer may access to DCU and its history data via running visualization UCC rather than local UCC on designer PC.

2.8 Step 7 - Set DCU network and connect control design PC

2.8.1 Single DCU

If single DCU is used no DCU network is built. Designer PC is connected directly to DCU via crossover cable. Procedure is as follows:

- Connect DCU to power supply.
- Connect designer PC directly to DCU using crossed (crossover) Ethernet cable with RJ45 connectors.
- Change IP and sub-net mask of designer PC according to DCU IP address settings. New DCU has IP address 10.0.0.10 and sub-net mask 255.255.255.0, thus PC may be set for example to IP:10.0.0.2, sub-net mask must correspond to DCU sub-net mask, i.e. 255.255.255.0.
- Try connection with DCU using for example ping command. For Windows OS open “Command prompt” application and type “ping 10.0.0.10”. DCU should respond properly.
- To set new IP and sub-net mask, load translated control application using default communication settings. In DCU design tool open your project, select corresponding translated unit in list of translated units, and click button “Load Unit With Default Comm. Settings”. Design tool will try to upload application into unit using IP and TCP port which is set in all new DCUs. Upload will load entire control application as well as new communication settings, i.e. IP address, sub-net mask, TCP port. Application is launched immediatelly after load, it can be stopped in DCU design tool by clicking on button “Control App.Mode” and selecting “Stop control application execution”.

2.8.2 Network of DCU units

Control system consisting of several DCUs is networked using standard LAN Ethernet technology, see section DCU communication capabilities. Basic LAN Ethernet network consists of a router and networked devices (in our case DCUs). Switches/hubs and other networking devices may expand local network if more DCUs need to be connected into the same network.

To set DCU network properly at least basic parameters of the DCU network router and IP addresses of DCUs must be specified. These settings may be done directly on the process site or, more comfortably, in office and send the router and DCUs to be installed on site already preset. Procedure of setting network may follow these steps:

- Connect the router to power supply.

- Connect designer PC to the router using a standard Ethernet cable with RJ45 connectors.
- Change IP and sub-net mask of designer PC according to router's initial settings. For example if router has initial settings 192.168.1.1, 255.255.255.0, set IP to 192.168.1.2, sub-net mask must be the same, i.e. 255.255.255.0.
- Using an internet explorer connect to router's administration pages and set router main LAN settings, i.e. IP and sub-net mask (for example router IP: 10.0.0.1, sub-net mask: 255.255.255.0) and confirm new settings.
- Change IP and sub-net mask of designer PC according to router's new settings. In our example set IP to 10.0.0.2 and keep sub-net mask.
- Connect one new DCU to router (network) using standard Ethernet cable.
- Try connection with DCU using for example ping command. For Windows OS open “Command prompt” application and type “ping 10.0.0.10”. DCU should respond properly.
- To set new IP and sub-net mask, load translated control application using default communication settings. In DCU design tool open your project, select corresponding translated unit in list of translated units, and click button “Load Unit With Default Comm. Settings”. Design tool will try to upload application into unit using IP and TCP port which is set in all new DCUs. Upload will load entire control application as well as new communication settings, i.e. IP address, sub-net mask, TCP port. Application is launched immediately after load, it can be stopped in DCU design tool by clicking on button “Control App.Mode” and selecting “Stop control application execution”.
- Repeat last three steps for all DCU units.

2.9 Step 8 - Load translated control application

New DCU has communication settings at default values (see section Step 6 - Set DCU communication settings). If a new DCU is to be loaded, load may be done using button “Load Unit With Default Comm. Settings”.

Once the first control application has been loaded into unit, communication settings inside DCU are set according to control application general parameters and all further loading will be executed using button “Load Control Application into Units”.

Control application is not started after loading. Its execution can be started using the design tool by clicking button “Control App. Mode” and selecting “Start control application execution”. Execution may be stopped via the same button by selecting “Stop control application execution”.

When DCU is loaded with a control application it appears in list of on-line control units.

2.10 Step 9 - (discontiued) On-chip simulation control - HIL validation

Check control application uploaded into DCU dynamic model using the process dynamic model. The idea behind this validation technique is to check if control

application designed in a functional diagram editor (in our case dynamic simulation tool) is the same which was uploaded into control unit. This validation technique is sometimes called hardware-in-the-loop (HIL) validation. In case of DCU we only need to

- Switch on-line mode inside DCU design tool into “On-chip simulation control mode”
- Open on-line monitoring/HIL simulation diagram (generated automatically after translation, see. Step 5). The diagram can be opened by selecting menu “On-line monitor/HIL simulation” → “Open on-line/HIL diagram”.
- Interconnect the dynamic model of the process with control application (see Step 4).
- Start simulation of the diagram. Design tool will send simulated process variables connected to control application physical inputs into control unit. DCU passes received simulated physical inputs as if they were real measured physical inputs, then evaluates control application and send back into simulating PC computed physical outputs instead of applying them as real physical outputs.

This validation technique may be quite time consuming. It is used for most sophisticated process controls of highly sensitive and dangerous processes. Ordinary process control does not require this validation and on-line real-time process validation is used instead (Step 10).

2.11 Step 10 - On-line control application validation.

In this step, actual real-time performance of DCU control application is monitored, analyzed, and compared with desired expected behavior obtained in simulation. To perform an effective control application validation, designer needs a set of tools that provide simultaneously the following services:

- Display control application diagram.
- Display control variables real-time values
- Override (force) any control variable
- Modify parameters of any control function
- Record history values of control variables

2.11.1 Display control application diagram

Actual control application functional diagram is displayed to recall designer actual control system structure. To display actual functional diagram, dynamic simulation tool (Xcos or Simulink) may be used. Designer may also import the functional diagram picture into visualization image directory. The imported image has to be called “capp_x.png”, where x is ID number of the corresponding DCU controller. Designer may than display diagram of DCU no. X together with actual values through visualization pages by clicking on

“Control System”->”DCU X”->”apps”->”Control application diagram”

2.11.2 Display control variables real-time values

To show actual real-time values, designer's PC must be connected to control network

(see previous sections) then several options are available to see actual values:

- Option 1: On-line monitoring using control variable value tables of design tool.
- Option 2: On-line monitoring using standard web-interface.

Any of these options requires running monitoring service for corresponding DCU. Monitoring service may be started/stopped separately for each DCU by selecting DCU of interest in "On-line control unit" list and by clicking on "Monitor" button and selecting requested operation.

Monitoring service is recording only control variables that have specified non-zero monitoring period. So if a control variable is required to be monitored during on-line control application validation, parameter "Out.variable monitoring during design" must be set to desired frequency of value update (1=every sample, 2=every 2nd sample, … up to 65535).

Option 1 allows to display selected control variables in tables. Advantage of this option is that not only actual value but also alarm and override actual states are visible. Additionally designer may group and display together variables according to their logical relationship (for example set-point, measured process variable, control action), which is not always the case for functional diagram display. To open one table display control variables or control functions in "Control Unit Elements" list. Then select control variables to be displayed (control functions whose outputs need to be displayed) and click the button "Vars/Fun.Outs Value/State from Rec". Actual values and states are taken from running monitoring service. Click "Update" button to refresh values and states.

Notice that there is also button "Vars/Fun.Outs Value/State from DCU". The same table is shown by clicking the button, however actual values are not taken from monitoring service but directly from DCU. That is why update of this table is much more time consuming.

Option 2 is equivalent to option 2, only instead of using DCU design tool, standard web-interface (see next section) is used to display and update tables of control variables.

2.11.3 Override (force) any control variable

Designer may need at any time to override a control variable value during control application validation. Override or forcing, sometimes called setting variable to manual, means setting variable to a specific user-defined value for some time. Designer uses this operation, to test control system behavior with respect to different process values, or to cut-off dynamic of a part of control system from other part (cascade of 2 PID controllers).

Two options are available to execute override operation:

- Option 1: Override with DCU design tool.
- Option 2: Override with web interface.

Option 1, override control variable in design tool procedure:

- Open control application functional diagram
- Open parameter window of control function block which outputs control

variable to be overrided.

- Modify parameter “Out.variable override/forcing” to 1, which means constant override.
- Modify parameter “Out.variable override/forcing operation” according to desired operation:
  - (1) output value = override value
  - (2) output value = computed value + override value
  - (3) output value = computed value * override value
- Modify parameter “Out.variable override/forcing value” to desired value.
- Close parameter window via “OK” button and save entire diagram.
- After saving the diagram wait some time (1-2min) so the operating system updates its information about file. Maybe save the diagram twice. It can happen that if diagram is saved an immediately after that the button “Update Parameters With Diagram” is pressed (see next steps) the previous values will be uploaded into control unit. The most secure way is to save and close the diagram. This way we are sure that the changes will be performaed.
- Within DCU design tool enlist control variables (control functions) in “Control Unit Elements” list and select control variable to be overrided or control function whose output will be overrided.
- Click button “Update Parameters With Diagram”.

Advantage of this override procedure is that override is kept also inside project within control application diagram.

Option 2 depends on used web interface. Principle is similar to option 1 only control variable and its parameters are selected and handled via web interface.

2.11.4 Modify parameters of any control function

Designer may also need at any time to modify a parameter of a control function implemented within control application. Typically coefficients of PI/D controller are adjusted during control application validation. Two options are available to execute parameter modification:

- Option 1: Modification using DCU design tool.
- Option 2: Modification using web interface.

Procedures for executing modifications using option 1 or 2 are equivalent to corresponding option procedures of control variable override. Only instead of modifying control variable override parameters, control function parameters are modified.

2.11.5 Record history values of control variables

Historical values and states of control variables set for monitoring are automatically recorded when monitoring service is running. Default time span for recorded signals is 1day. To set longer time span go to menu Visualization Server Setup→Data Recording

To display a recorded control variable signal, i.e. history values evolution of a

variable in DCU design tool, select control variables from “Control Unit Elements” list and click button “Vars/Fun.Outs Signals”.

2.12 Step 11 - Build user visualization and batch servers

Standard DCU Control System Platform visualization can be completely defined and deployed using main menu 'Visualization Server Setup'. Process data recording, visualization pages, alarm watch guards can be defined. For more details see DCU Visualization .

2.13 Step 12 - (discontinued) Control application tuning and optimization

Once the control application validated and performing reasonably well, designer may approach to optimization and tuning of control application elements which control important process dynamics. Important dynamics are those dynamics that can not be neglected, mostly slow dynamics and resonance modes. Basic tuning techniques as well as the most advanced control design techniques may be easily employed in DCU design tool.

DCU and its design tool strongly supports advanced control design techniques. This one optional designer step may become major part of entire control design effort if unstable, hard-to-control process dynamics are present within process. First, some technological adjustments to improve process behavior should be searched. But if non available wast specter of advanced design techniques may be employed to get process under control: experiment signal design, process model identification, controller tuning, control system optimization, advanced controller design including robust control, optimal control, model predictive control, pole placement, iterative feedback tuning and many others techniques.

Most of these techniques are available for DCU design tool since they are implemented as modules (toolboxes) of DCU design tool development platform, i.e. Scilab (Matlab). However to use any of these techniques, real-time process signals are needed. Process experiment procedure is implemented within the design tool as response to this need. According to designer specifications the procedure perform the following set of tasks:

- Set history recording for all requested control variables.
- Start history recording for all requested control variables.
- Set constant overrides on all requested control variables.
- Send user-defined override signal (dynamic override) to all requested control variables.
- Collect response signals for requested control variables and feed them back to DCU design tool.

Process experiment procedure is in the heart of many advanced control design techniques as it brings experimental information about process dynamics. Example of a process experiment procedure applied to a design technique is shown in Figure 4. In cas of DCU Design Tool, experimental procedure may be launched simply by clicking “Launch” button while selecting “Process Experiment Procedure” in “RT Process Variable Experimental Tools” selector.

Procedure first opens user interface window for definition of an experiment first. Designer defines experiment simply by:

- Selecting control variables to be in constant override during experiment. Override operation, override value, duration and delay of override with respect to experiment start must be defined for each constant override.
- Selecting control variables to be in dynamic override during experiment. Override operation, override signal vector variable name, and delay of override with respect to experiment start must be defined for each dynamic override.
- Selecting control variables to be recorded during experiment. Response signal vector variable name where recorded values will be placed must be defined.

One DCU controller can serve up to 4 dynamic overrides simultaneously. Override signal vector variables must be set global when defined to be visible to DCU design tool. The same holds for response signal vector variables. Once finished the process signal responses are stored in specified response signal vector variables. Designer can now use functions of advanced design techniques implemented within development platform (Scilab/Matlab).

Note: *Remote connection has to be used, if control system consists of control units and process data visualization and recording server and designer wishes to perform the optimization on its own computer that is not the server. See the section on establishing remote connection with UCC (User Command Center). In other words experiment procedure is always performed on the computer/server where all process data are recorded.*

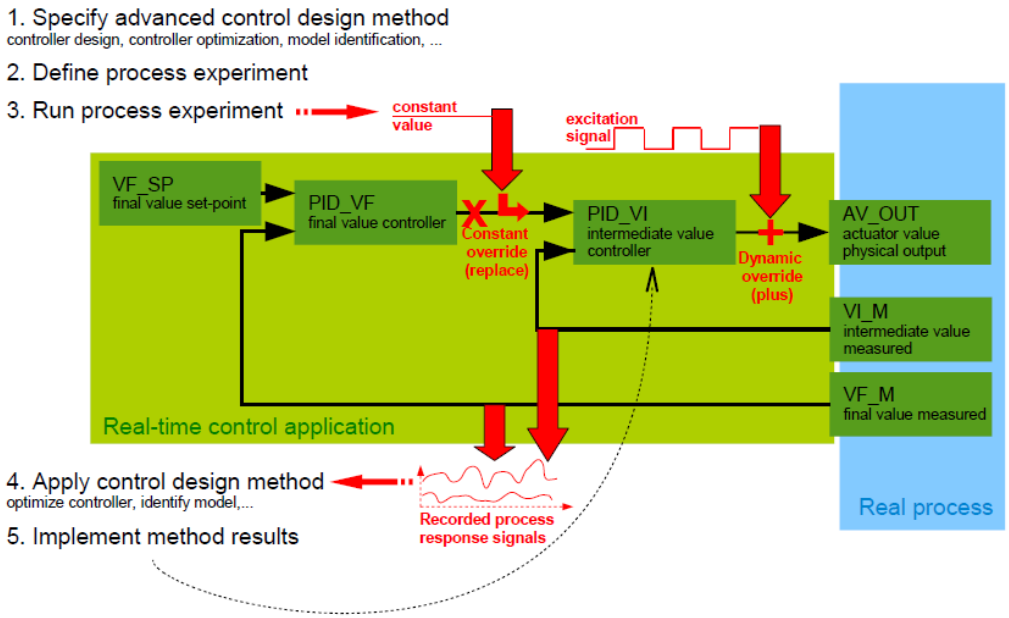


Figure 4: Example of typical process experiment procedure.

2.14 Setting remote connection with User Command Center

Remote MySQL database and its hosting computer/server must be set to accept

remote requests.

Setting remote computer/server

Computer/server must have allowed outside (non-local) connections with MySQL database. Firewall of the computer must be appropriately set to let pass through requests and responses of database. The simplest way to do so is to turn off the firewall of hosting PC.

Setting remote MySQL

In configuration file of MySQL make sure that parameter “skip-networking” is commented, i.e. set as follows:

skip-networking

Parameter “bind-address” will be set to *, which indicates that database will connect to any address:

bind-address = *

Create MySQL user with all privileges (grant all privileges) with ability to connect to any host (username@%).

CREATE USER 'admin'@'%' IDENTIFIED BY '***';

GRANT ALL PRIVILEGES ON * . * TO 'admin'@'%' IDENTIFIED BY '***' WITH GRANT OPTION MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0 ;

Remote connection with this MySQL database will pass via this user account.

Finally set remote connection settings in DCU design tool, see Figure 5.

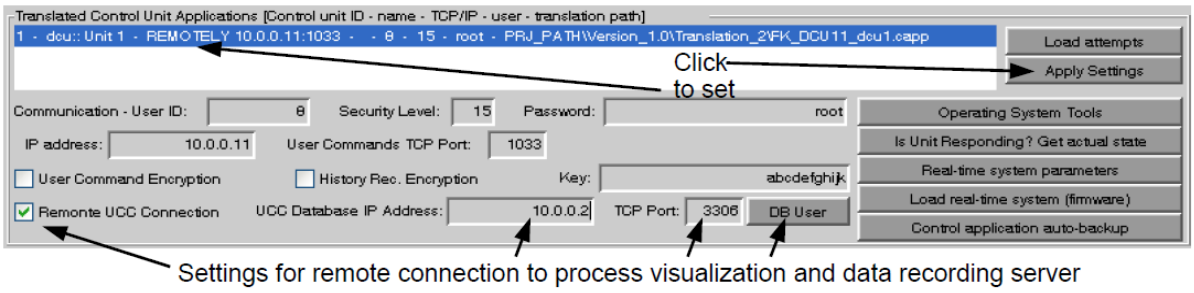


Figure 5: Remote connection settings.

3 DCU Control Design Tool

3.1 Introduction

DCU Control Design Tool is an application with graphical user interface that serves for programming, testing, and on-line monitoring of DCU (dynamic control unit) real-time controllers. The tool is programmed inside Matlab (Scilab) scientific computational platform using its specific language. The tool is one of many so called modules or toolboxes of this computational platform. Entire procedure of control design is not performed uniquely inside of the DCU design toolbox. Three main tools and its graphical interfaces are used for control design, see Figure 6.

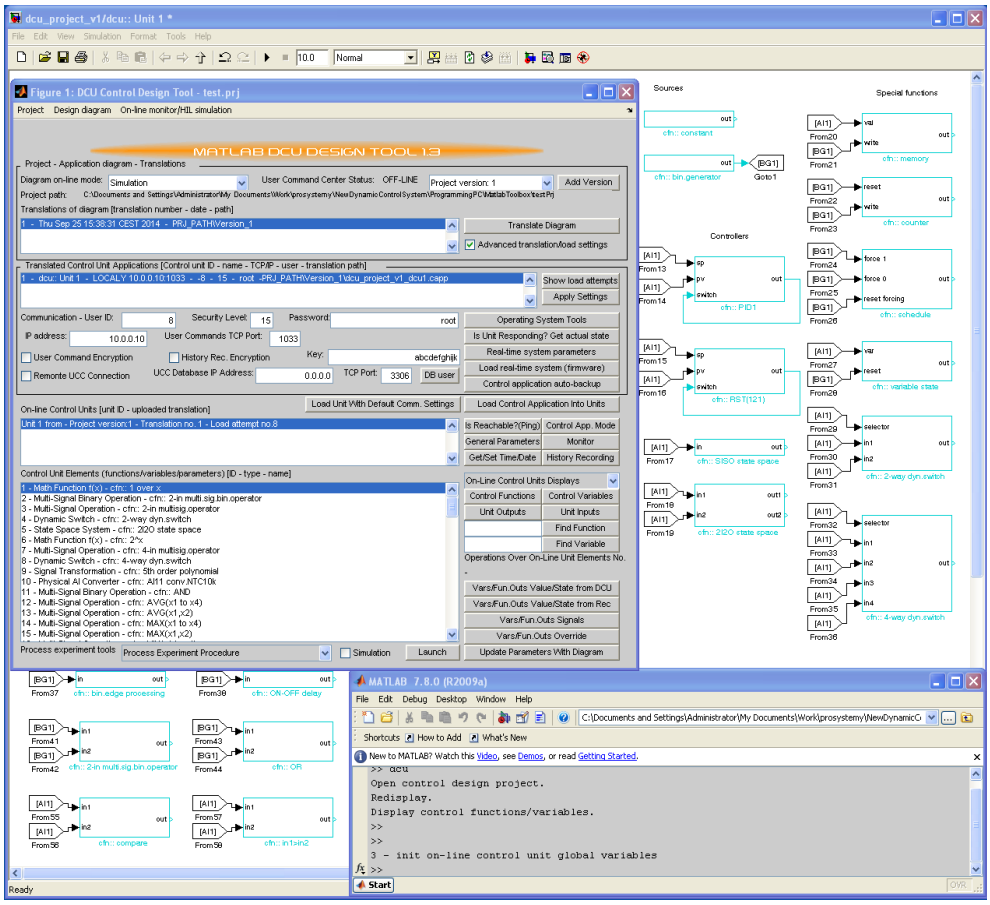


Figure 6: DCU control design toolbox main three tools.

The first design tool is DCU design toolbox with its graphical user interface, see Figure 8. It is used for project management, on-line monitoring, and control system management. User defines here control design projects, versions of the project, control application diagram translations, communications settings of translated real-time control units to connect on-line with the controller. Lists of control functions and process control variables of translated control units may be displayed. Units may be

connected on-line, actual values of process control variables may be displayed. Process experiment tools may be launched over selected control functions or variables.

Second tool used for control design is the Simulink (Xcos) toolbox, see Figure 7. It is used for programming real-time control applications. Simulink/Xcos is a dynamic simulator with comfortable interfaces for drawing and simulating diagrams of dynamic models and in our case control application diagrams. User programs real-time control application by drawing functional block diagram of desired control application. Library of available real-time control functions is available containing all standard industrial as well as advanced control functional blocks. Since the diagram is drawn in the dynamic simulator, control application may be validated via dynamic simulations. User may define different operational scenarios, include dynamic process models, place displays and graphs all around of the control application and launch simulation. Dynamic simulation belongs to the most advanced validation techniques available.

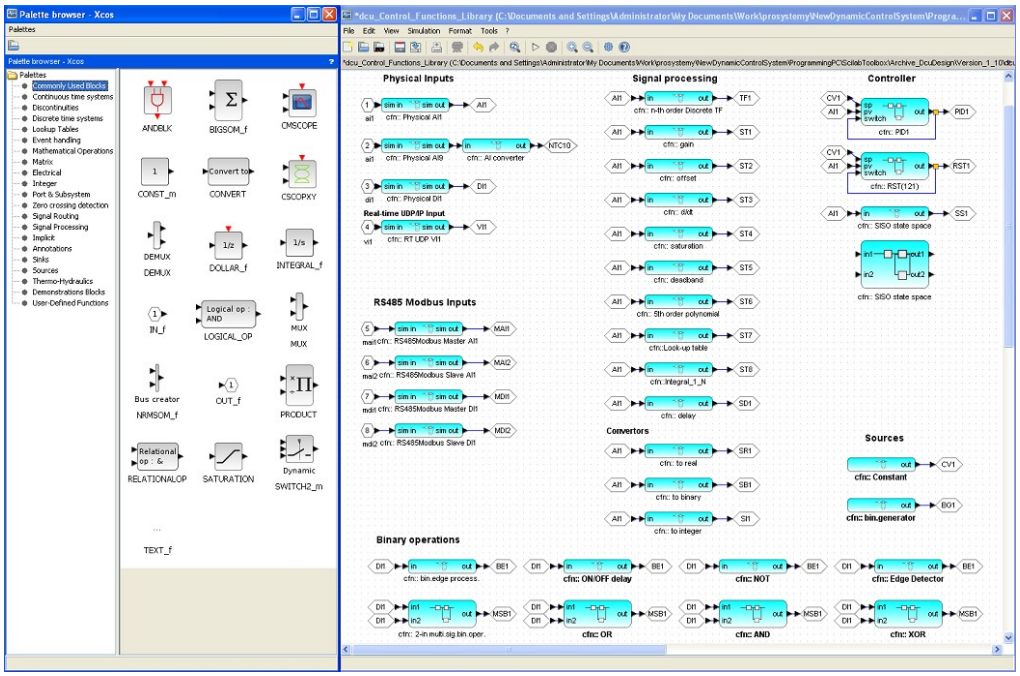


Figure 7: Graphical interface of dynamic simulator toolbox (Xcos).

Third design tool is the main command line window of the computational platform Matlab or Scilab (Figure 6 bottom right corner) and all available toolboxes that may be need for process experiments. For example, to prepare excitation signals, to analyze and display obtained process responses, etc. This third part of control design is optional and it is intended for experienced control systems engineers who require optimized and robust control system performance.

3.2 How to start DCU design tool

See Quick Start Guide for details.

3.3 DCU Design Toolbox main window menu

This section details main window menu locates at the top of the main window (Figure 8.).

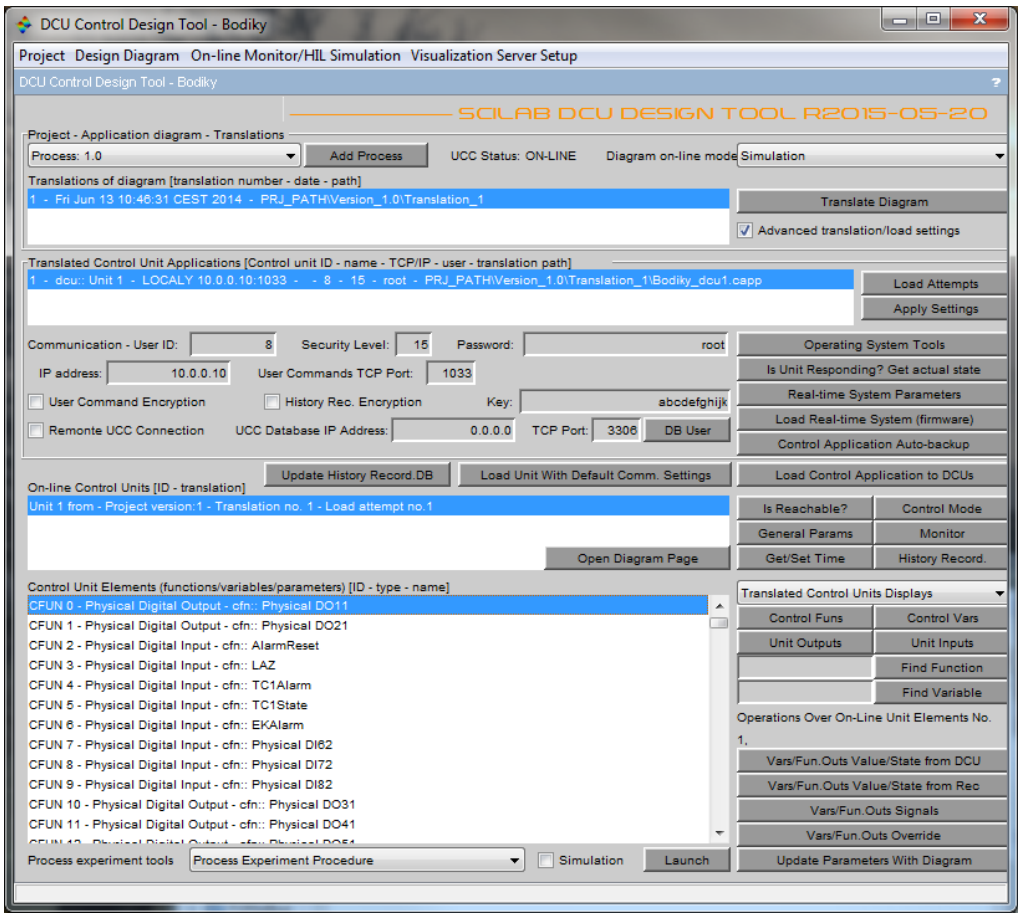


Figure 8: Graphical user interface main window of DCU control design toolbox.

Project menu groups actions concerning project, namely:

- **Open -** opens an existing project of DCU control design.
- **Save** - saves opened project in its actual state. No operation inside the design tool (translation, application load, ...) saves actual project. If at any time designer wishes to keep actual state of the project he must save the project using this menu.
- **New** - creates new project. User is asked for project file name and where to place it. We advise to create a new folder for each new project. Then it is asked to select an initial control application diagram (Xcos diagram). User may choose a diagram from a similar project or a diagram from one of examples. After selecting initial diagram, tool will automatically create Version 1.0 of the project, copy selected diagram into Version 1 folder, and open the copied diagram.
- **Close** - closes actually opened project, i.e. empties DCU design tool.

- **Settings** - opens window for setting
  - IP address of the designer PC (design tool needs to know IP address of the designer PC for some operations),
  - MySQL database login
  - MySQL database password
  - TCP base port for dynamic override. Dynamic override service uses ports starting at this base port up to base port + 31 to fetch signals into DCU. For example setting the port at 42200 means that ports 42200 up to 42231 may be used at any time by dynamic override service, i.e. whenever a process experiment is launched. All these TCP ports may not be used by other applications on designer's PC. Thus, all defined history recording TCP ports must be different from these dynamic override ports.

Design diagram menu groups actions concerning control application diagram (Xcos diagram), i.e. diagram that defines DCUs behavior:

- **Open actual -** opens actual diagram.
- **Attach new** - allows to attach a new diagram to project, but the existing diagram must be dropped first.
- **Drop actual** - deletes actual diagram path and file-name from project memory, but it is not erased from actual version directory.

On-line monitor/HIL simulation menu contains opening of on-line monitor/HIL simulation diagram.

- **Open on-line/HIL diagram -** opens actual on-line monitor/HIL simulation diagram. The diagram generated after translation of design diagram and serving to perform on-line monitoring or HIL validation of control application.

Visualization Server Setup menu contains actions related to definition and deploying visualization, operator panels, batch servers of DCU Control System. See section DCU Design Toolbox visualization setup for detailed description.

3.4 DCU Design Toolbox main window

Main window elements from top to bottom are described in this section.

Diagram on-line mode selector determines what behavior will have control application diagram when simulation of the diagram is launched. The following four modes are proposed.

- **Simulation** - normal dynamic simulation of control application is executed.
- **Real-time process control** - Switching into this mode will launch monitoring services for all on-line DCUs and starts monitoring inside each DCU. If the starting is successful, all on-line DCUs are monitored, i.e. all control variables with specified monitoring are recorded into history database. In this mode, launching diagram simulation will, instead of simulation, fetch actual values of control variables into the diagram. Hence, Xcos display or graph blocks will display real-time process data of control application.
- **On-chip process simulation control** - Launching diagram simulation in this

mode will start on-chip simulation mode inside each DCU as well as in control application diagram. In this mode, simulated process model (must be present in control application diagram) is considered to be controlled process and control application inside DCUs executes the control. In practice, physical input values simulated in diagram by a process model are fetched directly into DCUs where control application computes physical output actual values and send these back to simulation to be applied into process model. This control application validation technique is sometimes called hardware-in-the-loop validation.

- **On-PC real-time process control** - in this mode control application is evaluated in the simulated diagram using real-time physical input values. Computed physical outputs are sent back into DCUs and applied on real-time process. In this mode DCUs serve as an input/output card and control evaluation happens inside diagram.

User Command Center Status indicates if user command center responds properly to design tool (ON-LINE) or not (OFF-LINE).

Project process selector allows to select process (version) of the project to be worked on. Each process (version) has separate directory with its own control application diagram.

Add Process button adds a new process the project. A new directory is created in the project directory and user is asked to select a diagram to be copied to this new directory.

Project path text shows path to the actual project directory.

Actual diagram shows directory and file-name of the actual control application diagram.

Advanced translation and load settings check-box shows and hide information and tools of translations and translated DCUs.

Translate diagram button launches translation process over actual control application diagram. All DCUs found in control application are translated. For translation a new directory is created if demanded or an existing is replaced. If translation of a DCU is successful, translated application is saved into translation directory, the name of file has the following form:

<translated diagram file-name>_dcu<unit ID number>.capp

Translations of diagram list displays all translations of the actual project version. Selecting a translation will display in the list below all translated DCUs.

Translated Control Unit Applications list shows translated DCUs of selected translation. Number of DCUs is always equal to number of DCUs defined in translated control application diagram.

Show load attempts button print all attempts of control application load of the selected DCU translated application.

Communication settings defines how the design tool will communicate with each DCU. Selecting DCU in translated DCUs list will show actual communication settings for the selected DCU. These types of communications between designer PC and DCU are considered:

- Communication with DCU requiring some information or modifying a parameter is called user command.
- Streaming of historical process data from DCU into database is called history recording.

The following communication parameters need to be set:

- **DCU user settings:** Defines high-security-level user connecting to DCU. User ID, user security level (15 is highest level), and user password are defined.
- **TCP/IP setting:** IP address and TCP port for communication with DCU is defined. Local TCP ports that can be used for user commands to DCU is from 1033 up to 1036.
- **Encryption settings:** Enable or disable encryption for user commands and for history recording, set encryption key. One key is used for all encryptions.
- **Remote UCC settings:** If remote user command center communication is enabled, all user commands are transmitted vial local UCC to remote UCC using defined IP address, TCP port, and database login and password. This option is useful if a visualization server was already installed and it is recording process data for a network of DCUs. In such case designer may wish to work with DCUs and its historical data from its own designer PC without interrupting visualization server recording.

Set Communication Settings button will apply settings displayed in text fields of communication settings into selected DCU. The applied settings are used for application load and all actions applied on on-line control unit

Is Unit Reachable via this IP? (Ping) button will try to connect to IP address displayed in the text field of communication settings.

Is Unit Responding? Get actual state! button will try to retrieve information about actual state from DCU using communication settings present in the text field of communication settings. Designer can test the settings this way before applying them into selected DCU or he can check if DCU is on-line for example before launching application load.

Real-time system parameters button gets actual values of main DCU real-time system parameters and displays them in editable fields for modification. MAC address of the unit, internal high and low temperature alarms, and power supply high and low voltage alarms are displayed.

Load real-time system (firmware) button allows designer to select a file with a new version of real-time system and load it into DCU.

Control application auto-backup reads from selected DCU actual auto-backup delay and displays it in an editable window for modification. Control application auto-backup delay defines how long after a control application modification (a parameter change) a backup to removable EEPROM is made. It is defined in minutes and it may vary from 0 up to 254minutes, value 255 disables auto-backup. Backup procedure takes about 1 second and it blocks all control interrupts. This may be disturbing in case of fast control applications with sampling times below 100ms. Therefore designer may disable or delay auto-backup operation so that is done only from time to time, when changes in control application are accumulated. If auto-

backup disabled, removable EEPROM contains control application in a form it has been loaded into DCU.

Load Unit With Default Comm. Settings button allows designer to load translated control application into DCU using default communication settings (IP address 10.0.0.10, user ID 8, user security 15, user password "root") without defining them in communication settings text field. This is a practical tool for loading a new DCU.

Load Control Application Into Units button loads translated control application into DCUs using defined communication settings. Loaded units will appear in the list of On-line Control units below.

On-line Control Units list displays units with loaded control applications.

Is Reachable? (Ping) button will try to connect to IP address of selected on-line DCU.

General Parameters button gets actual values of control application general parameters and displays them in Control Unit Elements list.

Get/Set Time/Date button gets actual time and date of selected on-line unit and displays them in editable fields for modification. Designer may set any time and date or synchronize the DCU time with designer PC.

Control App. Mode button displays actual mode of control application execution (Stopped or Started) for selected on-line DCU. Designer may start or stop real-time control application execution.

Monitor button displays actual state of monitoring mode (Enabled/Disabled) for selected on-line DCU. Designer may enable or disable monitoring service inside DCU, i.e. streaming actual control variables values (those with defined monitoring) into PC (history recording service must be running in PC).

History Recording button displays actual state of history recording mode (Enabled/Disabled) for selected on-line DCU. Designer may enable or disable history recording service inside DCU, i.e. streaming actual control variables values (those set for history recording) into PC (history recording service must running in PC).

Open Diagram Page button tries to open web page with control application diagram on-line values of selected control unit. To use this button, visualization server must be set first (see 3.6DCU Design Toolbox visualization setup) and user must be logged into visualization web-pages with an account that enable to see control system web-pages.

Control Unit Elements list serves for display of control application elements for selected DCU. In particular, general parameters of control application, control functions or control variables may be displayed.

Control Unit Elements selector indicates on what DCU will be applied buttons "Control Functions", "Control Variables", "Unit Outputs", "Unit Inputs", "Find Function", "Find Variable". Selection "Translated Control Units Displays" applies these buttons on selected DCU from Translated Control Unit Applications list. Selection "On-line Control Units Displays" applies these buttons on selected on-line DCU. Thus, user can display list of control functions or variables from translated DCU as well as from on-line DCU.

Control Functions button displays list of all control functions for selected DCU.

Control Variables button displays list of all control variables for selected DCU.

Unit Outputs button displays list of control functions that have in its name the word "output" for selected DCU.

Unit Inputs button displays list of control functions that have in its name the word "input" for selected DCU.

Find Function button displays for selected DCU list of control functions that have in its name the string of characters specified in editable text field.

Find Variable button displays for selected DCU list of control variables that have in its name the string of characters specified in editable text field.

Vars/Fun.Outs Value/State from DCU button displays actual values and states (alarm, override) for selected control variables. If control functions are selected instead, outputs of these functions are considered to be displayed. To display the values and states a window with a list of selected control variables is opened. To get actual values and states selected on-line DCU is directly requested to fetch the data.

Vars/Fun.Outs Value/State from Rec button does exactly the same operation as the previous button, only to get actual values and states last values of history recording database of the selected on-line DCU is asked.

Vars/Fun.Outs Signal button display signals for selected control variables (function outputs). Designer must specify time span in the form of

- From time and date to time and date or
- Last N of seconds

Values are taken from history recording database of the selected on-line DCU.

Update Parameters With Diagram button allows to update parameters of control functions or control variable or application general parameters without re-loading entire control application. Designer must first modify parameter inside control application diagram, save the diagram and close it. Then he must display list of control functions/variables and select the modified function/variable. Finally he must click the button "Update Parameters With Diagram". The diagram is analyzed, parameters of selected control function/variable are read and sent to DCU. If general parameters of control application are to be modified, the procedure is the same, only designer need to display application general parameters of DCU before clicking the button "Update Parameters With Diagram". In the case of control variable parameters, override is not updated.

Process Experiment Tools selector shows available control design and optimization experimental tools for working with control variables, such as process experiment (applying excitations, recording responses), controller design, tuning and optimization, process model identification, etc.

Simulation check-box enable/disable simulation of the process experiment tool. If checked, process experiment tools are launched in simulation and not in real-time. Experiment tools running in simulation means that, when experiment tool is launched and some recorded process signals are required, user is asked to provide them.

Launch button starts selected experimental tool. The basic tool, which is a foundation for any more sophisticated tools is "Process Experiment Procedure". It allows to apply excitation signals onto selected control variables (dynamic override)

as well as constant overrides and recording process responses in the meantime.

3.5 DCU Design Toolbox – Process experiment tools

This section describes some of design tools available in the Process Experiment Tools selector.

Process Experiment Procedure launching will open an interactive window of the tool, see Figure 9. Inside the window there are 3 lists to fill in. To add a control variable to any of the lists display list of control variables in the main window and select variable(s) to be added to the list. Then in the tool window click "Add selection" button of the appropriate list. The selected variables will appear in the corresponding list. To modify timings and values for each selected control variable select control variable in the tool window and actual timings and values will appear below the list. Modify any value and click "Set" button.

List at the top displays selected control variables on which a constant override will be applied. List in the middle displays control variable on which a dynamic override will be applied, i.e. an excitation signal will be sent onto the control variable. List at the bottom shows to-be-recorded control variable values.

Ziegler-Nichols PID Tuning. This experimental procedure will propose coefficients of PID controller based on Ziegler-Nichols rules. The procedure applies a step on controller output in open loop and reads and analyzes process response. Note that control loop is opened during the procedure experiment. This may not be possible sometimes, another tuning technique must be used in such case. Process must be settled down (not in a transition) when launching the PID tuning procedure. Otherwise, the results will be biased.

First, a PID controller function must be selected from list of on-line DCU elements. Then, the launch button may be pressed. User is asked to chose:

Amplitude of the step that will be applied to controller output.

Estimated process response time (settling time). User estimate how long will take the controller process value (temperature, pressure,...) to move from one steady state to another if a step is applied to actuator. The value may be overestimated, but never underestimated.

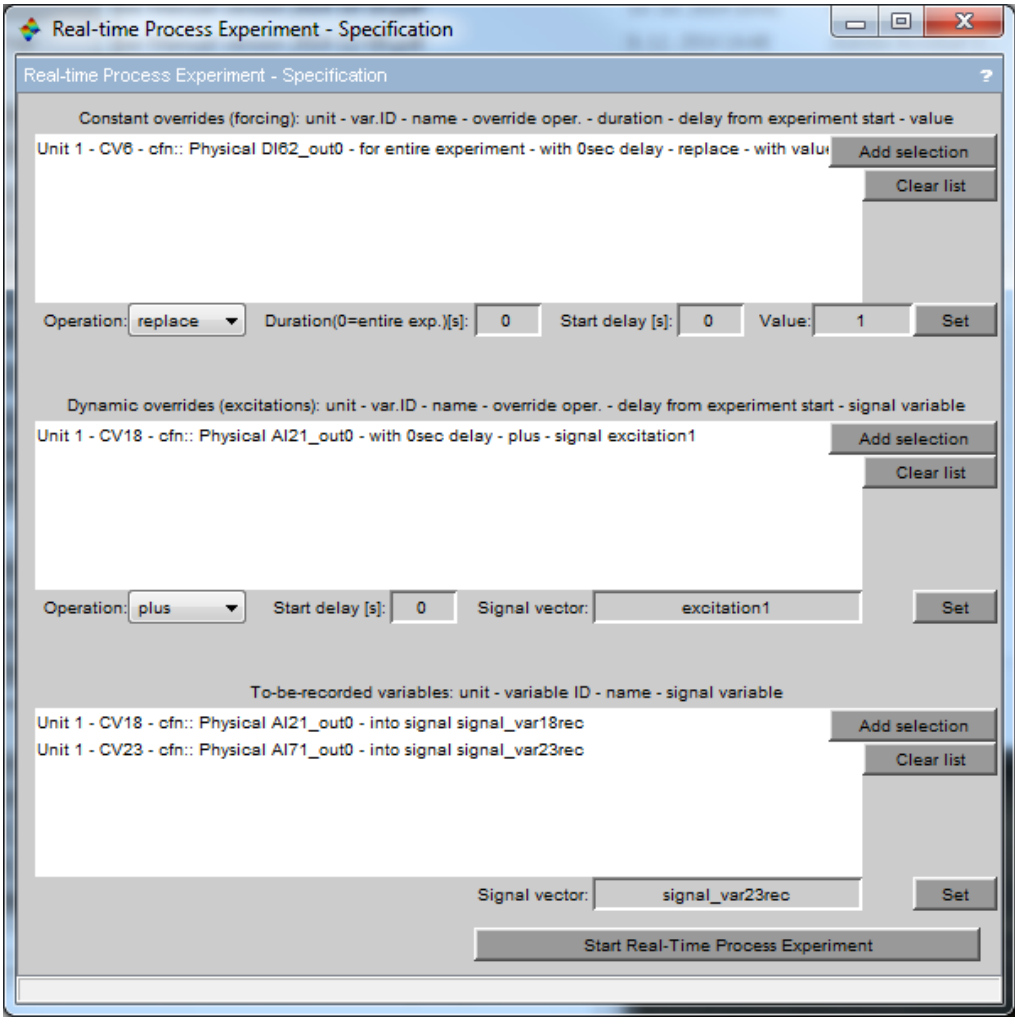


Figure 9: Real-time process experiment window.

3.6 DCU Design Toolbox visualization setup

Standard DCU Control System visualization may be defined/redefined/updated inside DCU Design Toolbox. Main menu item 'Visualization Server Setup ' gives access to all necessary elements :

- **Server Data Recording –** Opens window for definition of visualization server process recording parameters. See Figure 10 and related section for details.
- **Visualization Structure** - Opens window for definition of user visualization pages. See Figure 11 and related section for details.
- **Alarm Watch Guard** - Opens window for definition of alarm watch guards Alarm watch guard sends email/printing a page in case of some defined event or alarm. See Figure 13 and related section for details.
- **Server Virtual I/O** - Opens window for definition of real-time virtual inputs/output functions. Virtual input/output is process value received/transmitted from/to DCU controllers by server. See Figure 14 and

related section for details.

- **Operator Panels** – Opens window for editing defined operator panels.
- **Batch Servers** – Opens window for definition and modification of batch servers.
- **Set Visualization Server** – Updates visualization server settings according to parameters set in the definition windows mentioned above (data recording, visualization, alarm watch guard, real-time I/O windows). User is first asked if local server (designer computer) of remote server (another computer/server on local or global network) should be updated. In case of remote server user is asked for IP address of the server, TCP port of MySQL database on this server, login and password to MySQL database. Remote access requires some specific settings of designer computer and visualization server, see section 2.14Setting remote connection with User Command Center.
- **Open Visualization Web-Pages** – Tries to open main web page of project visualization. User is asked if local or remote visualization should be opened. In case of remote visualization, user specifies visualization server URL.

For more details on deploying visualization server see section 5DCU Visualization .

3.6.1 Data recording window

Server data recording parameters of actual project are defined in this window. The parameters defines communication and data recording behavior of visualization server or designer computer with respect to control units defined in the project.

Control Units list enlists all units of actual project. By selecting one control unit from the list its parameters are displayed below. Namely:

Communication settings - IP address, TCP port, server history recording TCP port (port on which server listens for data from DCU), etc. For details on communication parameters see section 2.7Step 6 - Set DCU communication settings. By default the communication settings are taken from translated control applications.

Time Synchronization with Server – Synchronization Period defines time period in hours for regular synchronization of time and date between DCU controller and visualization server.

Process Data Recording Time Spans section defines how for long will be recorded data kept in database. User may define time span for DCU events, time span for DCU state and time span for control variables values. Time span for control variables may be defined once for all control variables for specifically for each control variable (Specific Time Span check-box and list below).

Note: *Longer time spans means more data in database and in consequence more performance required from server by database system. Recording to large databases may slow other processes running on the server/designer computer. Amount of data recorded to database depends on time span and history recording sampling time defined for each control variable. Examples of time span, history recording settings, amount of recorded data:*

Large control application for one DCU56IO defined with

- *one sampling time Ts=1sec*

- *56 used physical I/Os*
- *about 300 control functions, and 300 control variables*
- *for physical I/Os control variables => monitoring sampling = 1sec, history recording sampling = 1sec*
- *for remaining control variables => monitoring sampling = 1sec, history recording sampling = NONE*
- *time span for all control variables 24h*

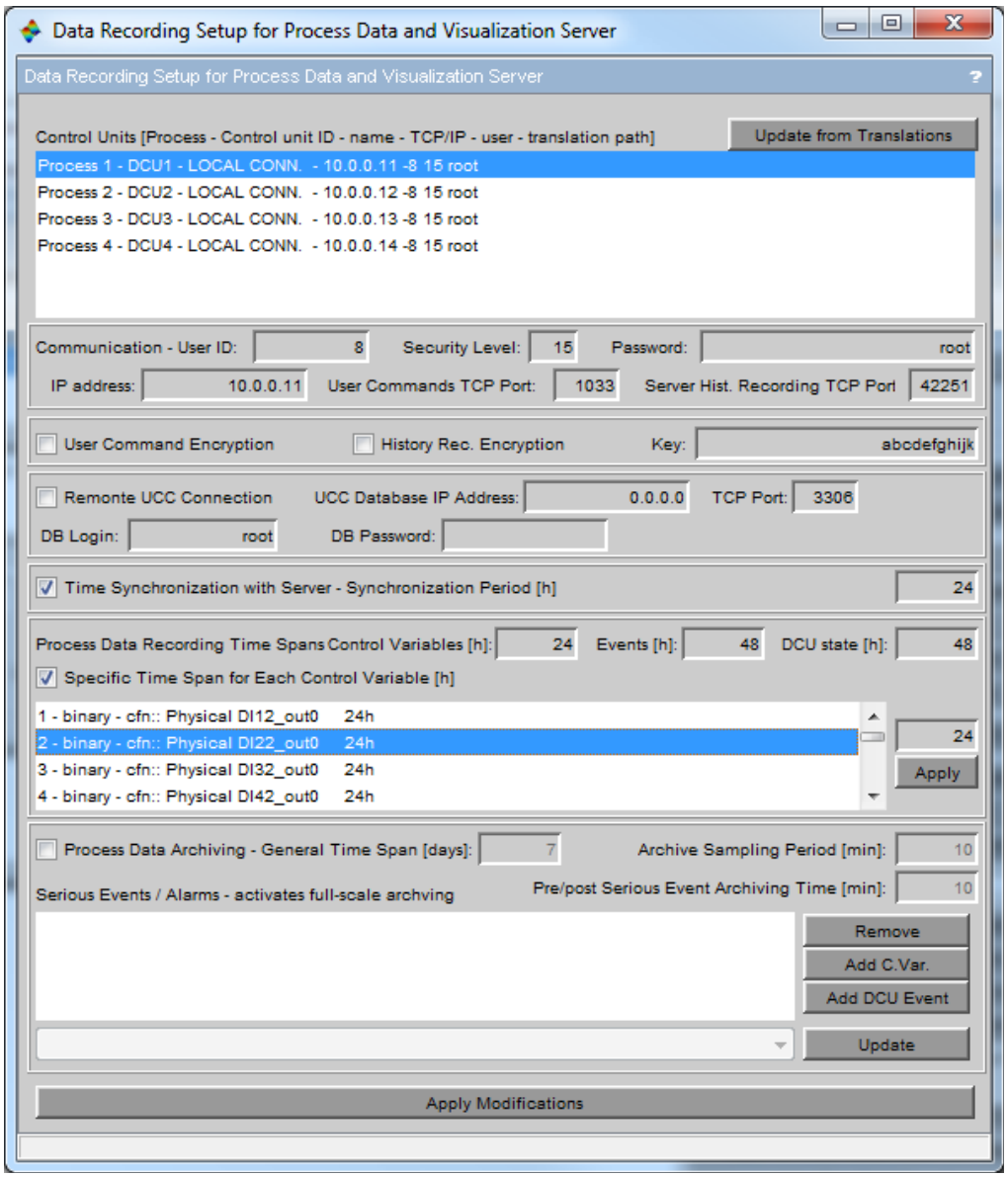


Figure 10: Data recording window.

Monitoring started (monitoring is set for each control variable with sampling 1sec):

Recording rate =

*(56IO control variables + 300 control variables)*60sec = 21360 records / min*

Data rate =

*21360 records/min * 30 Bytes for each record = 640800B/min = 625,8kB/min*

Amount of kept data =

*21360records/min * 24hours *60min = 922752000B = 901MB*

History recording started (history recording is set only for physical I/O control variable with sampling 1sec):

Recording rate =

*56IO control variables *60sec = 3360 records / min*

Data rate =

*3360 records/min * 30 Bytes for each record = 100800B/min = 98,4kB/min*

Amount of kept data =

*3360records/min * 24hours *60min = 4838400B = 4,7MB*

Process data Archiving section at the bottom of the window on Figure 10 is enabled by checking its check-box. Time span and sampling period for archiving of control variables can be set. Serious Events/Alarms list contains control variable alarms and/or DCU events that trigger full-scale archiving. Full scale archiving means that control variables are not archived with archiving sampling period but with control variable original history recording/monitoring sampling.

Pre/post Serious Event Archiving Time indicates how long before and how long after a serious event/alarm occurrence should continue full-scale archiving.

Apply Modification button saves parameters from edit text boxes to project variables

3.6.2 Visualization window

Standard web-page/operator panel visualization is defined in Visualization window, see Figure 11. In Visualization window designer defines:

- Visualization folder path
- Visualization user accounts
- Process views and its grouping – web pages displaying process or control system
- Event views and its grouping – web pages displaying alarms and events of control system. Alarm is occurrence of an excessive value in a process control variable. Alarms are defined in control application diagram together with other control variable properties, see section 4.3Control variable. Event is occurrence of som specific state inside DCU controller. Complete list of DCU events is in section 11Annex C: List of DCU events and alarms.
- Global alarm indication. A variable indicating global alarm of the control system.

Visualization Pages Path text box indicates designers path to visualization files. By default , they are placed in 'Visualization' folder inside project folder.

Visualization Users section defines accounts of visualization users. Each user is

defined by:

- ID of visualization user
- Login of visualization user
- Password of visualization user
- ID number of user defined inside DCU, visualization will communicate with DCU via this DCU user account
- Security level of DCU user, max. is 15 this gives access to all parameters
- Password of DCU user
- Show Control System yes(1) / no(0). If enabled, it is supposed, that user is designer. Visualization gives him access to control system menu where complete control system is displayed and can be modified.
- Allow Event Acknowledgment yes(1) / no(0) indicates if user has right to acknowledge alarms.
- Show Process View Groups – Group 1 … Group 10 yes(1) / no(0) defines which process view groups (see below) will be displayed to this user.

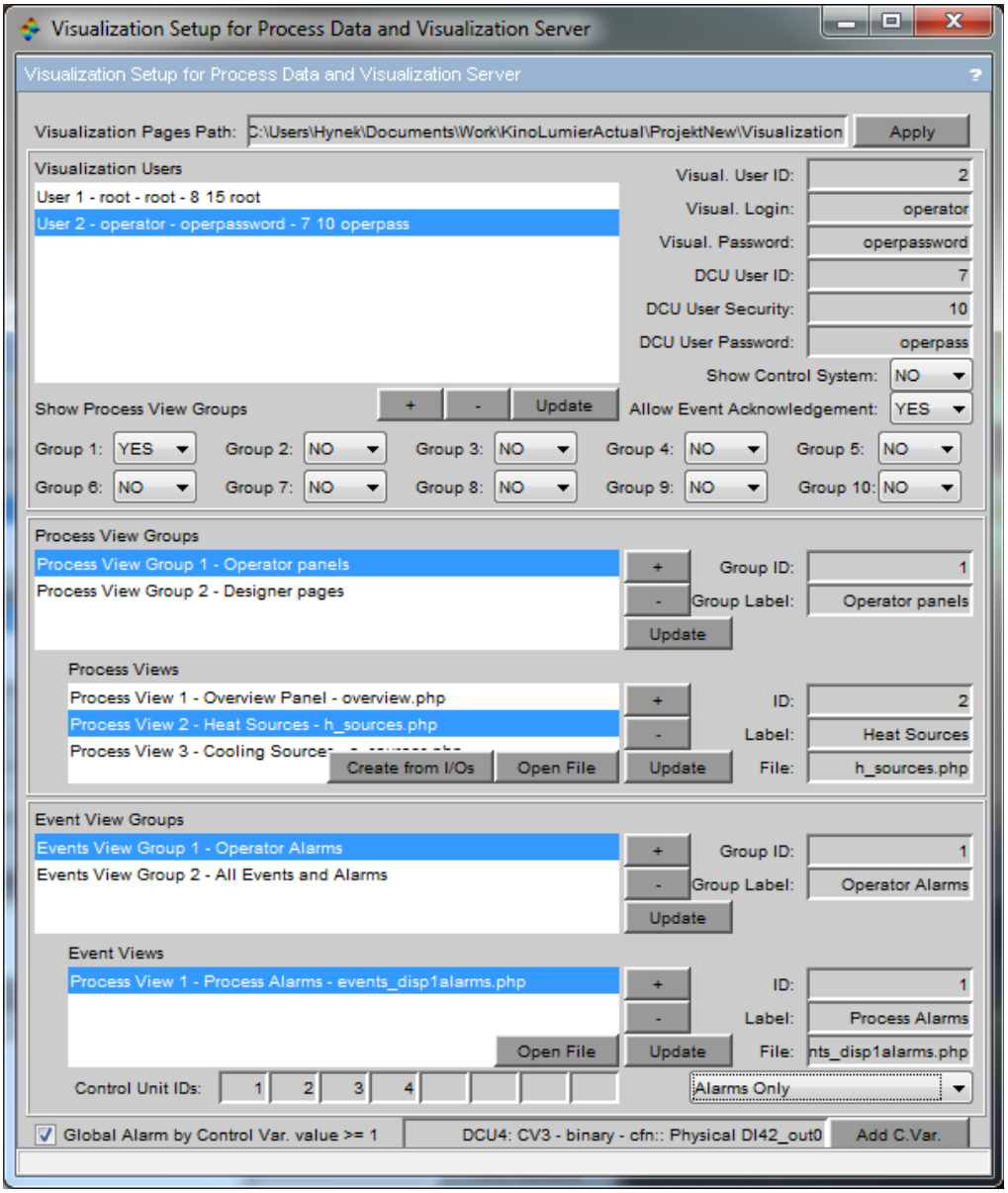


Figure 11: Visualization definition window.

Process View Groups section (middle part of Figure 11) defines groups of process views and inside each group its process views. One process view is one web-page displaying a part of controlled process, or control system. One group is a set of views with a common property. For example group that assemble views for process operator, group for control system designer with detailed technology diagrams, group of a specific sub-process, etc. The following two lists display process view groups and pages:

- **Process View Groups list** displays defined groups. A group is defined by its identification number (ID) that must be unique among groups and group label. The label is a text string displayed in visualization sub-menu to access group views, when PROCESS is selected (compare Figure 11 and 12). A group can be added (button '+'), removed (button '-') or modified (edit

selected group in the text fields and press the button 'Update'.

- **Process Views list** displays views of actually selected group. A view is defined by its identification number (ID) that must be unique inside group, view label, and process view file. The label is a text string displayed in visualization menu selection to access the view, compare Figure 11 and 12. The process view file is a file inside Visualization folder that defines web-page appearance. A view can be added (button '+'), removed (button '-') or modified (edit selected group in the text fields and press the button 'Update'. Process view file can be opened by pressing 'Open File' button.

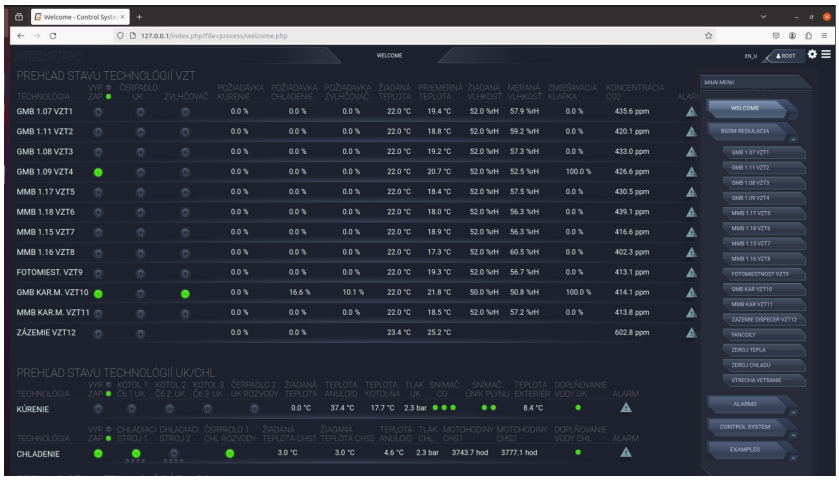


Figure 12: Example of process web visualization view groups.

File that defines process view web-page must be created by user. Because it is PHP script it has extension .php. Example of this file 'cpanel_1.php' is in Visualization folder. 'Create from I/Os' button inside Process View list provide simple mean for quick automatic creation of process view page. The page is created from inputs and outputs of selected controller. By clicking the button user is asked for ID number of controller and for name of created web-page file. For detail on building custom process view web-page see section 5.3Building process view page.

Event View Groups section (bottom part of Figure 11) defines groups of event views and inside each group appropriate event views. Event view is one web-page displaying events and/or alarms of control system. Group is again a set of views with a common property. For example group that assemble views for process operator, or group for control system designer. The following two lists display event view groups and pages:

- **Event View Groups list** displays defined groups. Similarly to process view groups, an event view group is defined by its identification number (ID) that must be unique among groups and group label. The label is a text string displayed in visualization sub-menu to access group views, when ALARM/EVENT is selected. A group can be added (button '+'), removed (button '-') or modified (edit selected group in the text fields and press the

button 'Update'.

- **Event Views list** displays views of actually selected group. A view is defined by its identification number that must be unique inside group, view label, event view file, controller ID numbers, and selection if 'alarms and events' or 'alarms only' will be displayed. The label is a text string displayed in visualization menu selection to access the view. The event view file is a file inside Visualization folder that defines web-page appearance. User does not need to create a new web page file. Default event view file "events_disp1alarms.php" may be used to display list of alarms. Control unit ID numbers indicates for which DCU controllers will be displayed alarms and/or events. Alarms Only/Alarms and Events selection defines if the page displays only process control alarms or alarms and DCU controller events. A view can be added (button '+'), removed (button '-') or modified (edit selected group in the text fields and press the button 'Update'.

Global Alarm by Control Variable check-box is at the bottom of Visualization window (see Figure 11). It enables/disables global alarm indication by a control variable value. Global alarm indication is a process status icon displayed on the visualization web-page, see ALARM icon in Figure 12. Normally, the global alarm is indicated if an unacknowledged alarm is present in control system database. By checking the check-box and selecting a global alarm control variable, the global alarm is indicated if value of selected control variable is equal or greater than 1.

3.6.3 Alarm watch guard window

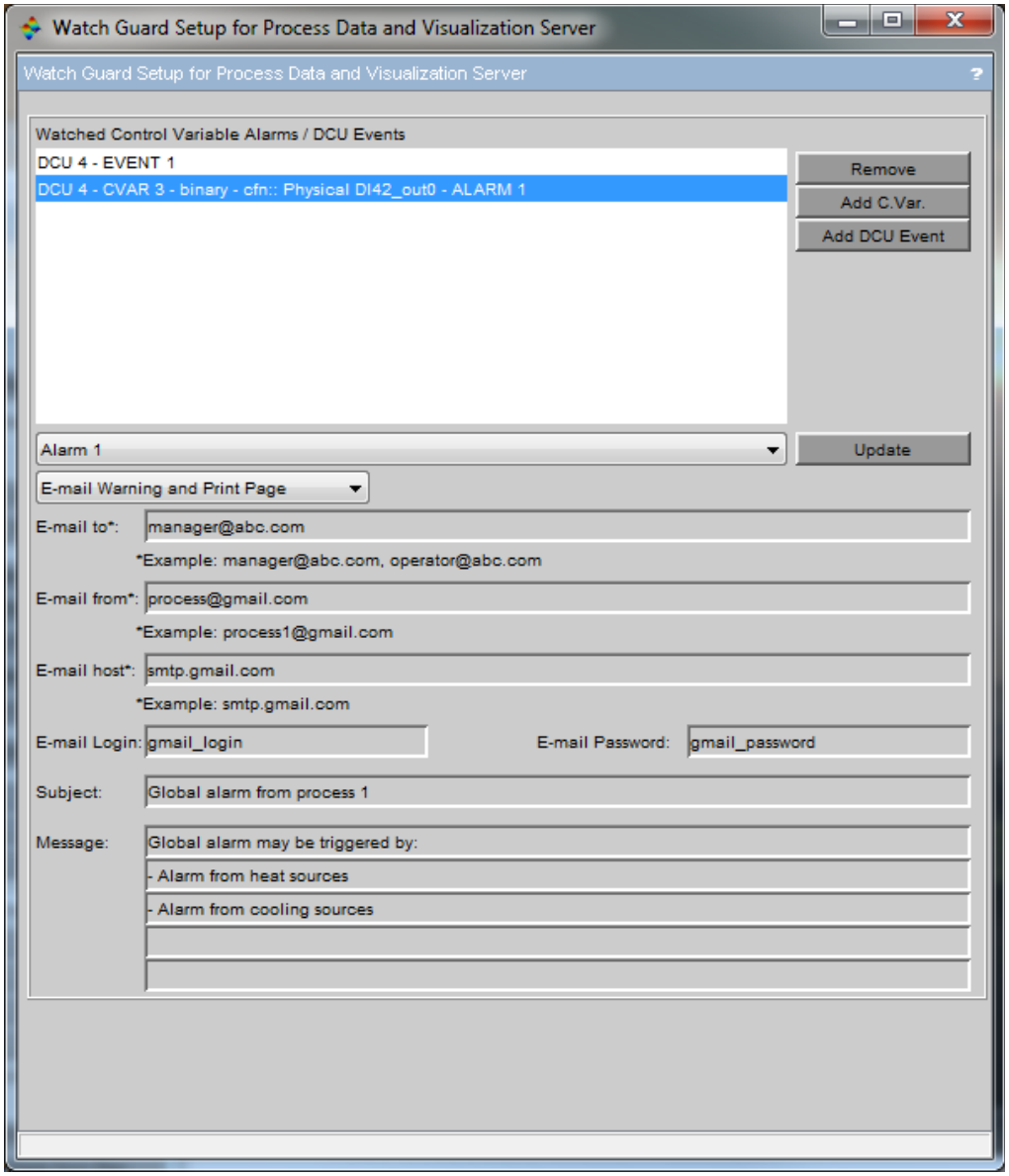


Figure 13: Alarm watch guard definition window.

Alarm watch guard is a service that is waiting for a specific process alarm or DCU controller event. When the alarm/event occurs, the service sends email and/or print alarm page on default printer. Multiple alarm watch guards may be defined each waiting either for a DCU event or for a control variable alarm value. Figure 13 shows window for alarm watch guards definitions. A list of alarm watch guards is at the top of the window. Below, user defines selected watch guard. On the right side of the list, buttons are available to add or remove selected watch guard. User can add watch guard of either control variable alarm (control variable must be selected in the main window of DCU Design Tool) or of DCU controller event (DCU must be selected in the main window of DCU Design Tool).

3.6.4 Server virtual I/O window

Real-time virtual inputs and outputs of visualization server are defined in this window, see Figure 14. Virtual input is a service that receives real-time process

values from network send by a sender using real-time UDP protocol, see 15. Sender is typically a DCU controller, server, or operator panel that has defined a corresponding virtual output (see sections 4.4.5 and 4.4.6). Virtual output is a service that sends values using real-time UDP protocol to a receiver (DCU controller). Received values as well as values to send are stored in database table "rtVirtualIO" created at the time of starting virtual I/O services. The table is created inside visualization configuration database "dcuUsercommConfig".

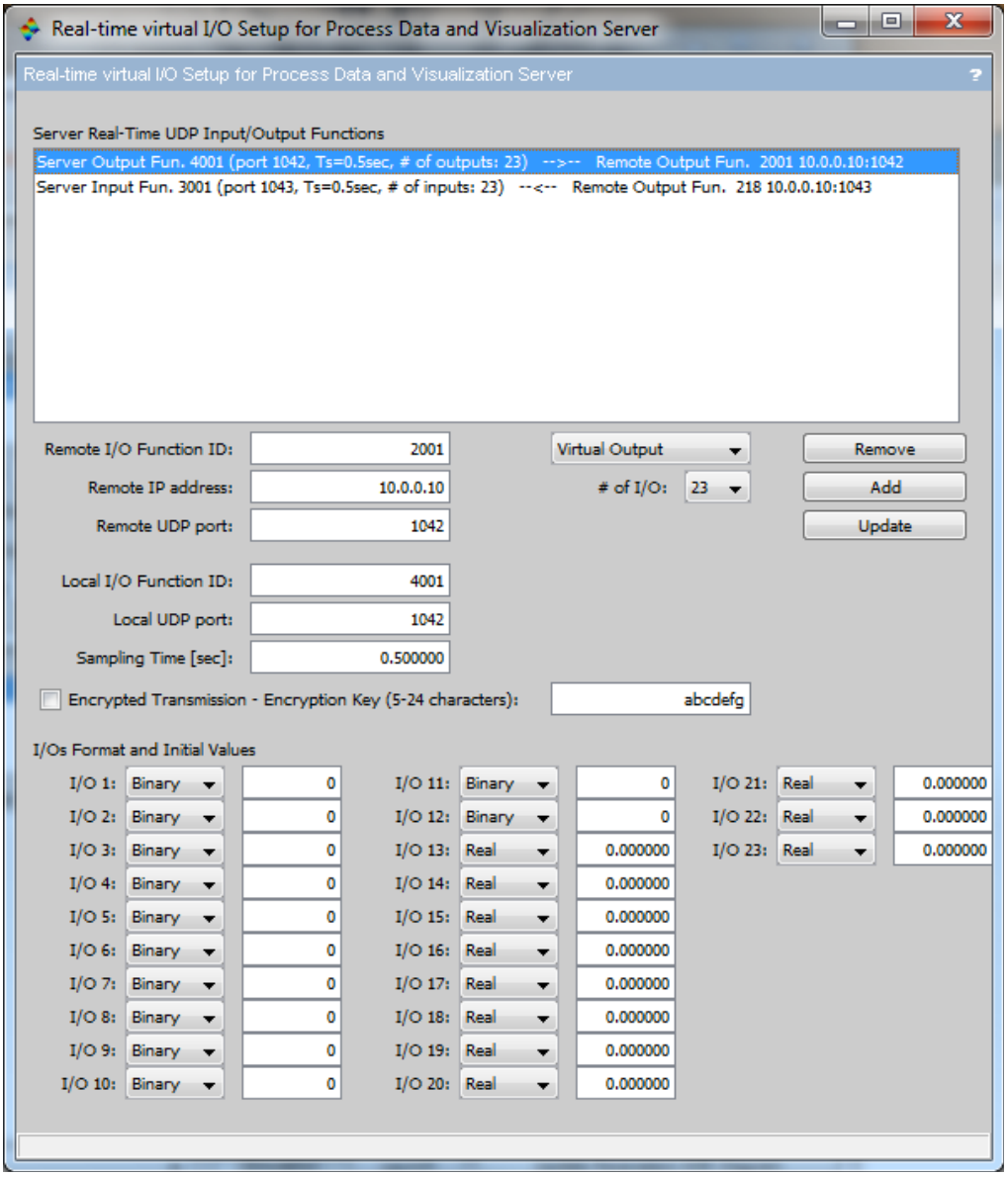


Figure 14: Real-time virtual I/O definition window

Each virtual input or output defined in the visualization server is considered to be a control function as is the case in DCU controllers (see sections 4.4.5 and 4.4.6). Translation of DCU control application add automatically server virtual I/O corresponding to virtual I/O defined in DCU control application. For example, if designer place virtual input to control application and set its IP address to history recording server IP (10.0.0.2), translation of this application add server virtual output

that will communicate with this input.

List of server virtual inputs/outputs is at the top of the window. Buttons Remove/Add/Update at the bottom of the list allows to remove selected I/O, add a new I/O, update actual settings of selected I/O respectively. Remaining elements below the list displays actual settings for selected virtual I/O control function. Each virtual I/O function may have up to 23 input/output ports. For each I/O port, format and initial value are defined.

Note: *Each virtual I/O function has to have a unique UDP port.*

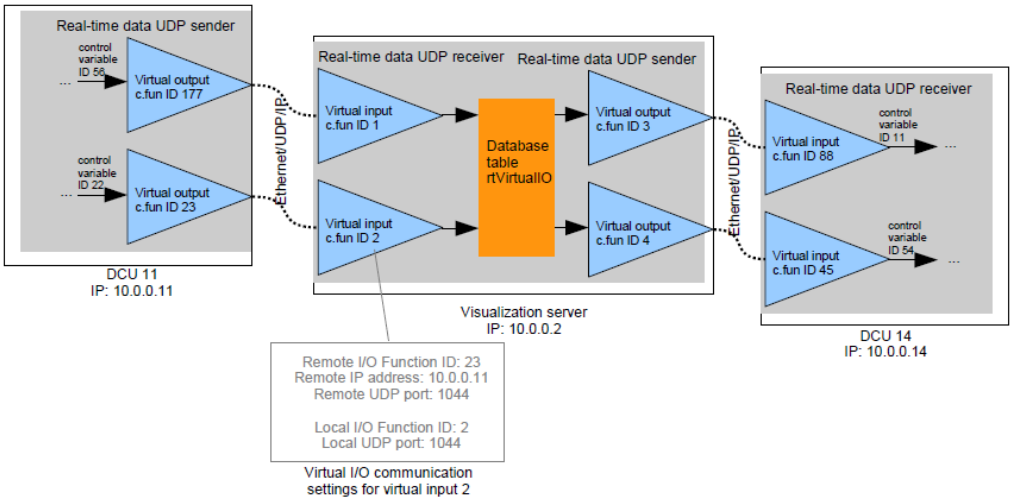


Figure 15: Real-time virtual input/output communication example.

3.6.5 Operator Panels

Operator panels are edited in this window, see 17. To create/add operator panel into the project, designer has to place one virtual input control function into a DCU control application and place into its name “OP” string (see IO control function library). IP address defined in this virtual input defines IP address of the new operator panel. After translation of the modified diagram, the new operator panel appears in the operator panel window selector (upper left corner of the window). The name of the panel corresponds to the virtual input control function name. The main feature of operator panel is its fast interaction with DCU control unit(s). Critical data are communicated between DCU and panel using RTUDP Virtual I/O functions (UDP/IP streams of real-time data). Remaining process data are streamed to the panel from server. Process is displayed and accessed using visualization application installed on operational panel. This application is more responsive than web-based visualization, see 16. Operator panel has the same software configuration as a visualization server. Any panel may be switched to server mode. By clicking the button “Set Panel Visualization”, computer is set to panel mode.

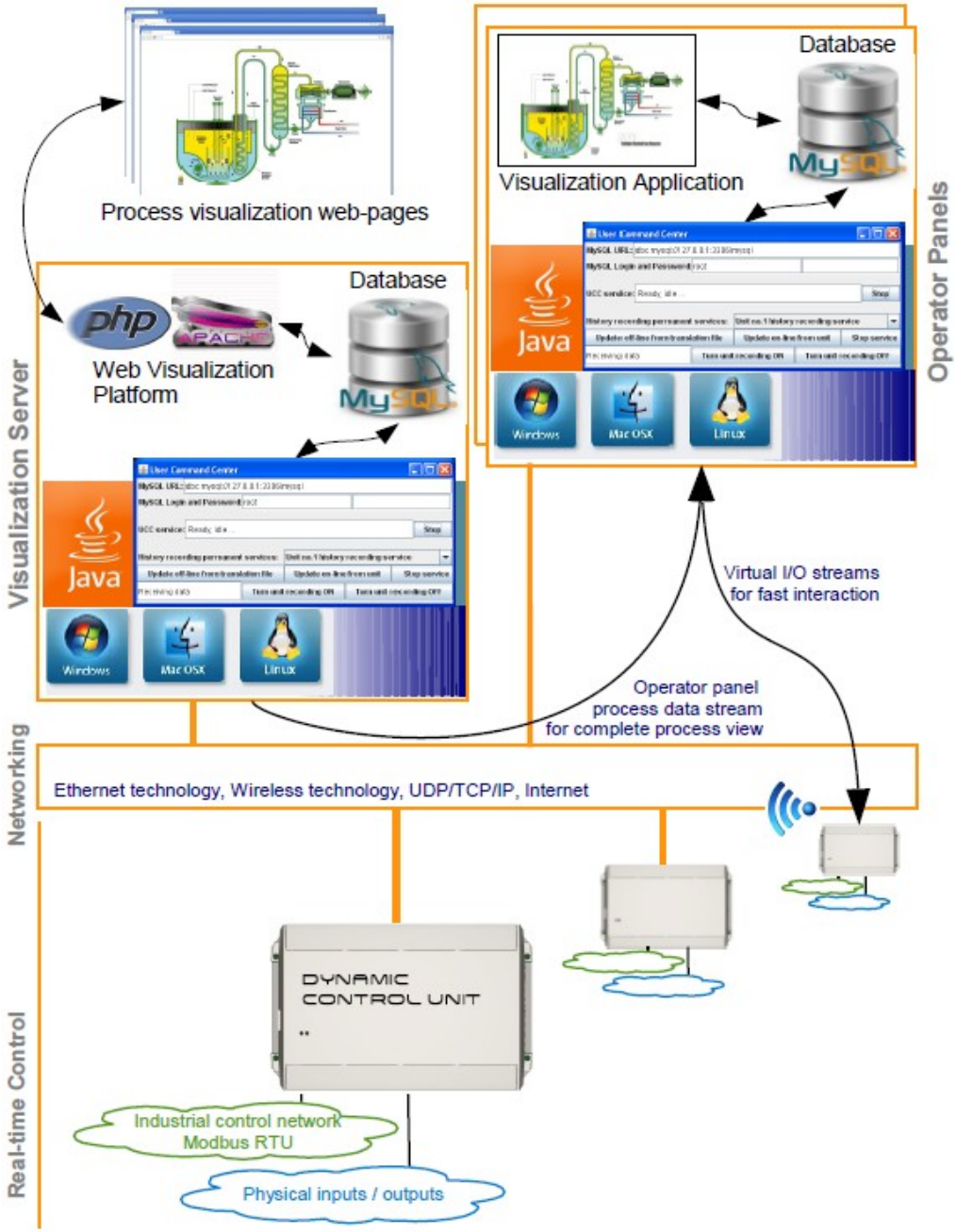


Figure 16: Structure of DCU control system with operator panels.

Upper part of the operator panel window in 17 defines general parameters of the operator panel. Namely:

- Operator panel name check-box (upper left corer) indicates selected operator panel.
- Panel IP address as defined in virtual input control function that defines the panel.
- Server → Operator panel streaming mode defines if server will send history

recording samples (generally only I/O values) or monitoring samples (all control application variables)

- Stream sampling defines minimal sampling period of the streamed process data.
- Stream recording time span defines how old data will be kept in the operator panel. Zero means that only actual values are kept.
- Streamed DCU IDs defines ID numbers of all units whose process values will be streamed from server into operator panel.
- Encrypted Streaming check-box and encryption key enable/disabled encryption of the streamed process data and defines used encryption key.
- Stream UDP port defines UDP port of the stream.

Lower part of the operator panels window starting with list of Panel Real-Time UDP Input/Output Functions defines virtual I/O functions of the operator panel. Automatically, there is one virtual output that corresponds to virtual input defined inside DCU needed for creation of the operator panel. Other virtual I/O functions may be defined to connect panel to other virtual I/O functions. For details on definition of virtual I/O see section 3.6.4Server virtual I/O window.

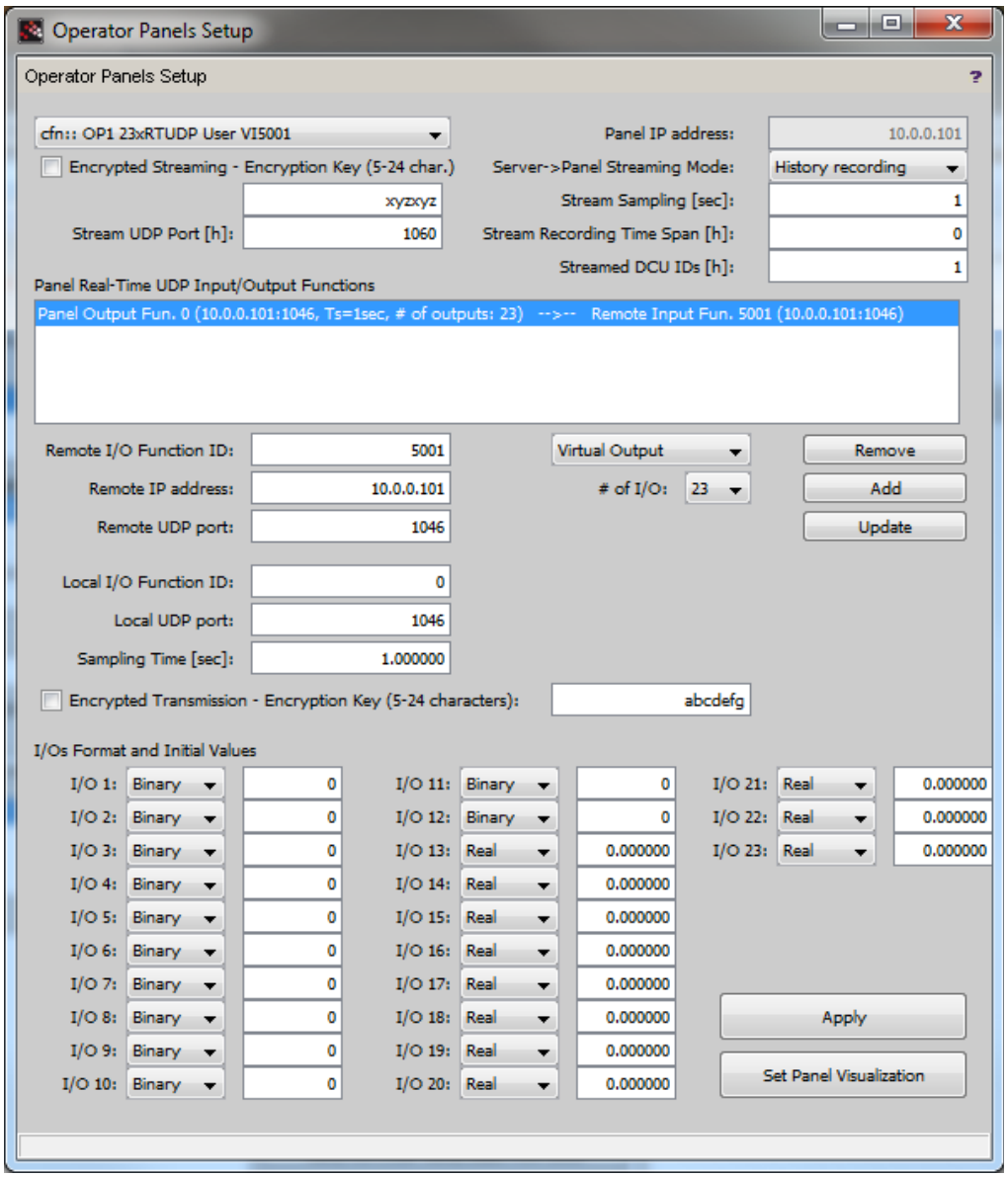


Figure 17: Operator panels window.

3.6.6 Batch servers window

Batch servers may be defined in batch server window. For details on batch servers see section DCU Batch service. In the top of the window, general parameters of the batch server are defined:

- Batch server label and icon image (label and image are defined in one editt boax and they are separated with double-divide '//' string)
- Max. number of parallel phases or batches

- Task output value in case of disabled batch
- Max. number of parameters for one task.

Buttons '+' and '-' on the side of the selector indicating the actual batch server serve for adding and removing batch servers. Button 'Apply' update batch server with values from edit boxes.

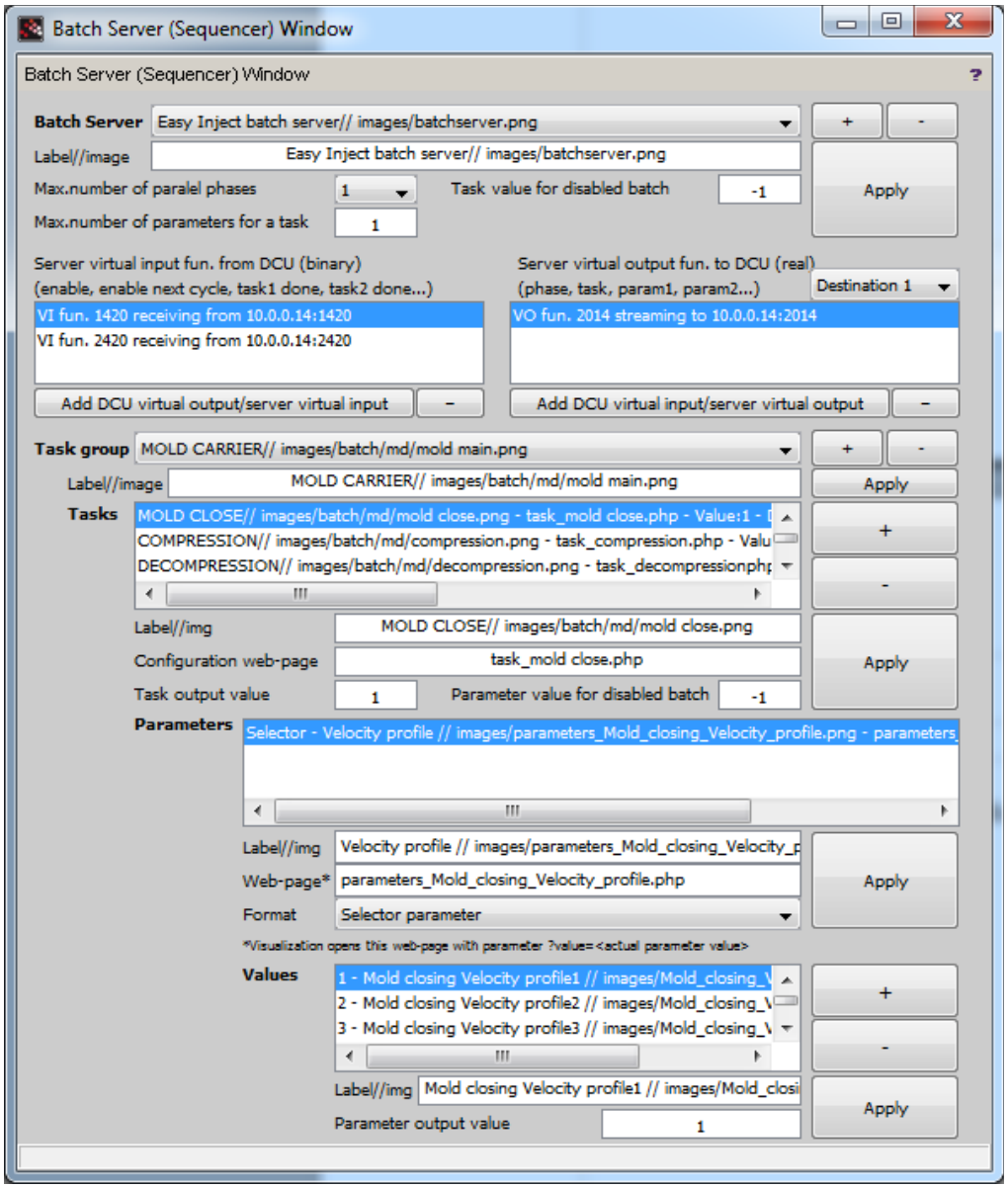


Illustration 18: Batch server window.

Below the top general-settings part are two lists containing visualization server RTUDP input/output functions appropriated to the batch server. To add a server input to the left list open 'Server virtual virtual I/O window', select a server virtual input and click in Bach server window on button 'Add DCU virtual output/server virtual input'. The selected input will appear in the list. To remove inappropriate input select it in list and click on '-' button. Each virtual input function has between 1 and 23

inputs. Sum of all inputs should correspond to the number of required inputs for the batch server: Nin = 2 + 2*Nt, where Nt is number of tasks. The order of inputs corresponds to order of task groups and tasks as they are defined in the window, see 28.

The same procedure may be used for adding/removing server outputs. Number of required outputs is Nout = Npph*(2+Np), where Npph is number of parallel phases and Np is max. number of task parameters.

Below the server inputs/outputs lists are selectors buttons, and edit boxes for definition of tasks groups, tasks, and task parameters. Label/image may be defined to any task group, task, parameter. Each task as well as task parameter may have its own configuration page. Format is additionally defined for parameter. If selector format is chosen, designer may define to-be-selected values its labels and icon images.

4 DCU control application

4.1.1 Overview

Control application is a real-time program for DCU controller. DCU controls process according to loaded control application. Control application is programmed using functional diagram only. The functional diagram contains interconnected functional blocks and some pre-defined parameters. Control application for one DCU is defined inside super-block (group of blocks) which has at the beginning of its name “dcu::” string, see Figure 19. Main control application may contain multiple DCU super-blocks. To edit control application of a DCU, double-click the DCU super-block. The actual diagram of the corresponding super-block opens and user can start add, remove, edit control functions and its connections. All control functions are in library diagram that is in DCU design tool directory. Notice, that control function has always at the beginning of its name the string “cfn::”. It indicates to compiler that it is control function and not some simulation block.

Observing control application diagram, designer sees only DCU super-block, functional blocks inside DCU and its interconnections. Internal structure of control application functional diagram is however more complex, see Figure 20. Each functional block represent control function and control variable (only in case the function has a computed output value). Moreover, the block contains parameter definitions for the control function and for control variable. Finally, entire DCU super-block has parameters that defines general parameters of the control application. These are defined in context of the super-block (Scilab/Xcos) or in mask initialization (Matlab/Simulink).

Control application loaded into DCU consists of the following elements:

- Set of general parameters
- Control functions
- Control variables

General parameters defines general properties of control application such as used sampling times, DCU communication settings for UDP/TCP/IP and Modbus RTU, user accounts, etc. Control functions together with control variables, which are functions outputs or say results represent desired behavior of DCU outputs with respect to DCU inputs.

There is slight difference between the structure of these elements directly present inside DCU and how they are presented (for definition) to control system designer. The reason is to simplify designer's work. In principle, parameters that can be automatically evaluated from the control application functional diagram are not requested to be specifically defined by designer. Also storage of parameters inside DCU is much more compact than inside of the design tools. We will focus on control application from the designer point of view.

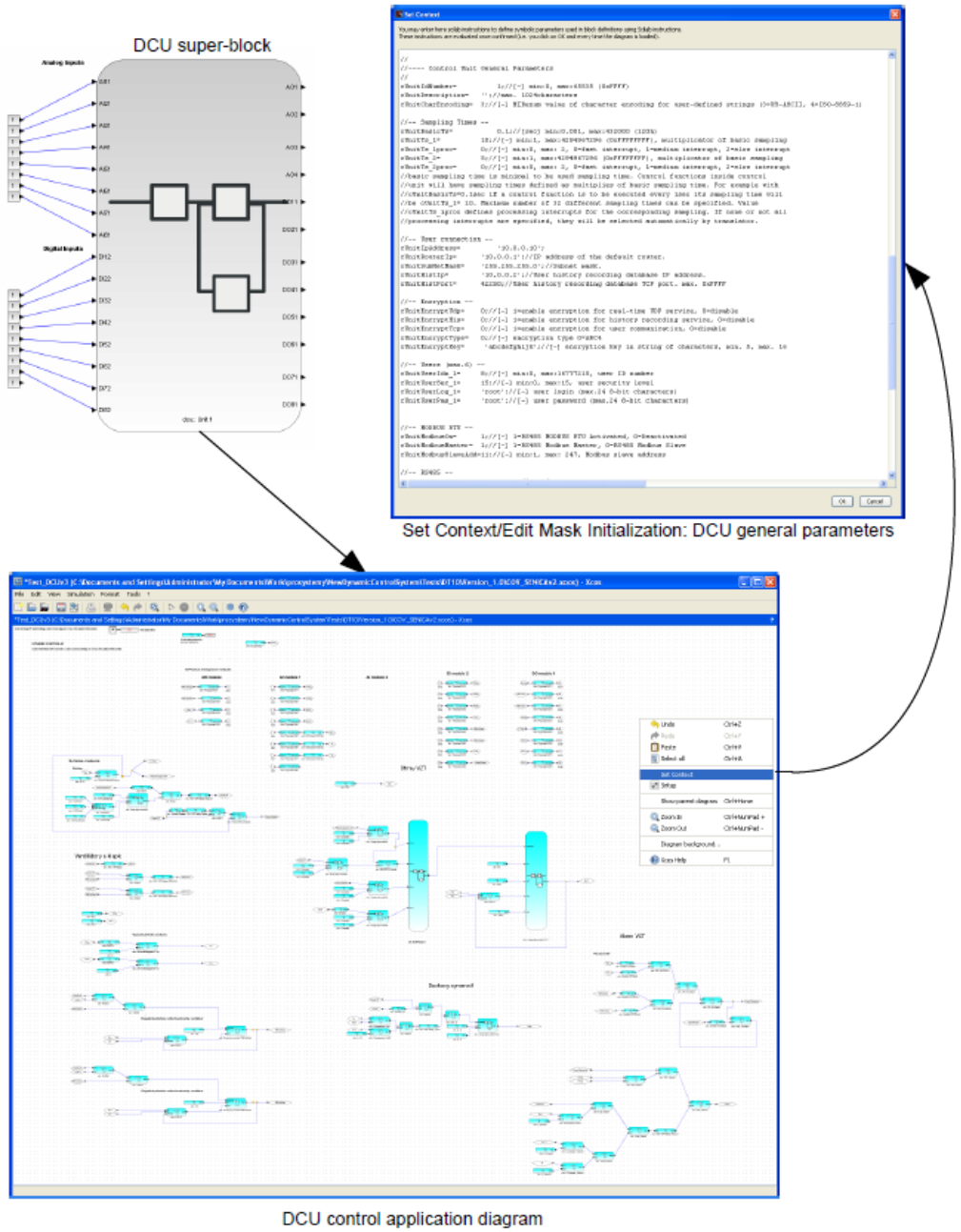


Figure 19: DCU control application diagram.

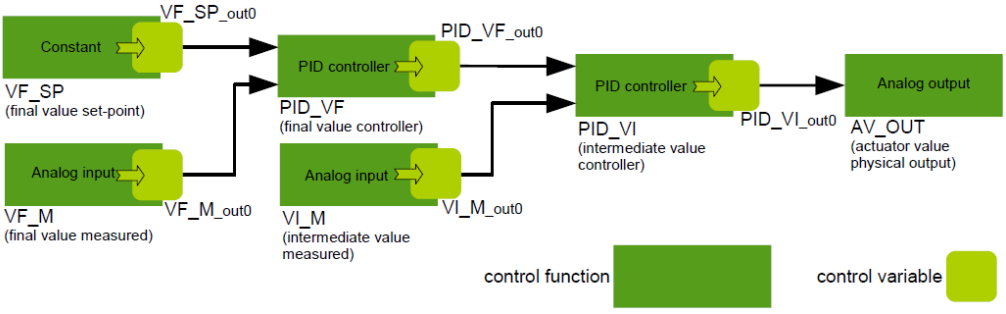


Figure 20: Internal structure of control application functional diagram.

4.2 General parameters

Each general parameter to be defined by designer is stored inside of control application functional diagram in one variable. Name of this variable is searched by a translator when translation of the diagram is invoked. Value of the parameter stored inside functional diagram is in case of a numeric-format parameter defined by a term:

variablename = variablevalue;

In case of a string-format parameter the parameter is defined as follows:

variablename = 'variablestring';

For example, for control unit ID number, the variable name is “cUnitIdNumber”. Its value is defined by the following line:

cUnitIdNumber= 1; : [-] min:0, max:65535 (0xFFFF)

Any text behind “;” character is omitted by the translator. List of all parameters to be defined by designer is below in the following form:

Variable name : default value : range : description

In case of Scilab Xcos diagram, the general parameters are defined in context of the DCU superblock. To access context, open diagram of DCU superblock, click right mouse button over diagram and select “Set Context”. In case of Matlab Simulink diagram, the general parameters are defined inside mask Initialization of DCU subsystem. To access initialization section of DCU, click right mouse button over DCU subsystem block and select “Edit Mask”. When window of Mask editor opens select “Initialization”.

The following sections describes all general parameters of a Dynamic Control Unit.

4.2.1 Unit identification

cUnitIdNumber : 1 : min.0, max.65535 : Control unit ID number. **It must be unique for entire control system project!**

cUnitDescription : " : min. 0, max. 1024characters : Optional description of control application.

cUnitCharEncoding : 3 : min. 3, max. 2999 :[-] MIBenum value of character encoding for user-defined strings (3=US-ASCII, 4=ISO-8859-1).

4.2.2 Sampling times

User may define up to 32 different sampling times to be specified within each control function. Defined sampling times may be distributed up to three separate computational interrupts (computational threads) - fast interrupt, medium interrupt, slow interrupt. The definition of sampling times follows the following rules.

One basic sampling time is defined in seconds, minimal allowed value is about 0.0001sec, maximum is about 432000sec (120h). Sampling times used for control function blocks are then defined as multiples of this basic sampling. Shortest control function sampling time is about 0.002sec. Shorter sampling time is allowed starting from 0.0005sec, however notice, that serving all 16 analog inputs (slowest I/Os of DCU) takes 0.0012sec, serving 8 analog inputs takes about 0.0006sec. And any defined control function sampling time must be always larger than time needed to serve all inputs/outputs.

Note that minimal sampling time for a larger application running in one thread and serving all 56 inputs and outputs is about 13msec (1.2msec serving IO + 10msec evaluation control application). Sampling time for smaller applications serving only some inputs / outputs starts from 1msec. If unsatisfactory, control application execution speed may be optimized by distributing control loops into three computational threads. Loops requiring fast sampling may be placed into fast interrupt thread and remaining loops into medium or slow interrupt thread. Computational thread is specified for each control function sampling time.

cUnitBasicTs : 0.1 :min. 0.0001sec, max. 432000sec (120h) : Basic sampling time in seconds. It is the shortest sampling time that may be used in application. If not all physical analog inputs are used minimum may be 0.001sec or even 0.0005sec.

cUnitTs_x : 1 : min. 1, max. 4294967296 : Sampling time no. x defined as multiplication of basic sampling. For example with cUnitBasicTs=0.1sec if a control function is to be executed every 1sec its sampling time will be cUnitTs_1= 10. Max. 32 sampling times may be defined.

cUnitTs_xproc: 0 : min. 0, max. 2 : Index of computational interrupt (thread) executing sampling time no. x. 0=fast interrupt, 1=medium interrupt, 2=slow interrupt. If some specifically complex computations are included into control application use slower sampling time in slower interrupt for their execution.

The index x in the control function sampling names ranges from 1 up to 32, thus cUnitTs_1 and cUnitTs_1proc defines first control function sampling time, cUnitTs_2 and cUnitTs_2proc second one, etc. The name of control function sampling (cUnitTs_1, cUnitTs_2,...) is to be placed into Sampling time field of control function parameters window.

cUnitRecordingTs : [sec] general history recording sampling time. Replaces -1 value in history recording definitions of control variable.

cUnitMonitoringTs : [sec] general monitoring sampling time. Replaces -1 value in monitoring definitions of control variable.

4.2.3 Connection settings

cUnitIpAddress : '10.0.0.10' : any IP address : IP address of control unit. It is defined as a string of characters.

cUnitRouterIp : '10.0.0.1' : any IP address : IP address of the default router. It is defined as a string of characters.

cUnitSubNetMask : '255.255.255.0' : any IP address mask : Subnet mask. It is defined as a string of characters.

cUnitHistIp : '10.0.0.2' : any IP address : User history recording server IP address. It is defined as a string of characters.

cUnitHistPort : 42250 : min. 0, max. 65535 : User history recording server listening TCP port. Port that is listened inside UCC that is running on a visualization computer.

4.2.4 Encryption

Actually, only ARC4 (RC4) serial cipher technique is implemented for encrypted communication. If the encryption is enabled inside DCU, the encryption must be set appropriately also on the side of computer inside UCC history recording service or inside DCU design tool.

cUnitEncryptUdp: 0 : min. 0, max. 1 : Enable/disable encryption of real-time UDP communication. 1=enable encryption for real-time UDP service, 0=disable.

cUnitEncryptHis : 0 : min. 0, max. 1 : Enable/disable encryption of history recording. 1=enable encryption for history recording service, 0=disable

cUnitEncryptTcp : 0 : min. 0, max. 1 : Enable/disable encryption of TCP/IP communication (user commands). 1=enable encryption for user communication, 0=disable

cUnitEncryptType : 0 : min. 0, max. 0 : Encryption type, 0=ARC4

cUnitEncryptKey : 'abcdefghijk' : any character string of max 24 1-byte characters : Encryption key in string of characters. It is advised to use between 5 and 16 characters for ARC4 encryption.

4.2.5 Users

Up to 6 user access accounts may be defined inside DCU. These accounts serve for defining access password and security level of the access. Security level ranges from lowest (0) up to highest (15). To modify control application general parameters, real-time system parameters, control function parameters, or load control application level 15 access is required. On the other hand, user may defined access level for modification, reading, or overriding (forcing) control variable value. **At least one account must have security level 15!**

cUnitUserIdn_x : 8 : min. 0, max. 16777215 : User ID number for user number x, max. 6 users may be defined inside DCU.

cUnitUserSec_x : 15 : min. 0, max. 15 : Security level of user number x, lowest security level is 0, highest is 15. At least one user must have security level 15 to have access real-time parameters and to general control application parameters.

cUnitUserLog_x : 'root' : any character string of max. 24 1-byte characters : User login.

cUnitUserPas_x : 'root' : any character string of max. 24 1-byte characters : User password.

The index x in the names ranges from 1 up to 6, thus cUnitUserIdn_1, cUnitUserSec_1, cUnitUserLog_1, and cUnitUserPas_1 define first user account.

4.2.6 Modbus RTU

Modbus RTU protocol communicates over RS485 serial line. DCU may be set as Modbus master or slave.

cUnitModbusOn : 1 : min. 0, max. 1 : Enable/disable modbus RTU. 1=RS485 MODBUS RTU enabled, 0=disabled

cUnitModbusMaster : 1 : min.0, max. 1 : Master/slave setting for DCU. 1=Modbus master, 0=Modbus slave.

cUnitModbusSlaveAdd : 11 : min. 1, max. 247 : Modbus slave address.

4.2.7 RS485 serial link

cUnitRS485BaudRate : 19200 : min. 96, max. 50000000 : Baud-rate of RS485 serial link [bit/sec].

cUnitRS485CharLen : 3 : min.0, max.3 : character length 0=5bits, 1=6bits, 2=7bits, 3=8bits

cUnitRS485ParityBit : 0 : min. 0, max.4(6): Parity bit setting. 0=even parity, 1=odd parity, 2=parity forced to 0 (space), 3=parity forced to 1 (Mark), 4=no parity, 6=multidrop.

cUnitRS485StopBit : 0 : min. 0, max. 2 : Stop bit setting. 0=1stop bit, 1=1.5stop bits, 2=2stop bits.

cUnitRS485Order : 0 : min. 0, max. 1 : Bit order setting. 0=least significant bit first, 1=most significant bit first.

4.2.8 Hardware-in-the-loop modes settings (discontinued)

cUnitPcIp : '10.0.0.2' : any IP address: For on-PC control mode, IP address of the PC controlling process. In case of on-chip simulation mode, the value defines IP address of the PC simulating process.

cUnitPcPort : 1038 : min. 0, max. 65535 : For on-PC control mode, local port of controlling PC for RT UDP communication. In case of on-chip simulation mode, the value defines local port for TCP communication of the simulating PC.

cUnitPcTiming : 1 : min. 1, max. 31(4294967296) : For on-PC control mode, sampling time (index into sampling time table) of physical IO data transmit. In case of on-chip simulation mode, the value defines maximal waiting time for AI/DI simulated data in sec.

cUnitPcControl : 138 : min. 0, max. 1024 : For on-PC control mode, remote control function ID number where to send actual I/O via real-time UDP/IP protocol.

cUnitOnChipSimCom : 4096 : min. 4096, max. 4096 : For on-chip simulation, user command code for on-chip simulation user PC communication (value should be 4096).

cUnitOnChipUserIdn : 8 : min. 0, max. 16777215 : For on-chip simulation, ID number of user performing on-chip simulation.

cUnitOnChipUserSec : 15 : min. 0, max. 15 : For on-chip simulation, security level of user performing on-chip simulation.

4.2.9 Low-level parameters

cUnitTsSoonTime : 2 : min. 1, max. cUnitBasicTs/PIT interrupt interval : Time interval (in number of PIT's interrupt intervals, one PIT interrupt interval is typically 100μsec) for which it is considered that next time sample will be soon. And so ethernet data transmit will be blocked in main and executed in application interrupt.

cUnitDynOvrdMinTDataCover : 15 : min. 1, max. 4294967296 : Minimum time cover in seconds for received dynamic override data signal to start next data receiving.

cUnitHistRecMaxWait_sec : 1 : min.0, max. 4294967296 : Maximum waiting time (seconds) for user history recording to be sent data. Total time = n seconds + m microseconds.

cUnitHistRecMaxWait_usec : 0 : min.0, max. 4294967296 : Maximum waiting time (microseconds) for user history recording to be sent data. Total time = n seconds + m microseconds.

cUnitHistRecMinBuff : 1 : min. 1, max. 8 : Min. amount of history recording transmit buffers which triggers erasing of actually (re-)transmitted data.

cUnitHistRecMaxTxFail : 5 : min. 1, max. 4294967296 : Maximum allowed number of data transmissions fails before reset connection and new connection opening for user-bus history recording is initiated.

cUnitMaxNOpenConnRetry : 5 : min. 1, max. 4294967296 : Maximum number of opening connection retries before longer waiting period.

cUnitWaitWithoutOpenConnRetry : 10 : min. 1, max. 3600 : Longer waiting time period without opening connection retries after maximum number of retries reached.

cUnitInternalAISampling : 100 : min. 1 max. 1000 : Sampling time for internal AD converter in number of PIT's, it should be longer than 20us (start-up time of AD converter).

4.3 Control variable

Control variable represents element of control application that holds result (output) of a control function evaluation. It keeps one scalar value of actual result (output), previous result value and, if needed in a control function for evaluation, even older results (outputs). Control function that has multiple outputs place each output value to one control function. Control variable values may be recorded into history recording server for monitoring and visualization purposes.

Control functions that use control function output value as input, reads its actual and previous values from appropriate control variable. Control variable value may be overrided (terms “forced” or “set to manual” are sometimes used) by user. In such case, instead of reading control function output values, user-defined value is read by control functions that use the variable as input.

Each control function that has an output must have defined one control variable for one its output. The definition of control variable consists of the following parameters:

cVarOvrOn : 0 : min. 0, max. 2 : Override definition. 0=override off, 1=constant override on, 2=dynamic override on (not used in definitions, only during on-line process experiments).

cVarOvrOper : 1 : min. 1, max. 3, : Override operation. Defines operation with computed output value and override value. 1=replace (override value is used instead of output value), 2=plus (values are added), 3=multiply (values are multiplied).

cVarOvrVal : 0 : - : If constant override is set, value of the override is stored here, if dynamic override is set, IP address of dynamic override source is specified.

cVarDefault : 0 : min. 0, max. 1 : Variable definition type. Setting this parameter to 0 means that next parameters of the variable will be set to default values and so user does not need to define them. Setting to 1 means that parameters are user-defined but only once for all outputs. Setting to 2 means all parameters are defined specifically for each output control variable. For last case, variables are defined as vectors of variables, i.e. in square brackets.

cVarInit: 0 : range of variable format : Initial value of control variable.

cVarRecording : 0 : min. 0, max. 65535 : Sampling time for history recording of the values into database. 0=no recording, N= recording every N-th sample.

cVarMonitor : 10 : min. 0, max. 65535 : sampling time for control design tool monitoring. 0=no recording, N= recording every N-th sample.

cVarSecur : [0,0,0] : min. [0,0,0], max. [15,15,15] : Three security levels for 1) reading value, 2) modifying parameters, 3) override. 0= (guest) up to 15= control engineer.

cVarAlarmOnDelay : 0 : min. 0, max. 65535 : On-delay for alarms in number of samples.

Alarms

Number of alarms is defined by vector sizes of alarm parameters. Empty vector [] means no alarms are defined for control variable.

cVarAlarmHL : [1] : min. 0, max. 1 : Alarms sense. High alarm = 1, low alarm = 0. High alarm means the if variable value is >= then alarm value, alarm is activated. Low alarm means that if variable value is <= then alarm value, alarm is activated.

cVarAlarmOff [1] : min. 0, max. 1 : 1=record also off-alarm event, 0=do not record.

cVarAlarmPriority : [254] : min. 0, max. 255 : Alarm priority level. 0=low-low warning, 1=low warning, …, 254=high-high alarm, 255=out of bound error value.

cVarAlarmValue : [0] : range of variable format : Alarm values.

Note that alarm values for each variable my be set also in visualization platform using "Visualization/process/alarms_definition_server.xml" file, see [8].

Variable description

cVarName defines control variable name. It is set automatically according to control function name as follows: control function name_outx, where x is from 0 up to number of function outputs - 1.

cVarDescription is reserved.

4.4 Control functions

Control function is an elementary piece of control application algorithm. A library of control functions is defined covering all possible needs of process control. The library is implemented within embedded real-time system of DCU. Designer defines control function :

- by type, i.e. by selecting function block from library
- by sampling time (cUnitTs_1, cUnitTs_2 ,...)
- by parameter set which is specific for each function type

Function is additionally defined by ID number of the function. However, this value is defined by translator and not by designer.

Next subsections present each control function and its specific parameters to be defined by user.

Note that there also exists a special function library called, for example, "special_block_functions_dcu_library_3.15.zcos". Functions in this library are not described in the following sections. They are described directly inside the library.

4.4.1 Physical analog input

Input ports

0 Only simulation input port to pass simulated process model analog input into control application.

Output ports

1 real

Description

Function reads electrical signal (0-10V / 0-20mA) from specified analog input terminal, converts the signal value into real number using specified range minimum and maximum.

Function-specific parameters

cFunPar1 : 1 : (1..8) : Terminal position. Position of analog input on AI module terminal.

cFunPar2 : 1 : (1..2) : Module number. Position of AI module where is placed analog input.

cFunPar3 : 0 : (0..2) : Conditioning of analog input signal.

0 = 0-10V or 0-20mA

1 = 2-10V or 4-20mA

2 = NTC/PT thermistor temperature measuring

cFunPar4 : [0, 100] : range of 64-bit floating point value : Range of measured variable [minV,maxV], where 0(2)V/0(4)mA corresponds to minV and 10V/20mA to maxV.

4.4.2 Physical analog output

Input ports

1 real

Output ports

0 Only simulation output port to pass control application output into simulated process model.

Description

Function sends on specified analog output terminal electrical signal (0-10V / 0-20mA) according to input variable value and specified variable value range.

Function-specific parameters

cFunPar1 : 1 : (1..8) : Terminal position. Position of analog output on AO module terminal.

cFunPar2 : 1 : (1..1) : Module number. Position of AO module where is placed analog output.

cFunPar3 : 0 : (0..1) : Conditioning of analog output signal.

0 = 0-10V or 0-20mA

1 = 2-10V or 4-20mA

cFunPar4 : [0, 100] : range of 64-bit floating point value : Range of input variable [minV,maxV], where 0(2)V/0(4)mA corresponds to minV and 10V/20mA to maxV.

4.4.3 Physical digital input

Input ports

0 Only simulation input port to pass simulated process model digital input into control application.

Output ports

1 binary

Description

Function reads if closed or opened contact is connected to specified digital input terminal.

Function-specific parameters

cFunPar1 : 1 : (1..8) : Terminal position. Position of digital input on DI module terminal.

cFunPar2 : 1 : (1..4) : Module number. Position of DI module where is placed digital input.

cFunPar3 : 0 : Function output logic.

0: closed contact=1,

1: closed contact=0.

4.4.4 Physical digital output

Input ports

1 binary

Output ports

0 Only simulation output port to pass control application output into simulated process model.

Description

Function sends on specified digital output terminal electrical signal 24VDC or closes contact (depends on module type) according to input variable value.

Function-specific parameters

cFunPar1 : 1 : (1..8) : Terminal position. Position of digital output on DO module terminal.

cFunPar2 : 1 : (1..4) : Module number. Position of DO module where is placed digital output.

cFunPar3 : 0 : Function input logic.

0: 1=closed contact/24VDC on output,

1: 0=closed contact/24VDC on output.

4.4.5 Virtual Input

Input ports

0 Only simulation input port to pass simulated process model input into control application.

Output ports

1 or 4 or 7 or 23 real or binary or integer or any combination of these formats.

Description

Receives value via real-time UDP/IP protocol from a sender virtual output control function, see 15. If "OP" string is in the name of this control function, translation will create operator panel definition, see section 3.6.5Operator Panels.

Function-specific parameters

cFunPar1 : [10,0,0,11] : any IP address : IP address of sender as a vector of four 1-byte values.

cFunPar2 : 1044 : (0..65535) : Sender UDP port.

cFunPar3 : 1044 : (0..65535) : Local UDP port.

cFunPar4 : 1 : (1..3) : Received value format. 1=real, 2=binary, 3=integer.

cFunPar5 : 1100 : (1024..65535) : Receiver control function address.

4.4.6 Virtual output

Input ports

1 or 4 or 7 or 23 real or binary or integer or any combination of these formats

Output ports

0 Only simulation output port to pass control application output into simulated process model.

Description

Sends value via real-time UDP/IP protocol.

Function-specific parameters

FunPar1 : [10,0,0,11] : any IP address : IP address of receiver as a vector of four 1-byte values.

cFunPar2 : 1044 : (0..65535) : Receiver UDP port.

cFunPar3 : 1044 : (0..65535) : Local UDP port.

cFunPar4 : 1100 : if ID of the sender (virtual input) control function (0..1023), if address of the sender control function (1024..65535) : Receiver control function ID/address. Values from 0 to 1024 indicates sender ID (number assigned to control function during translation of the diagram). However this number may change after diagram modification and another translate. Values from 1024 to 65535 are fixed and they indicate sender (virtual input) address specified in virtual input parameters.

Note 1: *To guarantee data transmission at a desired sampling rate, it is recommended to set sampling frequency of virtual output to double or triple of the desired sampling rate. For example, to transmit data with sampling time 1sec, set*

sampling time of the virtual input to 0.5 or 0.3 sec.

Note 2: *For sending data to* ***one DCU*** *use preferably* ***one virtual output with multiple input ports*** *(faster one-package transfer) and not several virtual outputs (multiple-package transfer). The data packages of several virtual outputs may be sent in short interval one after another and receiving unit may miss the second package while processing the first one. If more virtual outputs are needed to be used for sending data to one DCU, set different sampling times with no multiplicity for each virtual output. For example 0.3sec, 0.4sec, 0.5sec.*

4.4.7 RS485 Modbus RTU master analog output

Input ports
1 real

Output ports

0 Only simulation output port to pass control application output into simulated process model.

Description

Sends value via RS485 serial link and Modbus RTU protocol with unit set as master. Writing real/int into single holding register command 0x06 or writing multiple holding registers command 0x10 (16). Received address must correspond to virtual output function ID number.

Function-specific parameters

FunPar1 : 0.1 : (0.000001..65535*baudrate-one-bit-time) : Maximum waiting time for response in seconds.

cFunPar2 : 7 : (1..247) : Address of slave to communicate with.

cFunPar3 : 1 : (0..65535) : Register address of the slave.

cFunPar4 : 0 : (0..7) : Conversion type

0: 1x16bit-uint16

1:1x16bit-int16

2:2x16bit-uint32

3:2x16bit-int32

4:2x16bit-float

5:4x16bit-ulong

6:4x16bit-long

7:4x16bit-double

cFunPar5 : 0 : (0..1) : Transmission order

0:lowest 16-bit first

1:lowest 16-bit last

4.4.8 RS485 Modbus RTU slave analog output

Input ports

1 real

Output ports

0 Only simulation output port to pass control application output into simulated process model.

Description

Sends value via RS485 serial link and Modbus RTU protocol with unit set as slave. Accepting read holding register command 0x03 or read input register command 0x04, received register address must correspond to control function ID number.

Function-specific parameters

cFunPar1 : 0 : (0..7) : Conversion type

0: 1x16bit-uint16

1:1x16bit-int16

2:2x16bit-uint32

3:2x16bit-int32

4:2x16bit-float

5:4x16bit-ulong

6:4x16bit-long

7:4x16bit-double

cFunPar2 : 0 : (0..1) : Transmission order

0:lowest 16-bit first

1:lowest 16-bit last

cFunPar3 : 1100 : (1024..65535) : Register address. Address must be > 1023.

4.4.9 RS485 Modbus RTU master analog input

Input ports

0 Only simulation input port to pass simulated process model input into control application.

Output ports

1 real

Description

Receives value via RS485 serial link and Modbus RTU protocol with unit set as master. Reading real/int from holding registers command 0x03 or reading input registers command 0x04). Number of function outputs and selected data format defines number of read registers

Function-specific parameters

FunPar1 : 0.1 : (0.000001..65535*baudrate-one-bit-time) : Maximum waiting time for response in seconds.

cFunPar2 : 7 : (1..247) : Address of slave to communicate with.

cFunPar3 : 1 : (0..65535) : Register address of the slave.

cFunPar4 : 0 : (0..7) : Conversion type

0: 1x16bit-uint16

1:1x16bit-int16

2:2x16bit-uint32

3:2x16bit-int32

4:2x16bit-float

5:4x16bit-ulong

6:4x16bit-long

7:4x16bit-double

cFunPar5 : 0 : (0..1) : Transmission order

0:lowest 16-bit first

1:lowest 16-bit last

4.4.10 RS485 Modbus RTU slave analog input

Input ports

0 Only simulation input port to pass simulated process model input into control application.

Output ports

1 real

Description

Receives value via RS485 serial link and Modbus RTU protocol with unit set as slave. Accepting command Writing real/int into holding register command 0x06, or multiple holding registers 0x10 (16) in case of multi-register data format. Both commands are accepted and register address must correspond to control function ID number.

Function-specific parameters

cFunPar1 : 0 : (0..7) : Conversion type

0: 1x16bit-uint16

1:1x16bit-int16

2:2x16bit-uint32

3:2x16bit-int32

4:2x16bit-float

5:4x16bit-ulong

6:4x16bit-long

7:4x16bit-double

cFunPar2 : 0 : (0..1) : Transmission order

0:lowest 16-bit first

1:lowest 16-bit last

cFunPar3 : 1100 : (1024..65535) : Register address. Address must be > 1023.

4.4.11 RS485 Modbus RTU master digital output

Input ports
1 (n) binary

Output ports

0 Only simulation output port to pass control application output into simulated process model.

Description

Sends value via RS485 serial link and Modbus RTU protocol with unit set as master. Writing single coil command 0x05, or multiple coils command 0x0F (15), depending on number of inputs. Number of inputs defines number of coils to write

Function-specific parameters

FunPar1 : 0.1 : (0.000001..65535*baudrate-one-bit-time) : Maximum waiting time for response in seconds.

cFunPar2 : 7 : (1..247) : Address of slave to communicate with.

cFunPar3 : 1 : (0..65535) : Coil address of the slave.

4.4.12 RS485 Modbus RTU slave digital output

Input ports
1 binary

Output ports

0 Only simulation output port to pass control application output into simulated process model.

Description

Sends value via RS485 serial link and Modbus RTU protocol with unit set as slave. Function accepts read coils command 0x01 or read discrete inputs command 0x02.

Function-specific parameters

cFunPar1 : 1100 : (1024..65535) : Register address. Address must be > 1023.

4.4.13 RS485 Modbus RTU master digital input

Input ports
0 Only simulation input port to pass simulated process model input into control application.

Output ports

1 binary

Description

Receives value via RS485 serial link and Modbus RTU protocol with unit set as master. Reading coils command 0x01, or reading discrete inputs command 0x03.

Function-specific parameters

FunPar1 : 0.1 : (0.000001..65535*baudrate-one-bit-time) : Maximum waiting time for response in seconds.

cFunPar2 : 7 : (1..247) : Address of slave to communicate with.

cFunPar3 : 1 : (0..65535) : Register address of the slave.

cFunPar4 : 0 : (0..1) : Data reading type

0: coil

1:discrete register

4.4.14 RS485 Modbus RTU slave digital input

Input ports
0 Only simulation input port to pass simulated process model input into control application.

Output ports

1 binary

Description

Receives value via RS485 serial link and Modbus RTU protocol with unit set as slave. Accept writing single coil command 0x05, or writing multiple coils 0x0F (15).

Function-specific parameters

cFunPar1 : 1100 : (1024..65535) : Register address. Address must be > 1023.

4.4.15 Constant

Input ports
0

Output ports

1 real or binary or integer

Description

Constant value source.

Function-specific parameters

cFunPar1 : 0 : real value range : Constant value

cFunPar2 : 1 : (1..3) : Value format

1:real

2:binary

3:integer

4.4.16 Nth order discrete transfer function (IIR filter)

Input ports
1 real

Output ports

1 real

Description

Discrete transfer function (IIR filter):

$$output(t)=\frac{N(z^{-1})}{D(z^{-1})}=\frac{n_0+n_1z^{-1}+...n_iz^{-i}}{1+d_1z^{-1}+...d_jz^{-j}}input(t)=G(z^{-1})input(t)$$

Function-specific parameters

cFunPar1 : [0.001 2] : any real vector : Numerator coefficients [b0,b1,...]

cFunPar2 : [-0.9] : any real vector : Denominator coefficients [a1,...]

cFunPar3 : 0 : (0..1) : **Not realized.** Linear transition, when coeff. modified (0:no, 1:yes)

cFunPar4 : 0 : (0..N) : **Not realized.** Transition time coefficient (TF sampling time / transition time).

Note: If any block connected to the filter has different sampling time than the filter a real-to-real block with the same sampling as the controller is required, see 22 where the same problem is treated for signal transformation.

4.4.17 Math function with one argument f(x)

Input ports
1 real

Output ports

1 real

Description

Mathematical function with one argument: *y = f(x)*, where y = output(t) and x = input(t).

Function-specific parameters

cFunPar1 : 0 : (0..23) : Function type. 0:1/x, 1:2^x, 2:x^2, 3:x^3, 4:x^3, 5:sin(x), 6:cos(x), 7:tan(x), 8:e^x, 9:$\log_e(x)$, 10:$\log_{10}(x)$, 11:$\sqrt{x}$, 12:abs(x), 13:asin(x), 14:acos(x), 15:atan(x), 16:sinh(x), 17:cosh(x), 18:tanh(x), 19:asinh(x), 20:acosh(x), 21:atanh(x), 22:$\sqrt[3]{x}$, 23:$\log_2(x)$

4.4.18 Math function with two arguments f(x,y)

Input ports
2 real

Output ports

1 real

Description

Mathematical function with two arguments: *y = f(x1,x2)*, where y = output(t), x1 = input1(t), x2 = input2(t).

Function-specific parameters

cFunPar1 : 0 : (0..7) : Function type.

0:x1+x2

1:x1-x2

2:x1*x2

3:x1/x2

4:$x1^{x2}$

5:max(x1,x2)

6:min(x1,x2)

7:avg(x1,x2)

4.4.19 Comparison of two analog values

Input ports
2 real

Output ports

1 binary

Description

Comparison of two real input values, resulting output is true (1) or false (0).

Function-specific parameters

cFunPar1 : 0 : (0..10) : Comparison type.

0:in1>in2

1:in1<in2

2:in1>=in2

3:in1<=in2

4:in1==in2

5:in1==in2 ± Tolerance

6:in1>in2 with switch-off hysteresis

7:in1<in2 with switch-off hysteresis

8:in1>=in2 with switch-off hysteresis

9:in1<=in2 with switch-off hysteresis

10:in1==in2 with switch-off hysteresis

cFunPar2 : 0.1 : any positive real number : Tolerance (for type 5) or switch-off hysteresis value (for type 6 to 10) depending on comparison type.

4.4.20 Analog input converter

Input ports
1 real

Output ports

1 real

Description

Converts analog input voltage measurement to temperature. It is considered, that converter has its input connected to output of physical analog input function and that this analog input is set to "NTC/PT thermistor temperature measuring".

Function-specific parameters

cFunPar1 : [0,0,0] : see below : Defines type of connected physical AI module. Measuring circuit (see Figure 21) is defined here for measurement of connected thermistor resistance

[0,0,0]: module NTC10k (R1=56kohm, R2=390000kohm)

[1,0,0]: module PT1000 (R1=4.7kohm, R2 is not present)

[R1,R2,0]: specific on-measure AI module with measuring resistances R1, and R2. If R1 or R2 not used set value to 0.

cFunPar2 : 0 : (0..6) : Type of conversion from resistance to temperature.

0: NTC beta function Tactual = B / ln(Ractual/r_∞) with $r_\infty = R_0 e^{B/T0}$ where B, R0, T0 need to be defined

1: PTC polynomial function Tactual = $a_n(Ractual/R_0)^n + a_1(Ractual/R_0) + a_0$ where R0, a0, … an need to be defined

2: General look-up table: R → T where resistances Ri and corresponding temperatures Ti need to be defined

3: PT100 table with temperature in degrees Celsius

4: PT1000 table with temperature in degrees Celsius

5: PT100 table with temperature in degrees Fahrenheit

6: PT1000 table with temperature in degrees Fahrenheit

cFunPar3 : [4100,10000, 25] : any real vector : Vector of conversion coefficients. Meaning depends on selected conversion type:

NTC beta function : [B, R0, T0]

PTC polynomial function [R0, a0, a1, ...an]

General look-up table [R0, R1, R2, … T0, T1, T2, …]

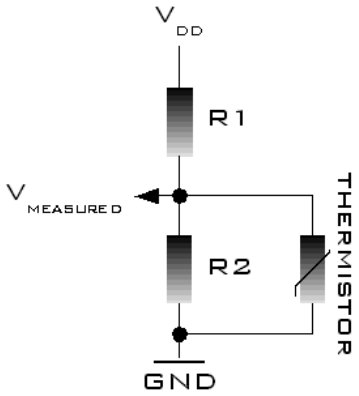


Figure 21: Measuring circuit for thermistors.

4.4.21 Dynamic switch

Input ports

1 real (selector), 2 or more real or binary or integer (switched inputs)

Output ports

1 real or binary or integer (selected input)

Description

Selects one input from switched inputs and pass its value to output according to selector input value and defined switching points [SWP1, SWP2,...SWPn]. Number of switching points n = number of switched inputs - 1. Rule is as follows:

> if selector input value belongs to set (-∞,SWP1] including SWP1 then output = switched input 1
>
> if selector input value belongs to set (SWP1, SWP2] including SWP1 and excluding SWP1 then output = switched input 2
>
> …
>
> if selector input value belongs to set (SWPn, ∞] excluding SWPn then output = switched input n+1

Function-specific parameters

cFunPar1 : 1 : (1..3) : Switching variable format. 1:real, 2:binary, 3:integer

cFunPar2 : [0] : any real vector : Switching point values [SWP1, SWP2,...].

4.4.22 Signal transformation d/dt, Gain, Offset, Saturation, Deadband, Polynom, Look-up, ∫ , Delay, format conversion

Input ports

1 real

Output ports

1 real

Description

Transform input value with specified function.

Function-specific parameters

cFunPar1 : (0..9) : Transformation function

0: Differentiation d/dt. output = d input /dt

1: Gain. output = gain*input

2: Offset. output = input + offset

3: Saturation.

if input ∈ (Lmin, Lmax) then input = output

if input < Lmin then output = Lmin

if input > Lmax then output = Lmax

4: Deadband.

if input ∈ (Lmin, Lmax) then input = 0

if input < Lmin then output = input - Lmin

if input > Lmax then output = input - Lmax

5: Polynomial function. output = $a_0 + a_1$*input + a_2*input^2 + …

6: Look-up table. output = TABLE(input) with TABLE defined by points [out0, in0], [out1, in1], …

7: Numerical integration $output(t) = \frac{1}{N}\sum_{i=1}^{N}[input(t-i)]$

8: Absolute value numerical integration

$$output(t) = \frac{1}{N}\sum_{i=1}^{N}[|input(t-i)|]$$

9: Square value numerical integration

$$output(t) = \frac{1}{N}\sum_{i=1}^{N}[input(t-i)]^2$$

10: Signal delay. Real (IOformat=1), binary (IOformat=2), integer (IOformat=3) signal delay in number N of time samples.

11: Signal format conversion from real/binary/integer input to real output.

12: Signal format conversion from real/binary/integer input to binary output.

13: Signal format conversion from real/binary/integer input to integer output.

cFunPar2 : [0] : any real vector : Coefficients of the transformation. Meaning depends on selected transformation:

for 0: []

for 1: [gain]

for 2: [offset]

for 3: [Lmin, Lmax]

for 4: [Lmin, Lmax]

for 5: [a0, a1, …]

for 6: [in1, in2, … out1, out2, …]

for 7, 8, 9: [N]

for 10: [N, IOformat]

Note: If any block connected to the signal transformation has different sampling time than the signal transformation a real-to-real block with the same sampling as the controller is required, see 22.

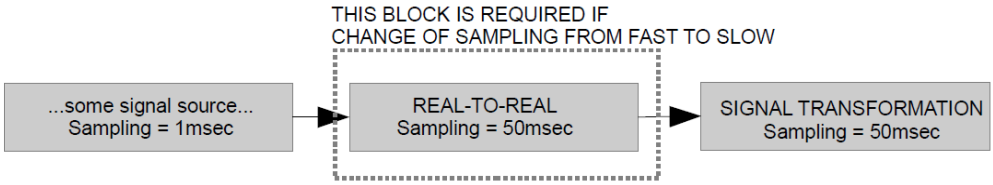


Figure 22: Connection of signal transformation with faster sampling signal source.

4.4.23 Multi-signal real-value operation

Input ports
N real

Output ports

1 real

Description

Multiple signal value operation for real variables.

Function-specific parameters

cFunPar1 : 0 : (0..4) : Function type

0:add

1:multiply

2:minimum

3:maximum

4:average

cFunPar2: 2 : (2..255) : Number of inputs.

4.4.24 Multi-signal binary-value operation

Input ports
N binary

Output ports

1 binary

Description

Multiple signal value operation for binary variables.

Function-specific parameters

cFunPar1 : 0 : (0..5) : Function type.

0:OR

1:AND

2:XOR

3:NAND

4:NOR

5:NXOR

cFunPar2: 2 : (2..255) : Number of inputs.

4.4.25 Binary edge processing, binary NOT

Input ports
1 binary

Output ports

1 binary

Description

Binary NOT may be set or processing binary signal edges - positive edge (transition from 0 to 1), negative edge (transition from 1 to 0). Delays of edges or pulse generation on appearance of an edge may be set.

Function-specific parameters

cFunPar1 : 0 : (0..1) : Processing function type.

0: on/off delay

1:positive/negative edge detection

cFunPar2 : 0 : (0..4294967295) Meaning depends on selected processing function. 0=unused, if cFunPar2=0 and cFunPar3=0, then NOT function applied.

For 0: On delay in number of samples

For 1: Positive edge pulse length in number of samples

cFunPar3 : 0 : (0..4294967295) Meaning depends on selected processing function. 0=unused, if cFunPar2=0 and cFunPar3=0, then NOT function applied.

For 0: Off delay in number of samples.

For 1: Negative edge pulse length in number of samples.

Note: If the block connected to the edge detector has different sampling time than the edge detector a binary-to-binary block with the same sampling as the edge detector is required, see 23.

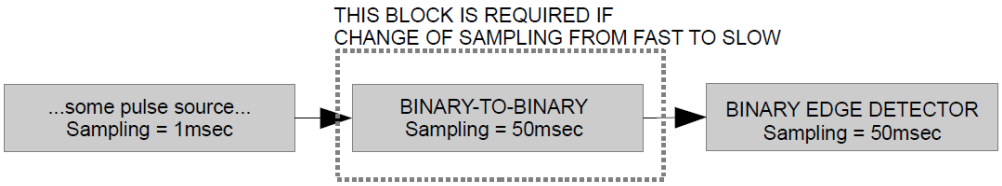


Figure 23: Connection of edge detector with faster sampling pulse source.

4.4.26 Memory

Input ports

1 real or binary or integer (value to memorize)

1 binary (write signal)

Output ports

1 real or binary or integer

Description

Set output to last value from input when write signal was 1. With write signal set to 1, actual output value = actual input value.

Function-specific parameters

cFunPar1 : 1 : (1..3) : Memorized value format (1:real, 2:binary, 3:integer).

cFunPar2 : 0 : (0..1) : Initialization type

0:initialization with actual input value

1:initialization with output variable initial value

4.4.27 Counter / timer

Input ports

2 binary (reset signal, write signal)

Output ports

1 real

Description

FOR INITIAL VALUE SET >=0 → COUNTER BEHAVIOR

The function increase by 1 actual output value when positive edge appears on write signal input. Resets output value to zero when 1 appears on reset signal input. When reset is at 1, output is kept at 0.

FOR INITIAL VALUE SET <0 → TIMER BEHAVIOR

If initial value of output control variable is set to a negative value, for example -1, the counter becomes TIMER. Timer counts each sample of its sampling time as soon as the write signal is set to 1. Reset input behavior is the same as for the counter. It resets timer output value to zero and keep it as long as 1 is on reset input.

Function-specific parameters

-

Note: If the block connected to the counter write signal has different sampling time than the counter a binary-to-binary block with the same sampling as the counter is required, see 24.

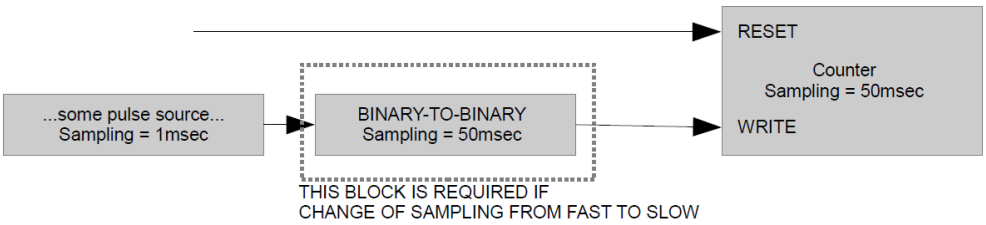


Figure 24: Connection of counter with faster sampling pulse source.

4.4.28 Schedule (time program)

Input ports

3 binary (force 1 signal, force 0 signal, reset forcing)

Output ports

1 binary

Description

Sets output to 1 or to 0 according to pre-defined time-table. Output may be forced to 1 for pre-defined time period if 1 appears on force 1 signal input. Output may be also forced to 0 for pre-defined time period if 1 appears on force 0 signal input. Forcing is reset if 1 appears on reset forcing input.

Function-specific parameters

cFunPar1 : 3600 : (1..4294967295) : Time period for forcing in number of samples. If a value <= 0 is set, forcing will end when next event occurs.

cFunPar2 - 16 : [44, 0, 7, 45, 31, 12] : see below : Definition of one event. First value of the vector specifies if a repetitive event is to be defined (33) or one-time event is defined (44).

Repetitive event definition

[33,

Value to set on output when time of event (0..1),

Hour of the event (0..23)

Minute of the event (0..59)

Day of week when event occurs 81=Sunday, 82=Monday, 83=Tuesday 84=Wednesday, 85=Thursday, 86=Friday, 87=Saturday. From 1 up to 7 days of week is defined

Months when events occurs. 1=January, … 12=December. From 1 up to 12 months is defined

]

Example: [33, 1, 7, 45, 82, 83, 84, 85, 10, 11, 12, 1, 2, 3, 4] means that output is set to

1 at 7h45 on Monday (82), Tuesday (83), Wednesday (84), Thursday (85), Friday (86) in cold months from October (10) up to April (4).

One-time event definition

[44

Value to set on output when time of event (0..1),

Hour of the event (0..23)

Minute of the event (0..59)

Day of the event (1..31)

Month of the event (1..12)

]

Example: [44, 0, 7, 45, 31, 12] means that output is set to 1 at 7h45 on December 31st

4.4.29 Variable state

Input ports

1 real or binary or integer (inspected variable)

1 binary (reset)

Output ports

1 binary

Description

Reads specified state of inspected variable or control function that outputs inspected variable and send it to output. If catching is set, output set to 1 is kept until 1 appears on reset input.

Function-specific parameters

cFunPar1 : 0 : (0..1) : Read element, control variable (0) or control function (1)

cFunPar2 : 0 : (0..1) Catching (0:no, 1: yes)

cFunPar3 : 0 : (0..31) : Read state ID number For control variable

0:initialization,

1:override,

2:dynamic override,

3:dynamic override finished,

4:alarm pending,

5: actual value need to be send by monitoring or history recording

6: type of actual value sending 1=monitoring, 0=history recording

16-31:alarm 1-16.

32: communication of modbus/rtudp failed => counter of failed communications reached limit 7

cFunPar4 : 0 : (0..4294967295) : On-delay in number of samples.

cFunPar5 : 0 : (0..4294967295) : Off-delay in number of samples.

4.4.30 State space

Input ports
Nu real

Output ports

Ny real

Description

Linear time invariant state space representation of dynamic system:

$x(t+1) = Ax(t) + Bu(t)$

$y(t) = Cx(t) + Du(t)$

where x(k) is a vector of states, u(t) is a vector of Nu actual inputs, y(t) is a vector of Ny actual outputs. A, B, C, and D are real matrices of corresponding dimensions.

Function-specific parameters

cFunPar1: [0, 0]: any real vector : Initial state vector x(0).

cFunPar2 : [[0.001 0.0002];[0.0003 0.001]] : any real matrix : Matrix A

cFunPar3 : [0.01;0.02] : any real matrix : Matrix B.

cFunPar4 : [0.03, 0.04] : any real matrix : Matrix C.

cFunPar5 : [0] : any real matrix : Matrix D.

4.4.31 RST / PID controller

Input ports
2 real (set-point and measured process value)

1 real (switch indicator)

Output ports

1 real

Description

General dynamic linear time-invariant controller [3] is defined by the following equation:

$u(t) = (1/S) (Tr(t) - Ry(t))$

where u(t) is controller output, r(t) is set-point, and y(t) is measured process variable. R,S, and T are discrete-time polynomials:

$T = t_0 + t_1*z^{-1} + t_2*z^{-2} + \ldots$

$R = r_0 + r_1*z^{-1} + r_2*z^{-2} + \ldots$

$S = s_0 + s_1*z^{-1} + s_2*z^{-2} + \ldots$

z^{-1} is delay operator ($z^{-1}y(t) = y(t-1)$) and r_i , t_i and s_i are controller coefficients.

Controller function allows to set transition period in case of parameters modification. Switching between multiple controllers is handled as well. Output of the dynamic switch block must be connected to switch indicator input. No block is allowed

between controllers and dynamic switch block to handle switching properly, see figure.

The RST controller is sufficiently general structure to cover many specific linear time-invariant controllers such as PI/PID. The controller can be defined as general RST controller but also as a specific-form controller. The following forms are actually supported:

Form 0. General discrete-time R-S-T controller, see above.

Form 1. Discrete-time PID controller structure 1. Continuous-time equivalent of PID controller has the following form:

$$u(t)=K_p[1+\frac{1}{T_i s}+\frac{T_d s}{1+(T_d/N)s}](r(t)-y(t))$$ where Kp is proportional gain, Ti is so called integral time, Td is differential coefficient and N helps approximate differentiation. In time domain the equation takes the following form:

$$u(t)=Kp(r(t)-y(t))+Kp\int(r(t)-y(t))dt+Kp\frac{d(r(t)-y(t))}{dt}$$

where the differentiation is only approximation of differentiation. Dicrete-time approximation of the continuous-time PID has general RST form with thwe following polynomials:

$R = r_0 + r_1 * z^{-1}$

$S = (1 - z^{-1})(1 + s_1' * z^{-1})$

T=R

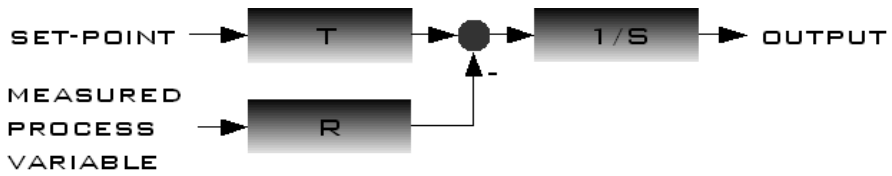


Figure 25: R-S-T controller structure.

Form 2. Discrete-time PID controller structure 2. Continuous-time equivalent corresponds to form 1, as well as RST controller polynomials R and S. The only difference is in polynomial T. Instead of complete polynomial R only static gain of the polynomial is used, i.e.

$T = r_0 + r_1$

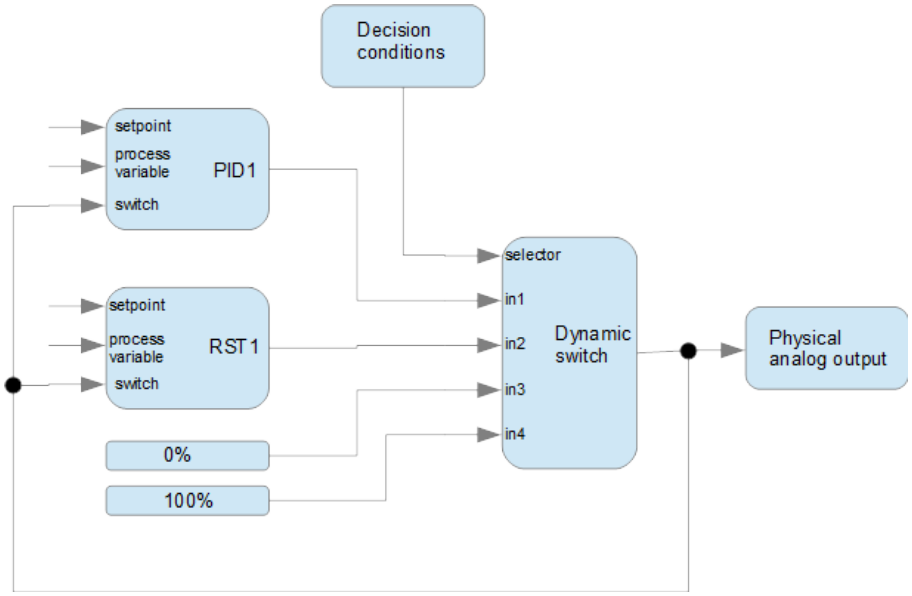


Figure 26: Example of correct controller switching diagram.

Function-specific parameters

cFunPar1 : 0 : (0..1) Parameters format

0: general RST controller

1: PID1 controller

2: PID2 controller

cFunPar2 : [0,100] : any real vector of two elements : Controller output limits [min, max].

cFunPar2 : see below : see below : Meaning depends on selected parameters format:

For RST: T polynomial coefficients [t0, t1,...]

For PID1: Proportional gain Kp

For PID2: Proportional gain Kp

cFunPar3 : see below : see below : Meaning depends on selected parameters format:

For RST: R polynomial coefficients [r0, r1, ...]

For PID1: Integral time Ti

For PID2: Integral time Ti

cFunPar3 : see below : see below : Meaning depends on selected parameters format:

For RST: S polynomial coefficients [s0, s1, s2, …]

For PID1: Differential coefficients [Td N]

For PID2: Differential coefficients [Td N]

cFunPar4 : 0 : (0..4294967295) : Transition period for parameter change in number of

samples

Note: If any block connected to the RST/PID controller set-point or measured process value has different sampling time than the controller a real-to-real block with the same sampling as the controller is required, see 22, 23 or 24 where the same problem is treated.

4.4.32 Binary signal generator

Input ports
0

Output ports

1 binary

Description

Binary signal generator. Allows to generate periodic pulse signal or pseudo-random binary sequence (PRBS). Generated PRBS is defined by generating register length and frequency divisor. The register length NR defines how many samples NS of the PRBS will be actually random. After that number of samples the sequence will be repeated. $NS = 2^{NR} - 1$ so to generate 1023 random samples, the register length must be at least 10. Frequency divisor tells how many times is repeated each generated sample. Consider that only NS samples is used. In frequency domain, PRBS has then the same magnitude for all frequencies up to half of sampling time frequency, but very low frequencies are attenuated. To amplify very low frequency, divisor may be raised. This brings attenuation of some of high frequencies but improve low frequency response.

Function-specific parameters

cFunPar1 : 0 (0..1) : Signal form

0: periodic pulse signal

1: pseudo-random binary sequence

cFunPar2 : 5 : (1..4294967295) : Meaning depends on signal form:

For 0: Length of 0 in number of samples

For 1 : Generating register length

cFunPar2 : 5 : (1..4294967295) : Meaning depends on signal form:

For 0: Length of 1 in number of samples

For 1: Frequency divisor

4.5 New special control functions

Library `special_block_functions_dcu_library` contains new special control functions that are developed for specific needs. User may use them as an ordinary control functions. Library contains their descriptions. Check the newest control function libraries to get all new developed control functions.

5 DCU Visualization

5.1 Introduction

For control system network with any number of DCU control units a single visualization server is needed (see 2). Visualization server is a computer that records historical data of control system and provides graphical user interface to see online and historical process data.

Requirements for visualization server are weak, any computer or even tablet is suitable. The requirements for web-based visualization server are as follows:

- Any newer version of Java Runtime Environment installed (JRE)
- Any AMP package installed and running (for example free Bitnami LAMP/WAMP/MAMP stacks)

Notice that visualization system is cross-platform system, i.e. OS (operating system) - independent. Operating systems like Windows, Linux, MacOS may be used provided they satisfy requirements defined above. Android OS is supported as well, but with special application developed for process data recording and visualization of DCU Control Systems.

5.2 Web-based visualization development step by step

To build a web-based visualization for a DCU control system, control applications for each control unit are expected to be finished and assembled in on control design project.

Step 1 – Start DCU Design Tool and open control design project containing all DCU controllers and control application diagrams and translate all control application diagrams, if not already done.

Step 2 – In DCU Design Tool define parameters of visualization server. Namely recording, visualization, alarm watch guard, virtual I/O parameters for (DCU Design Tool menu 'Visualization Server Setup ', for details see section 3.6DCU Design Toolbox visualization setup).

Step 3 – Click “Set Visualization Server” → Locally to set visualization server parameters on your computer.

Note: *“Set Visualization Server” sets for Apache web-server, among other parameters, a path to actual project visualization via “127.0.0.1/dcu/” path. If designer works on multiple visualizations of different projects, each time he switches the project and applies “Set Visualization Server”, Apache web-server need to be restarted to re-read actual project visualization path.*

Step 4 – Click “Open Visualization Web Pages” to check visualization structure, login, view pages, etc.

Step 5 – (optional) To check real behavior of visualization: 1) connect computer to controller network. 2) set computer's IP address to IP address of the server (typically 10.0.0.2), 3) load control applications into DCU controllers, 4) update visualization databases of DCU controllers.

Step 6 – Build process view web-pages defined in Visualization window in Step 2. For building process view web page, see section 5.3Building process view page. Each time you modify a view page you may check its appearance by clicking on the view in the visualization web pages.

Step 7 – If needed, adapt visualization files according to user requirements. At the beginning of index.php file you may set

- Visualization main title displayed on header (variable $processname)
- Directory for temporary files of visualization (variable $tempfiledir)
- Maximum waiting time for basic User Command Center DCU call (variable $dcuresponsewaiting) in multiples of 100msec. If local network is used for all DCU and visualization server, value between 30 and 50 (3 to 5 seconds) is sufficient. If a DCU is accessed remotely vie Internet, larger waiting time may be needed.
- Character set in the header of the web-page (search for "charset=") should be the same used for writing web-pages in a text-editor, for example utf-8.

Step 8 – Return back to step 2 or 5 or 6 or 7 until satisfied.

5.3 Building process view page

File that defines process view web-page is created by user. Because it is PHP script it has extension .php. Examples of this file are in "Visualization\content\examples" folder of any project created using DCU design tool. The example files contains examples of all possible objects that can be displayed in a visualization page.

5.3.1 Process view page structure

General settings In the first part of the process view file designer defines general settings of the view such as fonts, color, background image. Example of the general settings is shown below

```
//PROCESS FILE SETTINGS
$GLOBALS['security_lowest_level'] = 0; //allow only users with higher security level
$GLOBALS['security_exception_ids'] = array(1);
$GLOBALS['theme_color_background']='#2E5FAE';
$GLOBALS['theme_color_text']='#F2FFFF';
$GLOBALS['show_header'] = true;
$GLOBALS['show_panel_errors'] = false;
$GLOBALS['show_panel_time'] = true;
$GLOBALS['panel_time_step_unit'] = 3600;  // time step in seconds [sec]=1, [min]=60, [hr]=3600, [day]=3600*24
$GLOBALS['panel_time_step'] = 1;
$GLOBALS['show_panel_alarms'] = false;
```

Display templates definition and initialization Second part of the process view file (see below) defines initialization for displayed elements.

```
//COMPONENT INITIALIZATION
$dispvars = array();
$left_page = 0;
$top_page = 0;
$initialization_component = new component_default();
$initialization_component->set_font_family('Roboto');
$initialization_component->set_font_size(25);
$initialization_component->set_font_color('black');
```

Elements to display process Third part of the process view file contains definitions of displayed elements (objects). This is the main part of the process view page to be adjusted by designer. One displayed element is defined by a set of lines as shown in example below:

```
$component = new component_text($initialization_component);
$dispvars[] = $component;
/* element parameters */
$component->set_left($left_page);
$component->set_top($top_page);
$component->table_styles['width'] = '90%';  //aby bolo vycentrovane
$component->table_styles['text-align'] = 'center';
/* label parameters */
$component→set_label('Device name xxx');
```

Each line of text defines some property of the element such as position on web page, style, etc.

Detailed description can be found at “Visualization\content\examples” folder and at

https://www.prosystemy.sk/visualization

5.4 Web-based visualization deployment

See [8].

6 DCU Batch service

The following notation is used in DCU control system platform for batch definition. Batch is a user-defined sequence of phases (see 27). Each phase is, in fact, execution of a control task or of another batch. One batch may have one to infinite number of cycles (repetitions of the sequence of phases). Batches are executed by a batch server - a program for executing user-defined batches.

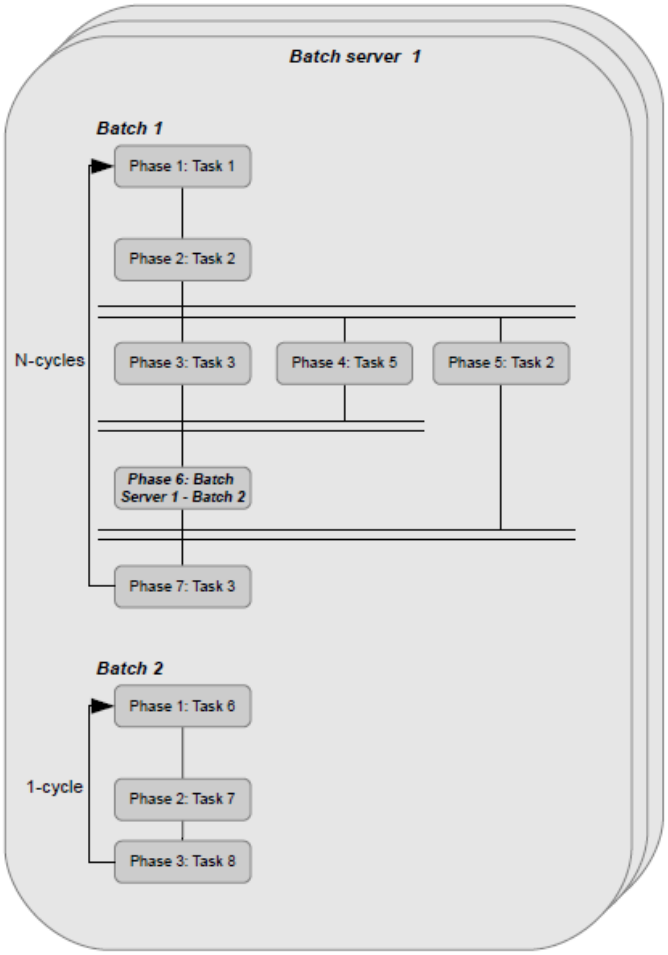


Figure 27: Principle of batch and batch server.

One batch server is defined by a set of control tasks organized in task groups and by server RTUDP input/output functions for communication with DCU controllers, see 28 Batch server definition contains also parameters that define maximum number of parallel phases or batches, and maximum number of task parameters. Batch servers are defined by control design engineer inside batch server window, see section Batch servers window.

Batch is supposed to be defined by user/operator of control system according to its technological requirements. One batch represents generally a recipe or a part of a

recipe for production of a specific product. Batch is defined in visualization system using web-based or operator panel batch editor application.

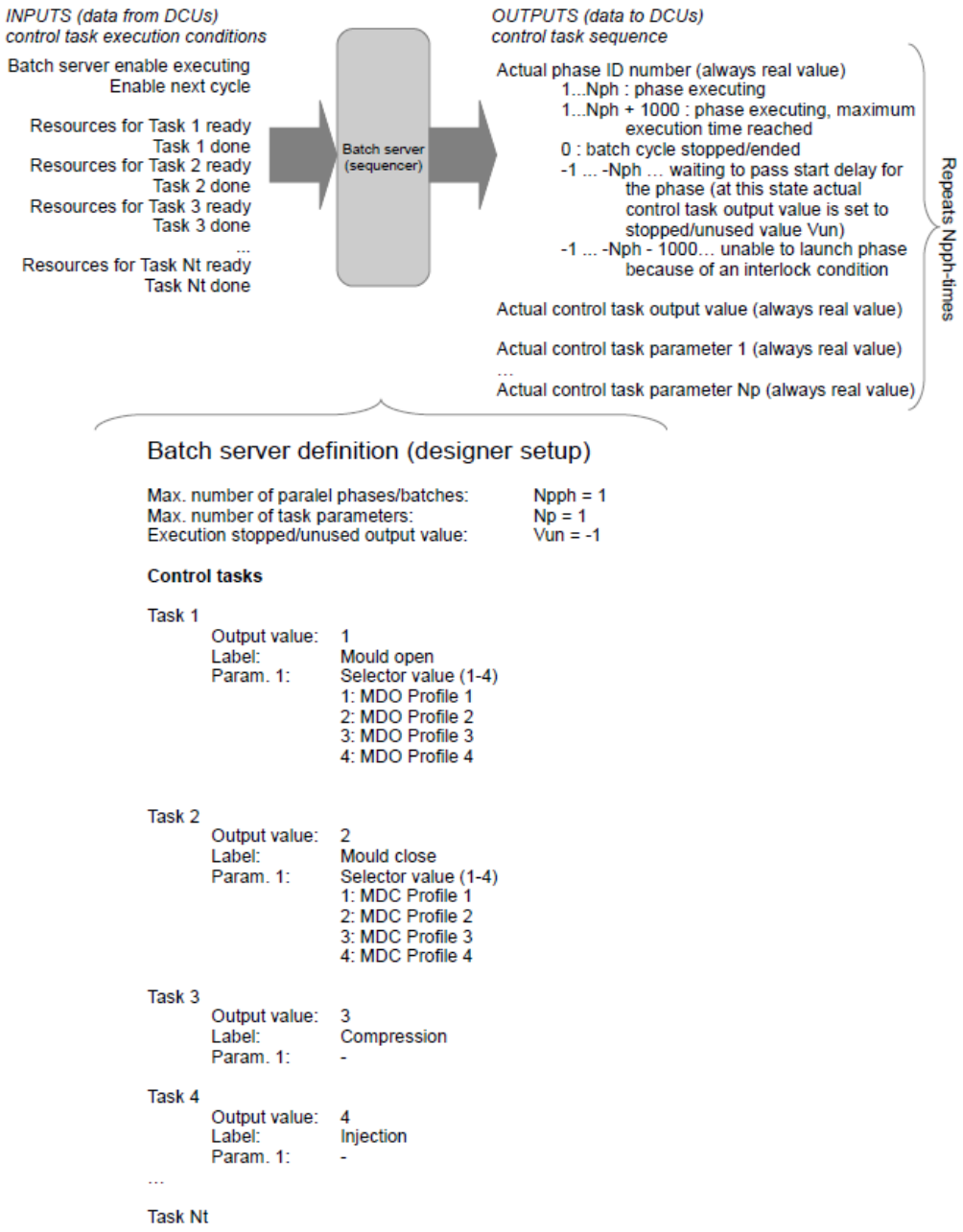


Figure 28: Batch server structure.

7 DCU hardware

7.1 Electrical specifications

7.1.1 DCU controller electrical wiring

It is recommended to use a separate power supply for DCU controller and other separate power supply for remaining control devices, such as actuators, Modbus I/O devices, active sensors. It is also recommended to use one separate power supply for Modbus devices from one producer. Examples of electrical wiring of DCU controller and connected control devices, sensors and actuators are shown in Figure 29

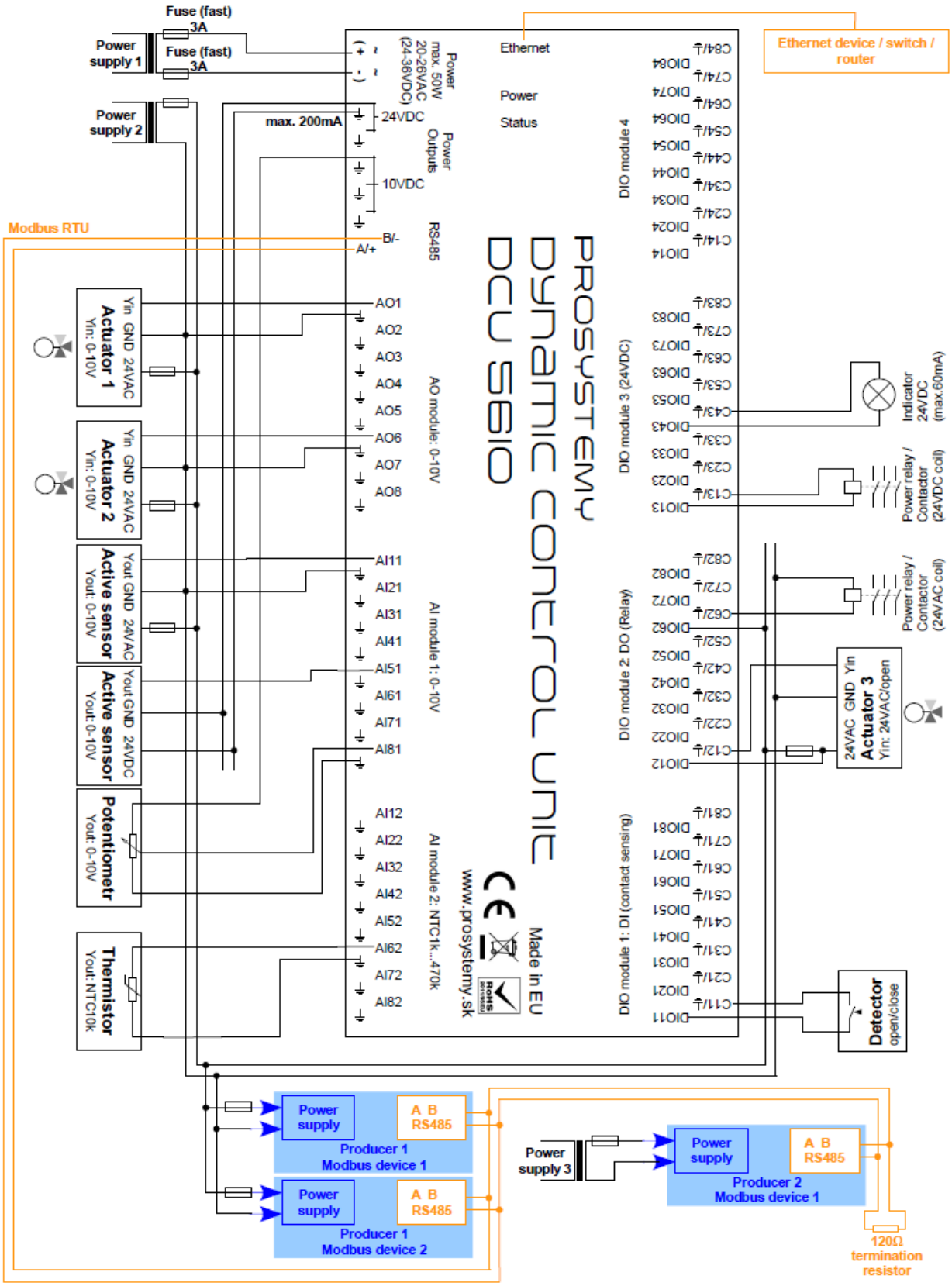


Figure 29: Examples of wiring control devices to DCU controller.

7.1.2 Analog inputs

16 analog inputs (AI) selectable in 2 modules of 8 inputs. 12-bit A/D converters are implemented. The following types of AI modules are available(see [5]):

- **AI module 0-10V.** 8x 0-10V
- **AI module 0-20mA.** 8x 0-20mA to measure 0-20mA signal connect in parallel with measured signal a 500ohm resistor between measuring terminal and ground.
- **AI module NTC10k.** 8x NTC thermistor temperature measuring module. The module is specifically designed for NTC with resistance between 10kW and 20kW, but it can be used for other thermistors while loosing either precision or measuring range, see Annex A. Measuring circuit is shown in Figure 30, resistances are R1=56kohm, R2=390000kohm. Specific on-measure module with other resistance configuration may be ordered.
- **AI module PT1000.** 8x PT1000 or Ni1000 thermistor temperature measuring module, see Annexe B for precision and range. Measuring circuit is shown in the Figure, resistances are R1=4k7kohm, R2 is not present. Specific on-measure module with other resistance configuration may be ordered.
- **AI module on-measure.** User-defined combination of analog inputs, see [5].

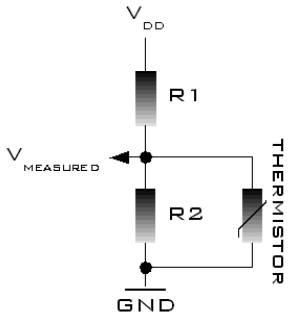


Figure 30: Measuring circuits for
AI module NTC10k and PT1000.

7.1.3 Analog outputs

8 analog outputs (AO) in 1 module of 8 outputs. 12-bit D/A converters are implemented. The following types of AO modules are available:

- **AO module 0-10V.** 8x 0-10V output, max. 50mA each.
- **AO module 0-20mA.** 8x 0-20mA output, loop power supply voltage typically ranges between 12VDC and 36DC, the most common is 24VDC.

7.1.4 Digital (binary) inputs and outputs

32 digital (binary) inputs or outputs (DIO) selectable in 4 modules of 8 channels as digital input or digital output. The following types of DIO modules are available:

- **DI module.** 8x digital input sensing open/close contact. Open contact is

indicated for resistance higher than ≈ 30kohm, closed contact is for resistance lower than ≈ 15kohm.

- **DO module 24VDC.** 8x digital output 24VDC 60mA / 0V
- **DO module relay.** 8x digital output relay contact normally opened, max. 50V, 0.5A.

7.1.5 Communication connectors

RJ45 connector for Ethernet TCP/IP and UDP/IP connectivity.

Terminal A/B/GND for RS485 serial link connectivity.

7.1.6 Auxiliary circuits

Real-time and date clock with backup battery integrated.

Auxiliary power supply for AI reference purposes: 10VDC/10mA

Auxiliary power supply for instrumentation devices power supply purposes: 24VDC/0.2A.

7.1.7 Power supply

Unit power supply possibilities:

- AC: 24VAC, max. 26VAC.
- DC: from 15VDC up to 30VDC. For voltage equal and below 24VDC the auxiliary power supply for instrumentation devices of 24VDC does not produce 24volts.

Power consumption depends on DCU configuration, in all cases it is less than 2A.

7.2 Physical specifications

Operational temperatures: from 0°C up to 50°C.

Dimensions of DCU56IO are as follows: 230mm x 139mm x 56mm.

Dimensions of DCU28IO are as follows: 160mm x 100mm x 25mm.

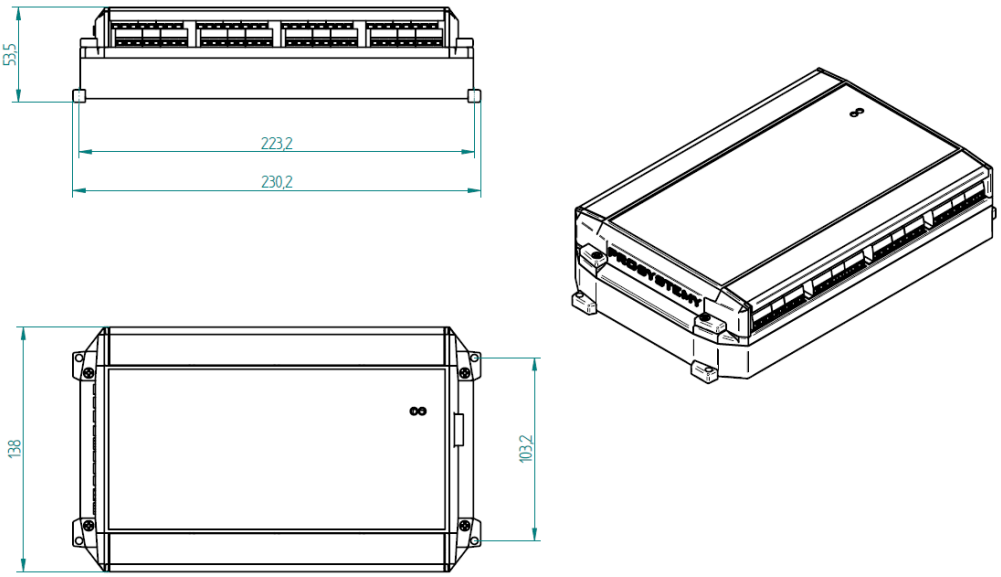
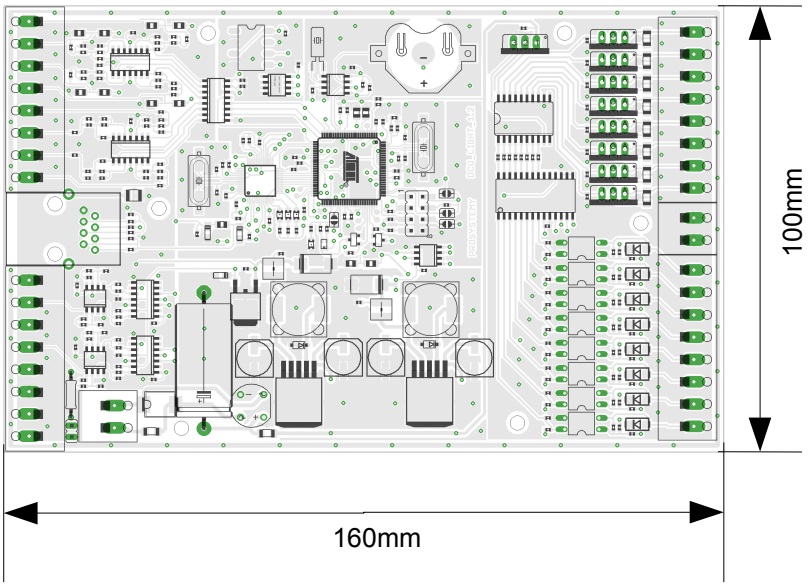


Figure 31: DCU56IO dimensions.



Výška: 25mm

Figure 32: DCU28IO dimensions.

7.3 Reset to default communication settings

It may happen that actual communication settings of DCU are corrupted during modification or lost by user and it is not possible to connect to DCU. Namely IP address may be forgot or ID number or password of DCU administration user (user

with highest security level 15) is lost or encryption keys are lost. In such cases, DCU communication settings may be reset to the following default values:

- **DCU user access.** Only one user is defined
  - User ID number = 8
  - User security level=15
  - User password=”root”
- **TCP/IP access.**
  - DCU IP address = 10.0.0.10
  - User command TCP port = 1033
- **Encryption.**
  - User command encryption - disabled.
  - History recording encryption - disabled.
  - Real-time UDP encryption - disabled.

To do so a hardware communication reset must be performed on DCU. The procedure of the hardware reset is as follows:

- Switch off power supply of the DCU
- Remove the upper cover of the controller
- Remove all I/O modules as well as power module.
- Remove intermediate cover, so that the main board of the unit is accessible
- Replace power module back into the main board.
- Switch-on power supply of DCU
- Connect reset pin to a ground for at least 5 seconds, see Figure 33.
- Switch-off power supply and reassemble control unit.

Note that new version of control unit DCU30IO has micro-switch for resetting to default parameters.

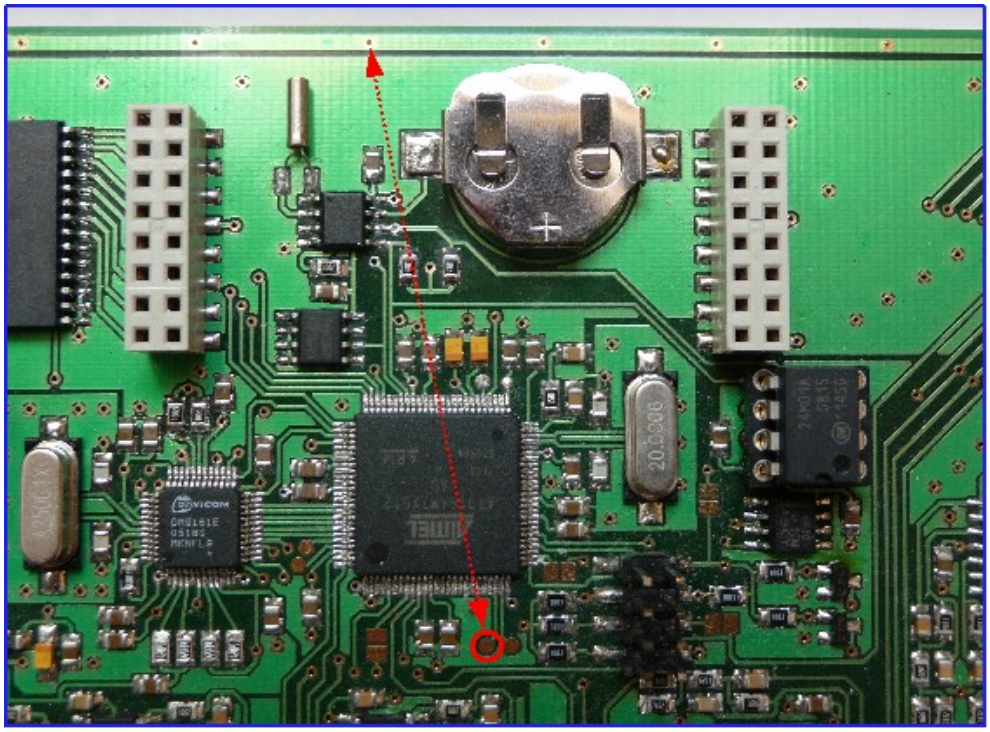


Figure 33: Hardware reset on DCU controller.

7.4 Loading of embedded real-time system

7.4.1 Load using design tool

New version of embedded real-time system (firmware) may be uploaded into DCU using DCU design tool. Binary file of real-time system is available on web-page of DCU. To upload new real-time system into DCU :

- Create or open a DCU design project.
- Translate a control application functional diagram and click on the translation so that the communication settings appear.
- Connect to DCU and check if connection established
- Click the button “Load real-time system (firmware)” and select binary file with the new real-time system. Load may take a few minutes.

Be sure that there will be no power supply interruption during upload!

8 Reference

[1] Scilab Online Help, help.scilab.org/

[2] Simulink Online Help, www.mathworks.com/help/simulink/

[3] I.D. Landau, G.Zito, Digital Control Systems, Design, Identification, and Implementation, Springer-Verlag, 2007

[4] DCU56IO Technical Specifications, Prosystemy, 2015.

[5] DCU56IO IO Modules Technical Specifications, Prosystemy, 2015.

[6] DCU Control System Platform – Overview, Prosystemy, 2016

[7] DCU Control System Platform – Quick Start Guide, Prosystemy, 2016.

[8] DCU Control System Platform – Visualization Server Deployment, Prosystemy, 2016.

9 Annex A: Precision of AI module NTC10 for different thermistors

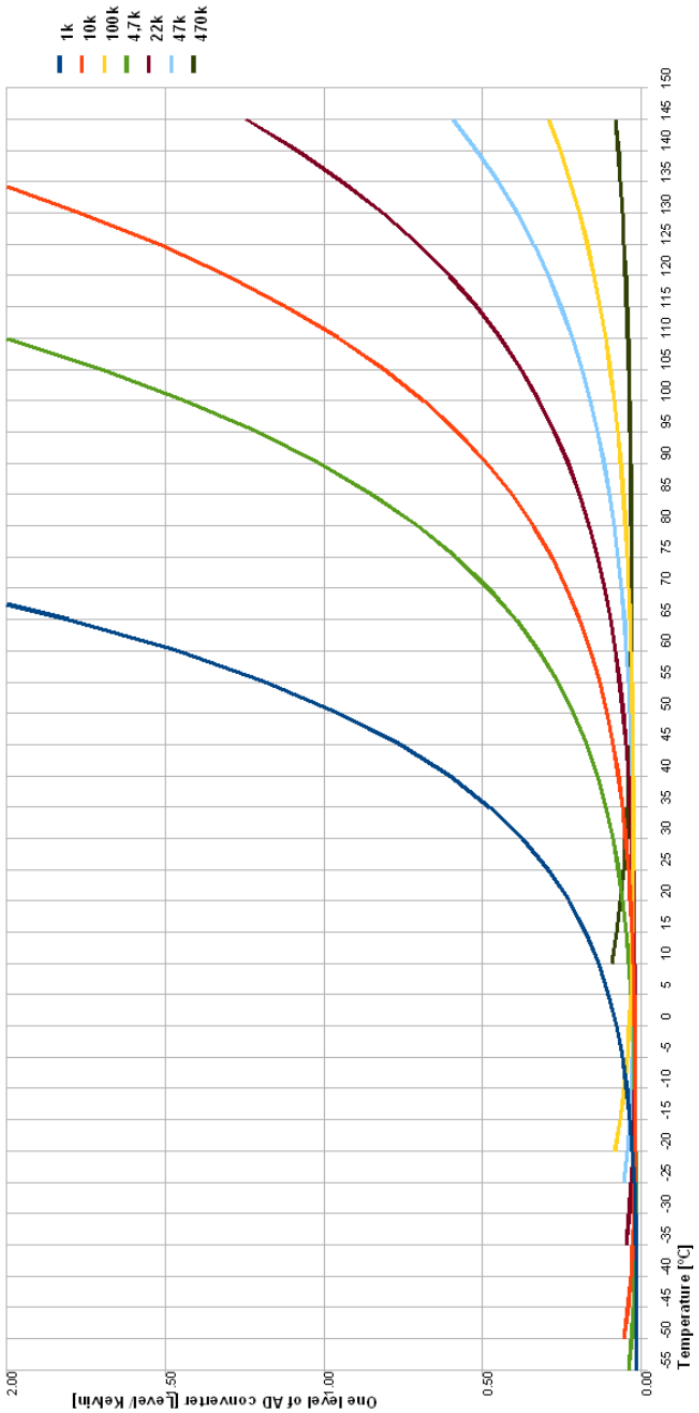


Figure 34: Precision of AI module NTC10.

10 Annex B: Precision of AI module PT1000 for different thermistors

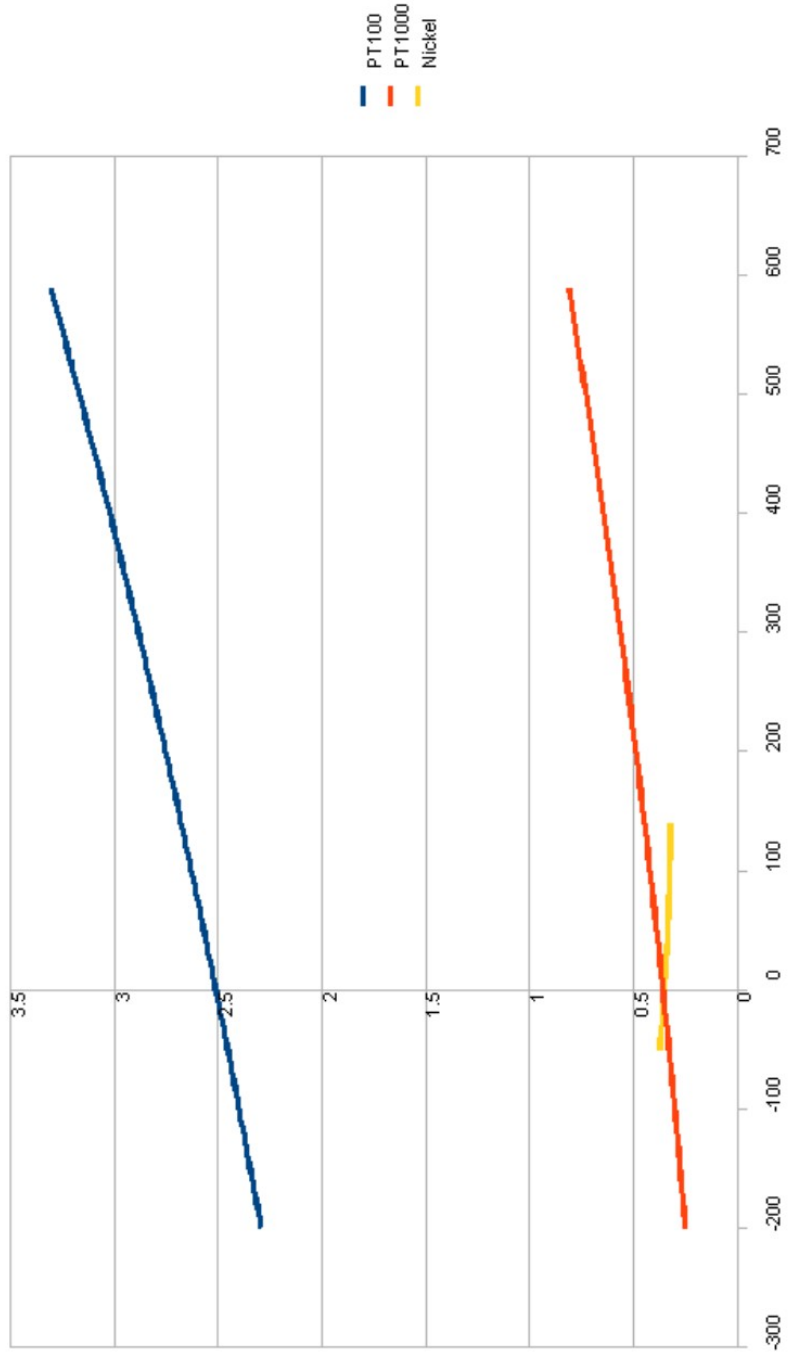


Figure 35: Precision of AI module PT1000.

11 Annex C: List of DCU events and alarms

11.1 DCU events and alarms

EVT_RTSYS_OVRLD 1 : Real-time control application overload problem. Control application was not finished while new sample evaluation was required. Event parameter: param=0 (fast interrupt), =1 (medium interrupt), =2 (slow interrupt)

EVT_RTSYS_LDAPP 2 : Real-time control application load executed

EVT_RTSYS_LDSYS 3 : Real-time system load executed

EVT_RTSYS_MANVAR 4 : Real-time control application variable forcing (manual override) executed. Parameter is bit0-15 new control variable override option, bit16-31 control variable ID.

EVT_RTSYS_MODPAR 5 : Real-time control application or system parameter modified

EVT_RTSYS_RESET 6 : Real-time system reset

EVT_RTSYS_HIST_OVERFLOW 7 : Real-time system history recording on user bus has signalled overflow, i.e. no free buffers for transmission. Event parameter: parameter=0 => encountered in c.app. execution interrupt; =1 encountered in main code

EVT_RTSYS_HIST_NOOVERFLOW 8 : Real-time system history recording on user bus has signalled end of overflow, i.e. free buffers are again ready Event parameter: parameter=0 => encountered in c.app. execution interrupt; =1 encountered in main code

EVT_RTSYS_EVENT_OVERFLOW 9 : Real-time system events history recording on user bus has signalled overflow, i.e. some events will not be recorded on user-bus because of events buffer overflow. Event parameter: event parameter indicates number of buffer overflows

EVT_RTSYS_RECORD_UNDELIVERED 10 : User-bus history recording service was unable to deliver data (too many retransmissions). Event parameter: paramater indicates how many undeliver events has occured

EVT_RTSYS_RECORD_CONNECTED 11 : User-bus history recording service succsessfully opened connection with database

EVT_RTSYS_RECORD_CONN_LOST 12 : User-bus history recording service was unable to transmit data to database too many times, connection probably lost

EVT_RTSYS_UNKNOWN_FORMAT_RTUDP 13 : Real-time UDP receiver has received and throwed away some data with unknown/unsupported format, format id is in parameter of the event

EVT_RTAPP_NO_UDPCONN 14 : Real-time UDP receiver/sender could not find an UDP connection structure, parameter indicates virtual input/output function number bit 0-15, and max. number of UDP connection bit16-31

EVT_RTAPP_RTUDP_RX_DISRUPT 15 : Real-time UDP receiver has received a value which is not directly next value, parameter indicates sender IP address

EVT_RTAPP_UNKNOWN_FORMAT_RTUDP 16 : Real-time UDP input/output

function has defined some unknown/unsupported format data will not be processed, format id is in parameter of the event

EVT_RTSYS_RTUDP_OVERFLOW 17 : Real-time system RT UDP service has signalled overflow, i.e. no free buffers for transmission

EVT_RTSYS_RTUDP_NOOVERFLOW 18 : Real-time system RT UDP service has signalled end of overflow, i.e. free buffers are again

EVT_RTAPP_AIO_UNKNOWN_RANGE 19 : Physical analog input/output function has specified an unknown value range, parameter indicates function Id. (bit0-15) and format (bit16-31)

EVT_RTSYS_LOW_TEMP_ALARM 20 : Control unit low internal temperature alarm (below 5degC=41degF), parameter indicates actual alarm value

EVT_RTSYS_LOW_TEMP_ALARM_OFF 21 : Control unit low internal temperature alarm (below 5degC=41degF) has ended, parameter indicates the most extreme value reach during alarm

EVT_RTSYS_HIGH_TEMP_ALARM 22 : Control unit high internal temperature alarm (above 80degC=176degF), parameter indicates actual alarm value

EVT_RTSYS_HIGH_TEMP_ALARM_OFF 23 : Control unit low internal temperature alarm (above 80degC=176degF) has ended, parameter indicates the most extreme value reach during alarm

EVT_RTSYS_LOW_POWER_ALARM 24 : Control unit low power supply voltage alarm, parameter indicates actual alarm value

EVT_RTSYS_LOW_POWER_ALARM_OFF 25 : Control unit low power supply voltage alarm has ended, parameter indicates the most extreme value reach during alarm

EVT_RTSYS_HIGH_POWER_ALARM 26 : Control unit high power supply voltage alarm, parameter indicates actual alarm value

EVT_RTSYS_HIGH_POWER_ALARM_OFF 27 : Control unit high power supply voltage alarm has ended, parameter indicates the most extreme value reach during alarm

EVT_RTAPP_DYNOVRD_LATE 28 : Dynamic override was cancelled because its processing was late or slow with respect to time schedule, parameter indicates control variable ID number

EVT_RTAPP_DYNOVRD_OPEN_FAIL 29 : Dynamic override was cancelled because TCP connection opening with source has failed, parameter indicates control variable ID number

EVT_RTAPP_MODBUS_BADCRC 30 : MODBUS received frame with incorrect CRC,

EVT_RTAPP_MODBUS_BADADDR 31 : MODBUS received bad register/coil/input address, i.e. control function id., parameter contains received address value

EVT_RTAPP_MODBUS_BADSIZE 32 : MODBUS received frame with incorrect value (size, amount of registers,...)

EVT_RTAPP_MODBUS_WRONGSLAVE 33 : MODBUS wrong slave address

responded to request, parameter indicates wrong address

EVT_RTAPP_MODBUS_EXCEPTREC 34 : MODBUS slave responded with exception. Parameter indicates slave address bit1-7, and exception code bit8-15

EVT_RTAPP_MODBUS_WAITRESP 35 : MODBUS slave did not responded in specified time, parameter indicates slave address

EVT_RTAPP_MODBUS_NOTALLDATA 36 : MODBUS slave did not send all announced data. Parameter indicates slave address bit1-7, bit8-15 amount of really received data bytes, bit16-23 amount of announced data bytes

EVT_RTAPP_MODBUS_WRONGDATA 37 : MODBUS master send wrong data, parameter indicates modbus virtual I/O function id

EVT_RTAPP_MODBUS_NDATA 38 : MODBUS master send request to write too many values, parameter indicates modbus virtual I/O function id

EVT_RTAPP_MODBUS_NOTINTIME 39 : MODBUS virtual I/O function did not process data within specified sampling time since this moment (modbus not in time alarm on event), parameter indicates modbus virtual I/O function id

EVT_RTAPP_MODBUS_INTIME 40 : MODBUS virtual I/O function succeded to process data within specified sampling (modbus not in time alarm off event), parameter indicates modbus virtual I/O function id

EVT_RTSYS_FLASH_ACCESS 41 : RT system was unable to write into FLASH, parameter indicates what data were not changed. Event parameter:

1 - switch into on-pc control mode failed

2 - switch into on-chip simulation mode failed

3 - modification of on-chip simulation mode parameters

EVT_RTSYS_ONCHIP_SIMUL_CONNECTED 42 : Control unit in on-chip simulation control mode has successfully opened connection with simulation PC/server

EVT_RTSYS_RTC_RESTORED 43 : RTC clock restored operation after previous failure

EVT_RTSYS_RTC_FAIL 44 : RTC clock failed to operate correctly

EVT_RTSYS_LACK_CRYPT_MACHINE 45 : Not enough crypting machines, some data will not be encrypted/decrypted

EVT_RTSYS_TEMP_READ_FAIL 46 : Reading internal control unit temperature failed

EVT_RTSYS_POW_READ_FAIL 47 : Reading control unit power failed

EVT_RTSYS_HISTORY_RECORD_CONNECTED 48 : Control unit has successfully opened connection with history recording PC/server, parameter indicates number of opening retries

EVT_RTAPP_MODBUS_COMFAIL 49 : MODBUS communication failed and will be restarted

EVT_RTSYS_STARTUP 50 : Real-time system startup, parameter indicates time of corresponding shutdown : bit0-5 (min), bit6-11 (h), bit12-17 (day), bit18-23 (month),

bit24-31 (year - 2000)

EVT_RTSYS_BACKUP_LOADED 51 : Control application was at startuploaded from backup, because backup was not equal to actual control application, parameter indicates application size

EVT_RTSYS_TCPIP_CLEANUP 52 : TCP/IP was blocked for too long time, cleanup of all related structure was made, parameter indicates type of block

EVT_RTSYS_RESET_RTSYS 53 : internal error caused reset, parameter indicates debug state of real-time system

EVT_RTSYS_RESET_FAPP 54 : internal error caused reset, parameter indicates debug state of fast interrupt

EVT_RTSYS_RESET_MAPP 55 : internal error caused reset, parameter indicates debug state of medium interrupt

EVT_RTSYS_RESET_SAPP 56 : internal error caused reset, parameter indicates debug state of slow interrupt

EVT_RTSYS_SPURIOUS_INT 57 : spurious interrupt during enable/disable interrupts

EVT_RTSYS_OVRLD_END 58 : real-time control application overload problem ended. Control application is finished on-time. Parameter indicates interrupt 0=fast, 1=medium, 2=slow

EVT_RTSYS_REM_EEPROM_KO 106 : removable EEPROM (for control application backup) failed to communicate properly

EVT_RTSYS_FIXED_EEPROM_KO 107 : fixed EEPROM (for control application description) failed to communicate properly

EVT_RTSYS_IO_PROCESSING_UNFINISHED 108 : waiting for finishing IO processing too long, parameter is actual RT system state

EVT_RTAPP_ALARM(x) (60+x) : Real-time control application alarm value number x was reached for a control variable defined in event parameter (x is from 0 to 15)

EVT_RTAPP_OFFALARM(x) (80+x) : Real-time control application alarm value number x was deactivated for a control variable defined in event parameter (x is from 0 to 15)

11.2 Events of visualization

Iinstead of unit event time, visualization user ID is registered

EVT_VIS_RTSYS_MODIF 101 : modification of real-time system parameters. Event parameter:

bit0 indicates if the operation was successful (0) or not(1)

bit1-8 visualization user ID (0-255)

EVT_VIS_CVAR_OVRD 102 : modification of control variable override, Event

parameter:

bit0 indicates if the operation was successful (0) or not(1)

bit1-8 visualization user ID (0-255)

bit9-18: control variable ID,

bit19-22: override status, bit23-26: operation

EVT_VIS_CVAR_PARAM 103 : modification of control variable parameters, Event parameter:

bit0 indicates if the operation was successful (0) or not(1)

bit1-8 visualization user ID (0-255)

bit9-18: control variable ID,

EVT_VIS_CAPP_MOD 104 : modification of control modes. Event parameter:

bit0 indicates if the operation was successful (0) or not(1)

bit1-8 visualization user ID (0-255)

bit9-31: new control mode states

EVT_VIS_DCU_TIME 105 : modification of DCU time and date. Event parameter:

bit0 indicates if the operation was successful (0) or not(1)

bit1-8 visualization user ID (0-255)

bit9: server time used (1), user-defined time used (0)

EVT_VIS_DCU_LOAD 109 : control application upload. Event parameter:

bit0 indicates if the operation was successful (0) or not(1)

bit1-8 visualization user ID (0-255)

bit16-32: ID number of DCU control unit

EVT_VIS_DCU_FUNPAR_MODIF 111 : modification of control function parameter. Event parameter:

bit0 indicates if the operation was successful (0) or not(1)

bit1-8 visualization user ID (0-255)

bit9-15: parameter index

bit16-32: ID number of control function