Tomáš Bezák

Usage of IEC 61131 and IEC 61499 Standards for Creating Distributed Control Systems

Scientific Monographs in Automation and Computer Science

Edited by
Prof. Dr. Peter Husar (Ilmenau University of Technology) and
Dr. Kvetoslava Resetova (Slovak University of Technology in Bratislava)

Vol. 3

USAGE OF IEC 61131 AND IEC 61499 STANDARDS FOR CREATING DISTRIBUTED CONTROL SYSTEMS

Tomáš Bezák

Universitätsverlag Ilmenau
2012

Impressum

Bibliographic information of the German National Library
The German National Library lists this publication in the German national bibliography, with detailed bibliographic information on the Internet at http://dnb.d-nb.de.

Author's acknowledgement to Jana Green for translation.

This scientific monograph originated from the author's dissertation thesis defended at the Slovak University of Technology in Bratislava, Faculty of Materials Science and Technology in Trnava.

Reviewers:
Peter Husár, Professor, Ph.D.
Aleš Janota, Professor, Ph.D.
Augustín Gese, Ph.D.

Author's contact address:
Tomáš Bezák, Ph.D.
Slovak University of Technology in Bratislava
Faculty of Materials Science and Technology in Trnava

Ilmenau Technical University / University Library
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Production and delivery
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISSN 2193-6439 (Print)
ISBN 978-3-86360-015-0 (Print)
URN urn:nbn:de:gbv:ilm1-2012100032

Titelfoto: photocase.com

Abstract

This publication deals with the application of standards for industrial automation during distributed control systems design. Control systems design consists of a choice between two approaches based on the standards, IEC 61131 and IEC 61499. The question is which of the standards to use for distributed control systems design. The most commonly used standards are briefly listed in the introduction section. Then follows a more detailed description of the IEC 61131 and IEC 61499 standards, future development of the IEC 61499 standard and its usage during the creation of distributed control systems. Further on are lists and descriptions of existing commercial and research software tools, which are necessary in implementing this standard. The main section deals with the methodology for standard application comparison and criteria selection for comparing. This methodology is then verified on real control systems. The final section includes methodology generalization for suitable approach selection, resulting in recommendations for which standard to choose during creation of distributed control systems.

Key words

distributed control system(s), IEC 61131, IEC 61499

LIST OF ACRONYMS

IEC	- International Electrotechnical Commission
ISO	- International Organization for Standardization
BSI	- British Standards Institution
UKAS	- United Kingdom Accreditation Service
SWEDAC	- Swedish Board for Accreditation and Conformity Assessment
GAMP	- Good Automated Manufacturing Practice
ISPE	- International Society for Pharmacoepidemiology
IT	- Information Technology
GxP	- Good Practice Quality Guidelines and Regulations
PLC	- Programmable Logic Controller
MAP	- Manufacturing Messaging Services
IPMCS	- Industrial Process, Measurement and Control System
XML	- Extensible Markup Language
OPC	- Object Language Embedded for Process Control
ST	- Structured Text
LAD	- Ladder Diagram
IL	- Instruction List
SFC	- Sequential Function Chart
FBD	- Function Block Diagram
FORTRAN	- Formula Translation
FB	- Function Block
ECC	- Execution Control Chart
FAT	- Factory Acceptance Test
SAT	- Site Acceptance Test
HMI	- Human Machine Interface

CBA	- Component Based Automation
SIFB	- Service Interface Function Blocks
ODECE	- Open Distributed Embedded Control Environment
FBDK	- Function Block Development Kit
FBRT	- Function Block Runtime
NCES	- Net Condition / Event Systems
OPC	- Open Connectivity
PN	- ProfiNet
CPU	- Central Processor Unit
DI	- Digital Input
DO	- Digital Output
AI	- Analog Input
PS	- Power Source
DP	- Decentralized Periphery
IP	- Internet Protocol
HW	- Hardware
OB	- Organization Block
FC	- Function
SFB	- System Function Block
SFC	- System Function
DB	- Data Block
TCP	- Transmission Control Protocol
ACK	- Acknowledge
RAM	- Random Access Memory
EPROM	- Erasable Programmable Read-Only Memory

LIST OF SYMBOLS

T_{celk}	- Total response time of remote system
T_{poz}	- Requirement delivery time
T_{sprac}	- Requirement processing time
T_{odp}	- Response delivery time
T_{cyklu}	- Total time of PLC cycle
T_{vstup}	- Input processing time
$T_{program}$	- Program execution time
$T_{výstup}$	- Output processing time
P_{celk}	- Overall PLC memory consumption
P_{prog}	- Program memory consumption
P_{prac}	- Work memory consumption
P_{syst}	- System memory consumption

INTRODUCTION

Distributed management was not taken into account by control system design. Therefore even the standards which these systems follow do not offer direct options for how such systems should be designed. They have come more and more to the forefront in the present day. Based on this fact, it is necessary to create new standards, which are primarily designed for distributed system development.

When creating the distributed control systems, we can choose one of two possible approaches based on IEC 61131 and IEC 61499 standards. The first of them has been used for a long time in the PLC sphere, but it does not directly support the creation of extensive distributed systems. That is why it is quite difficult to design such systems using the IEC 61131 standard. The more difficult the system is, the more problems that can occur. Sometimes it can be quite impossible to create a system because it is just too big. More difficult operations obviously increase the costs needed for system design, in some cases they can even exceed the hardware costs. The second of the standards, IEC 61499, offers a direct solution for distributed control system creation. That is the reason why it is more widely preferred. The truth is that it is a new standard, which is why it is supported by only a few device manufacturers. Most programming tools are intended for research purposes only. Also, most of the people who develop these systems do not have any experience with the IEC 61499 standard. This raises the question of which approach and which standard is more effective and suitable for individual system designing.

The publication focuses on:

- IEC 61131 and IEC 61499 standards analysis and prognosis of their development;
- Analysis of existing application of these standards in practice;
- Criteria determination to compare the suitability for system use, designed on the base of both standards;
- Preparation of the model distributed system by using both approaches;
- To compare the influence of the chosen approach on selected criteria;
- Comparing the suitability of both approaches for real distributed system creation based on quality parameters.

1. CURRENT STATUS IN INDUSTRIAL AUTOMATION STANDARDS

Each industry sector will soon come to the development point when it is necessary to unify different approaches. Without creating the standards, development cannot progress.

Usage of the standards enables the following:

- To reduce installation and startup costs – installation and startup of the systems based on the same standard is identical, or similar.
- To reduce the need to keep high level of stocks – standard devices are widely available even after several years from placing on the market
- Components compatibility – new device should be easily compatible with any other device built on the same standard
- Safety increase

Usage of the standards in industry:

- Improves communication
- Provides practical usage of professional skills
- Represents years of experience and reduces the need to start each project from scratch

1.1 IEC standards

The international industry community accepted that it is necessary to design new standards for programmable logic controllers. The International Electrotechnical Commision (IEC) was founded in order to design and standardize programmable controllers, including hardware design, installation, testing, documentation, programming and communication. As a result of its activities, IEC 61131 and IEC 61499 were designed as well, which together with new technologies have a dramatic impact and influence on design and implementation of the industrial control systems.

These two standards are closely connected and create the base for control system development in the present, as well as in progressive technologies in the near future.

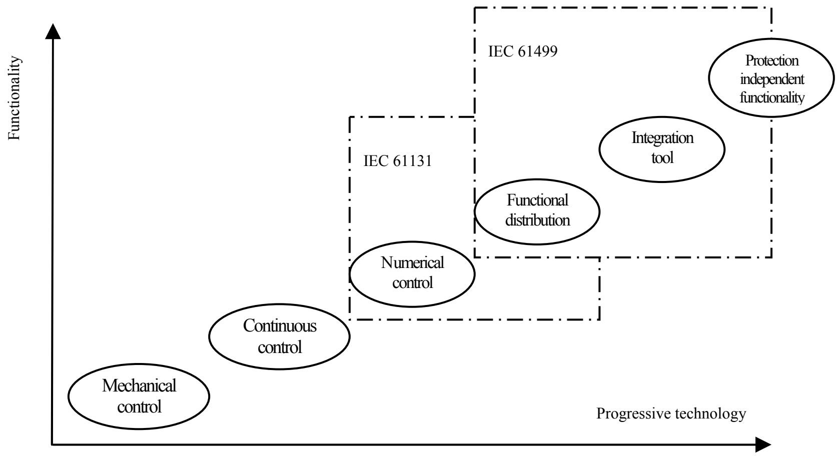


***Fig. 1** Technology development for industrial control [24]*

1.1.1 IEC 61131 standard

IEC 61131 standard is based on well-tested techniques [25], which are at present used in various forms and in many control products. It comprehensively defines the whole software design process for programmable logic controllers, possibly control systems [11], including programming languages, implementation, communication and technical documentation.

This standard is used for the whole complex of problems related to programmable controllers and it is devided into 8 parts:

- IEC 61131-1 Basic information – basic terminology and terms are defined in this part;
- IEC 61131-2 Hardware requirements and its testing – electronic and mechanical structure and verification tests;
- IEC 61131-3 Programming languages – structure of PLC software, languages and program performance;
- IEC 61131-4 User instructions – instructions for selecting, installation and maintenance of programmable controllers;
- IEC 61131-5 Message specification – software resources for communication between devices based on MAP (Manufacturing Messaging Services);
- IEC 61131-6 Communication through industrial networks – PLC software resources for communication using IEC Fieldbus;
- IEC 61131-7 Fuzzy control programming – software resources including standard; function blocks for operation with fuzzy logic in PLC;

- IEC 61131-8 Directives for languages implementation used for PLC – application and implementation directives for IEC 1131-3 languages.

1.1.2 IEC 61499 standard

IEC 61499 was developed especially as a methodology for distributed control system modelling. It defines concepts and models in a way where software implemented in the form of mutually concurrent function blocks can be used to define distributed control system behaviour. Function blocks within the industrial systems represent the basic concept for definition of robust, reusable software components. Each function block has its own set of input parameters, which are evaluated by internal algorithms during processing. Algorithm results are recorded in block outputs. Complex applications can be created from function block networks, which are formed through their input –output connection.

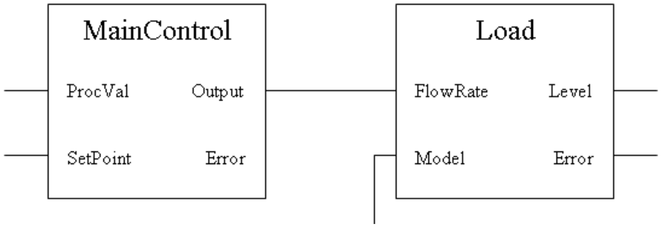


Fig. 2 *Function block connection*

IEC 61499 standard, based on the function blocks concept, which are defined in the language standard of the programmable controllers IEC 61131-3, provide together with the application layer of the communication fieldbus, the software interface. This interface enables cooperation between remote function blocks by using an industrial network.

IEC 61499 standard consists of 2 parts. Firstly, it includes the function block architecture and concepts for block–oriented systems design and modelling. This part includes the following spheres:

General requirements – contain introduction, focus and normative references (for other standards as well), definitions and reference models:

- rules for the function block type declaration and the instances behaviour of function block types,
- rules for function block usage in configuration of distributed industrial control system (IPMCS),
- rules for function block usage in connection with communication requirements of distributed industrial control system,
- rules for function block usage in application, resource and device management in IPMCS,
- Requirements for system and standard compatibility.

Second part defines software tool requirements for definition design of the function block types and management of the function block libraries. It contains an extending supplement for type definition of XML documents used for the function block design exchange between software tools from different suppliers. This part focuses mainly on design, implementation, usage and maintenance of industrial distributed control systems constructed with architectural usage and concepts defined in the first part. The XML language is used to store and access the function block definitions, which enables design transmission through the internet and offers an opportunity to browse them by using internet browsers and save different attributes, including the information about the version and allocation of the graphic details.

Faced with a large number of industrial standards remains the issue of which norm or standard is the best solution for our application. Each standard contains precise rules for system creation, but only in theory. It would be a waste of time to look for a specific recommendation for real systems usage in the standards. When creating the distributed control systems, there are several alternatives [24] for how to proceed. It is not simple to select a specific solution without the knowledge of appropriate standards and issues.

2. IEC 61131 AND IEC 61499 STANDARDS AND THEIR FUTURE DEVELOPMENT

IEC-61131 [6] standard for PLC programming was widely adopted by suppliers and users of automation technologies. However, IEC 61131-1 restrictions play more and more of an important role for achievement of automation systems' elasticity and flexibility. Although all major PLC suppliers design their product according to the IEC 61131 standard, some differences between the various PLC brands are the reason for not allowing the replacement of one PLC brand with another one. The need for decentralized management is growing for many reasons such as flexibility and reliability (a centrally-managed system represents a single point of failure, in case of failure of any system part). The International Electrotechnical Commission (IEC) developed a new standard, IEC 61499, in order to make the automation systems reconfigurable, interoperable and portable. IEC 61499 standard is based on function blocks conception. Function block type IEC 61499 is in contrast to the event driven function block type IEC 61131. IEC 61499 is an innovation, but has great potential for future growth. Modularity and re-usability are currently more and more important for PLC programming. Modular code offers greater efficiency and time saving, especially for medium- and small-sized businesses. Productivity is much more improved thanks to the introduction of IEC 61499 function blocks. Small companies will be able to incorporate their intellectual property collected in IEC 61499 functional blocks into components libraries used in the future.

For configuration according to IEC 61131-3, the program implementation is processed by periodical or non-periodical events. When activated, function block instances are performed in a pre-defined order. Function block instance activation in the distributed system can be carried out according to a predefined cyclical plan as well. However, IEC 61499 supports a more generalized model, where the central mechanism provided by the activation under the plan, cannot exist at all. Function block execution is controlled by events and is very fast, but can be cyclical as well. This is important for many control programs in modern devices.

Communication and input-output functions in the IEC 61131-3 model are only loose-bounded to the variables, which are called in program by the "access channels" mechanism and "global and direct represented variables". There are no communication function blocks available between systems for industrial fieldbusses. OPC (Object language embedded for Process Control) is used to create communication links independently from the application. IEC 61499 contains communication function blocks, which can be implemented very easily and support different network access, so the function blocks can be easily distributed to network devices. Function block model according to IEC 61499, allows the existence of several alternative algorithms in the function block body, which are selected according to defined external events or conditions (e.g. initialization algorithm, standard algorithm and algorithm called when failure or defect occurs). All this is possible because, IEC 61499 function blocks can be designed by different programming languages. There is only limited flexibility in the case of fieldbus, because only basic function blocks can be used.

As the production management becomes more and more distributed, the attributes like encapsulation and the function block reusable possibility by end users, device suppliers and system integrators, become increasingly important and widespread. As an extension of the programming languages IEC 61131-3 supports, IEC 61499 not only uses the algorithm encapsulation, but subprograms or even system applications as well. The system designer, without any programming experience, is able to design applications and use them again in the future. Such technology does not exist for use of industrial fieldbus.

Because of adaptation to frequent changes in product composition and volume, as well as the frequent introduction of new technologies, the industrial processes will offer more opportunities for physical reuse in the future. There are no doubts that thanks to the use of the IEC 61499 [58] concept, the technical costs will be reduced and systems will be more flexible and better maintained.

2.1 IEC 61131

As defined in IEC 61131-3 [6], standard programming languages for PLC are: Structured Text (ST), Ladder Diagram (LAD), Instruction List (IL), Sequential Function Chart (SFC) and Function Block Diagram (FBD) [18]. PLC manufacturers implement programming languages on the base of IEC 61131-3 standard, but not equally. This causes compatibility problems. For example, a program in LAD language from a certain manufacturer, although seeming similar, cannot be imported to PLC from another manufacturer.

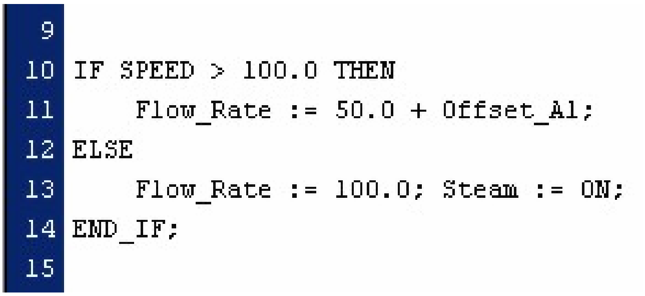
Function block IEC 61131-3 is a subprogram with parameters and local data. But the syntax of specific implementation [1] can contain many different details. Function block does not need to contain all programming languages (e.g. SFC is not supported in add-on instructions of Allen-

Bradley, Implementation of Rockwell function blocks). Access to memory also varies between individual manufacturers. Some PLC support only variables (attributes) assigned to a block (local attributes), while others can access the global variables as well.

Languages IEC 61131-3 [12]:

a) text

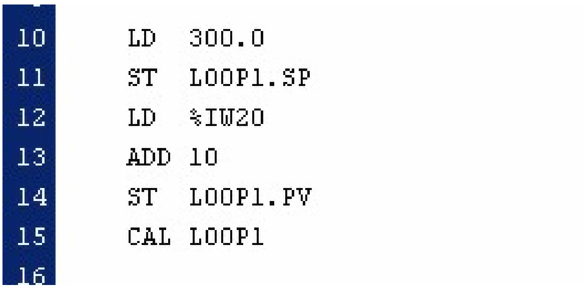
- Structured Text Language (STL) – high-level programming language supporting structured programming. It has strong language structure similar to PASCAL.



```

IF SPEED > 100.0 THEN
    Flow_Rate := 50.0 + Offset_A1;
ELSE
    Flow_Rate := 100.0; Steam := ON;
END_IF;

```

***Fig. 3** Partial program record in STL*

- Instruction List (IL) – low–level programming language similar to languages used in modern PLT controllers.



```
LD  300.0
ST  LOOP1.SP
LD  %IW20
ADD 10
ST  LOOP1.PV
CAL LOOP1
```

Fig. 4 *Record of program function block call in IL*

b) graphic

- Ladder Diagram (LAD) [59] – graphic language based on relay logic, technique used for programming in present generation of PLC controllers.

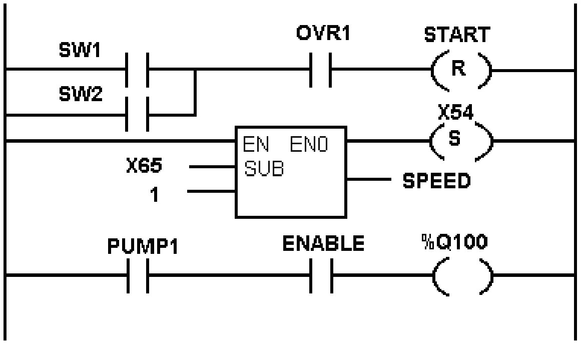


Fig. 5 *Program flow record in LAD*

- Function Block Diagram (FBD) – language describing signals and data flows through functional block links – reusable software elements.

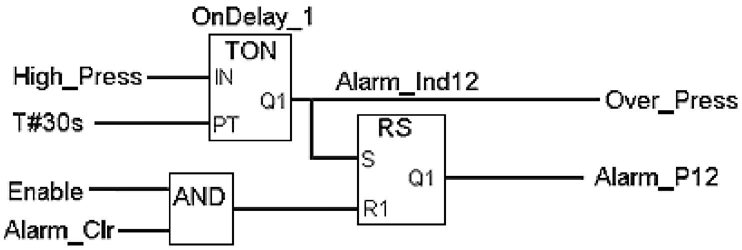


Fig. 6 Program flow record in FBD

- Sequential Function Chart (SFC) – graphically illustrates sequential behaviour, control program sequence running. It is used to define time or event based control sequences.

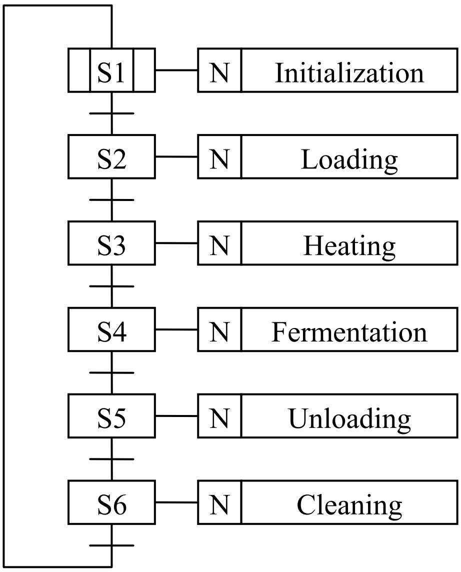
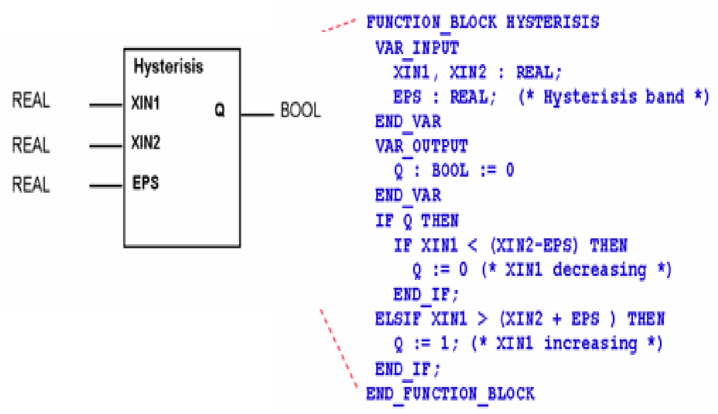


Fig. 7 Record by using SFC

One of the main features of this standard is to use function blocks - parts of the control program [14], which are limited so they can be used in different parts of the same program, or in other programs or projects as well. Function blocks can store data or algorithms. That is a great advantage over other similar concepts in programming languages (e.g. FORTRAN, C). The function block describes the data flow as well as its structure.



```
FUNCTION_BLOCK HYSTERISIS
 VAR_INPUT
   XIN1, XIN2 : REAL;
   EPS : REAL;  (* Hysterisis band *)
 END_VAR
 VAR_OUTPUT
   Q : BOOL := 0
 END_VAR
 IF Q THEN
   IF XIN1 < (XIN2-EPS) THEN
     Q := 0 (* XIN1 decreasing *)
   END_IF;
 ELSIF XIN1 > (XIN2 + EPS ) THEN
   Q := 1; (* XIN1 increasing *)
 END_IF;
END_FUNCTION_BLOCK
```

Fig. 8 *Example of function block definition*

Main parts of the IEC software model are pictured in Figure 8. These parts are needed for the PLC software environment. The model consists of several layers, each layer absorbs features located inside of it.

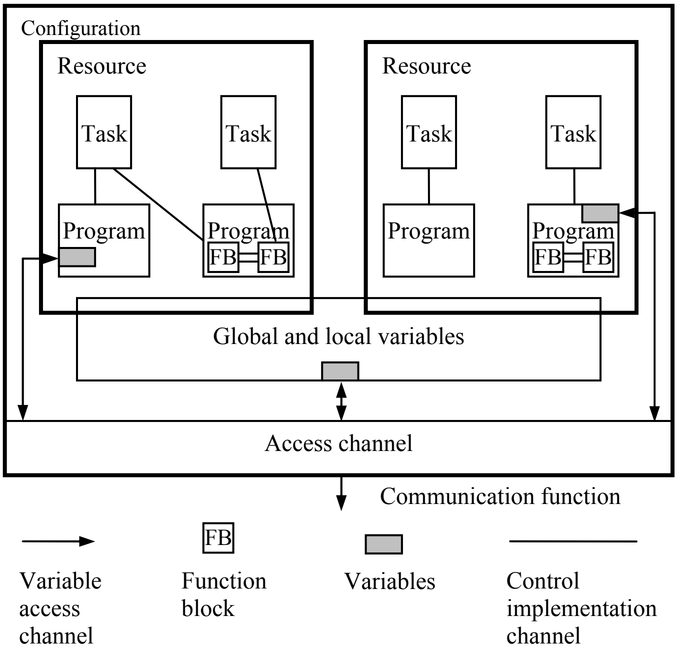


***Fig. 9** IEC 61131-3 software model*

The main elements of the IEC software model are:

a) *Configuration* – is a top layer defined as a language element, which corresponds with a programmable control system.

b) *Means* – in each configuration are one or more resources, which provides support for all elements needed for program execution.

c) *Program* – can be created by multiple different software elements, each of them can be written in one of the different languages listed in IEC. Program consists of a number of interrelated function blocks, which enable data exchange through software connections. Program can read and write the input-output variables and communicate with other programs.

d) *Task* – can be configured to control the program group or function blocks, which are executed, based on periodicity or event.

e) *Function block* – This concept is one of the most important elements of IEC 1131-3 standard for software support of the hierarchical design. Using the function blocks, it is possible to create a program from smaller, easily controlled blocks. Each block can be programmed through using other function blocks - clear, hierarchically structured programs can be created this way. Function blocks in distributed systems form the base for the IEC 61499 standard. Function blocks consist of two parts: data, which defines the inputs and outputs and the part where the algorithm is defined; code time, which always runs during function block execution.

f) *Functions* - are software elements that provide one result based on required inputs.

g) *Global and local variables* – standard allows declaring of variables in different software elements. Global variables are available to any software elements inside the program, while the local variables are available only for included software elements.

h) *Accurately representing variables* – enables the data to be recorded in and read from known memory area in PLC.

i) *Access channels* – are the last element of the model. They provide resources for data and information transmission between different configurations. Each configuration contains variables that are accessible to other remote configurations.

The concept of the application itself [1] [27] [31] [38] is not defined by the standard. Application must contain all control requirements for activation, control and finish. Control is necessary for physical sensors and actuator handling, event scheduling, and interaction with operating terminals. Because the standard allows the resources to be activated independently, large-scale PLC system can handle multiple independent applications simultaneously (Fig. 10).

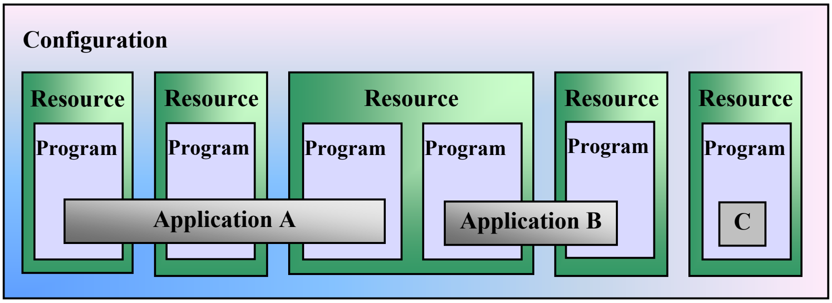


***Fig. 10** Assigning application to resource*

Standard [25] describes programs, function blocks and functions as program organization units – software elements, important attributes that could be called in different application parts. Behaviour and structure of such program organization units is defined by type declaration. The function blocks copies made from particular types are identified as function block instances.

One of the important characteristics of IEC 61131-3 standard is a strong emphasis on the hierarchical design of the application. A program can be seen as a network of function blocks and functions. Each function block is a copy or an instance of the type definition. (Figure 11).

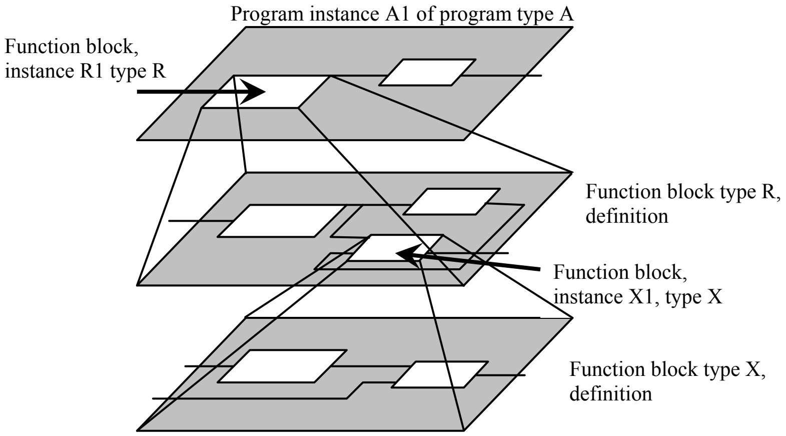

Program instance A1 of program type A
Function block, instance R1 type R
Function block type R, definition
Function block, instance X1, type X
Function block type X, definition

Fig. 11 *Hierarchical structure decomposition*

2.2 IEC 61499

IEC 61499, IEC introduced in 2005, is considered to be the next generation in improvements for PLC system engineering. The basic principle of IEC 61499 consists of event controlled function blocks (FB), which are called only if some of their event inputs are activated. Relevant data inputs are updated at the same time. FB stays inactive during the remaining operating time. This will significantly improve efficiency, reducing energy consumption and workload of the communication channel. Also, use of function blocks as a top-level representation provides a complete system overview of all devices, communication layers and programs. It is very easy to transfer the project with function blocks to other devices. Function encapsulation into function blocks increases the possibility of reuse. As a result, the library of standard component completed from function blocks maximizes the efficiency during project revision.

There are three types of function blocks [15]: basic, composite and service. Basic function block is an elementary component in the IEC 61499 function block system. Basic block must contain Execution Control Chart (ECC) [60] – state controller with conditional jumps and relevant algorithms running on conditions. IEC 61499 function blocks have two kinds of input and output based on event or data. Function block is performed only if the event input is activated. All associated data inputs and outputs (assignment is marked as a vertical line connecting event and data input/output) are updated at the same time. According to the definition in IEC 61499, function block inner algorithm can be written in several languages, for example in languages included in IEC 61131 (ST, LAD, IL, SFC, FBD) [55] and in high-level languages as well (e.g. C or Java).

A network of basic and composite function blocks creates a body of composite function blocks. Hierarchical structure, which assures the code's reuse, can be built in this way. The composite blocks have their own interface as well as the basic function blocks. Inputs and outputs of the composite function block can be directly connected with inputs (outputs) of the composite function blocks.

The result is an application formed by a network of basic and composite function blocks. System configuration combines application logic with the device topology, abstract definition of communication networks and accurate assignment of function blocks to the devices. Service interfaces [6] are designed to cover hardware dependencies of the applications. Service interface is a "Black Box". Internal logic definition of the service interfaces is not limited much by the standard. Service interface is defined by multiple sequence of event specifications, which describe interaction between hardware resources and function blocks. This method of specification can be useful for documenting the service interface operation, especially if this interface is hidden, or created in low-level programming languages. Service interfaces can be used for implementation of various communication protocols, database interfaces or Human Machine Interfaces (HMI).

IEC 61499 standard defines the basic model and methodology for a function block describing in form, which is independent from implementation. Standard is the first step in providing design methodology for distributed application development and modelling. Methodology can be used by system designers to build distributed control systems. System is defined in terms of logically connected function blocks running on different process resources. Use of the IEC 61499 standard consists of two stages in distributed control system designing:

- function design phase – function requirements are represented as a series of blocks, defining the basic features of software components and their main connections.
- functional distribution phase – is necessary to define distribution management to process resources. Standard provides models and concepts to define the distribution functionality of the function blocks into connected function blocks.

Standard enables the function blocks, which include software functionality and algorithm to be defined in a standard form. Standard defines the range of the communication blocks as server and client blocks, which are used to formalize data exchange between blocks in physically different process resources.

Programming using this standard requires similar use of function blocks as objects in object-oriented software programming, which model entity and concepts behaviour in real-world settings. That is why they share many advantages from software object use:

- objects reflect the real world - during design the applications are more natural and intuitively display the real world entities associated with applications as objects.
- Objects are stable - objects demonstrate software elements that are not changing significantly. In many cases identical object classes within a wide application range are developed and used.

- Objects reduce difficulty – programmer can work with objects without knowing and understanding how these two objects operate from the inside. Application is being developed through creating and linking references – it is not necessary to understand the objects in detail, specifically their inside.
- Objects are reusable – immediately after object development and testing, it can become a part of the implementation tools, or library.

It is a great advantage [16] for system developers and end users because function blocks share many of these features:

- control software range developed for the application is reduced by functional block usage
- time needed to develop management systems is minimized
- control system using identical types of the functional blocks shows more consistent behaviour
- control system quality improves

Usage of the tested and validated blocks speed up the application testing in FAT or SAT.

2.2.1 System design

Designing the software for any project can be very difficult and it includes multiple aspects of distributed management, including the software running on different software resources. There is a requirement for multiple graphic design previews, which allow it to be defined and analysed in terms of other aspects. Some of the previews express abstract design aspects, some are needed to see the system's physical structure, or its software organisation. The most complex design requires at least four different design previews and set of scenarios. This architectural form is known as 4+1 preview model, which is used for design previews of object–oriented software and is equally applicable for creating the same design previews used for distributed control systems (Figure 12).

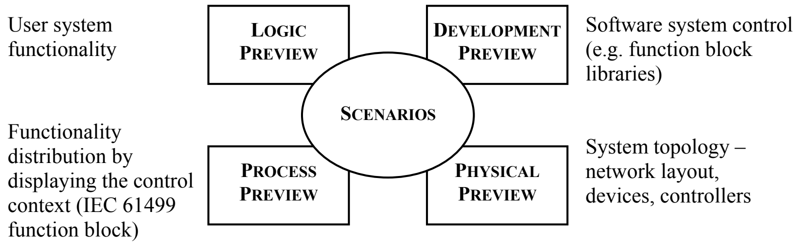

User system functionality
LOGIC PREVIEW
DEVELOPMENT PREVIEW
Software system control (e.g. function block libraries)
SCENARIOS
Functionality distribution by displaying the control context (IEC 61499 function block)
PROCESS PREVIEW
PHYSICAL PREVIEW
System topology – network layout, devices, controllers

Fig. 12 *4+1 System development preview model*

Logic preview

This design view is used to show function requirements of the system. It expresses the software functionality requested by system user. Major software function blocks and major interfaces between them are reproduced in distributed system design.

Process preview

Process preview contains many non-functional system requirements, including performance or system distribution. During application of IEC 61499 standard, which provides the architecture showing the implementation preview of the distributed system, a network of interconnected function blocks is created.

Development preview

Preview shows the organisation of developed software integrated into a larger system, relations between software components. Creating a large scale distributed control system includes a number of software libraries and modules.

Physical view

It shows physical devices in distributed systems and controllers within the system and reflects different network communication connections between them.

2.2.2 Models and concepts

The main objective of IEC 61499 is not the programming methodology, but the architecture and model for distributed systems. The standard provides concepts and a set of models describing distributed systems programmed by using function blocks, which describe the implementation of such control systems in an explicit and formal way. To have a formal and standardized approach for system description allows systems to be verified, compared and understood.

The standard uses few models which together form the architecture of a distributed system oriented on function blocks.

System model

Defines relations between communication devices and applications. Application can exist on a single device or can contain functions distributed for a larger number of devices. Distributed application is shown as a function block network.

Device model

Device is able to support one or more resources. The resource of IEC 61499 has similar features as resource defined in IEC 61131-3 standard. The resource provides independent performance and function block network control. The model contains the device interface which provides services that allow resources to exchange data with input-output points on the device and communication interface providing communication services for data exchange through external networks with remote device resources.

Resource model

The resource provides equipment and service for performance of single or multiple function block application fragments. Function blocks of distributed systems are placed into device resources, which are interconnected. Resource provides interfaces for communication systems and for specified process devices. Important feature of the resource is its support for independent execution – resource can be recorded, configured, activated or stopped without influencing other resources in the same device, or network.

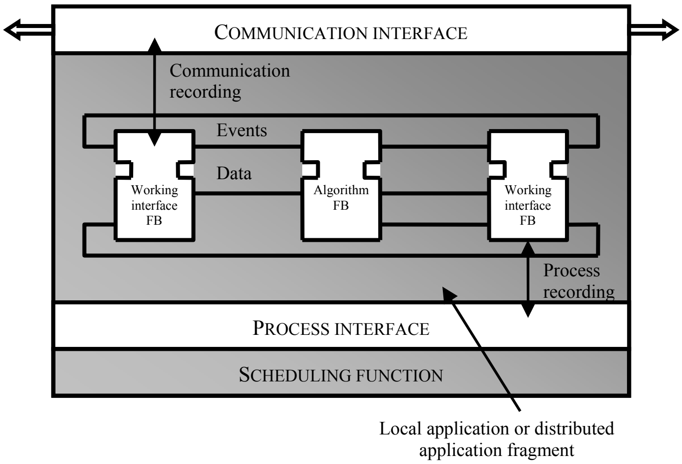


Fig. 13 *Resource model*

Application model

IEC 61499 application is defined as a network of interconnected function blocks depending on event or data flows. Application can be divided and distributed in multiple resources. Further decomposition is possible inside of the application by using the subprograms. The application defines relationship between events and data flows, which are required between different blocks. The resources on which the function blocks are distributed must ensure that the events are used for particular algorithm timing in function blocks with correct priority and time. Resources are responsible for accumulation of variable values in function blocks between application callings. The application contains function block instances and connection definitions.

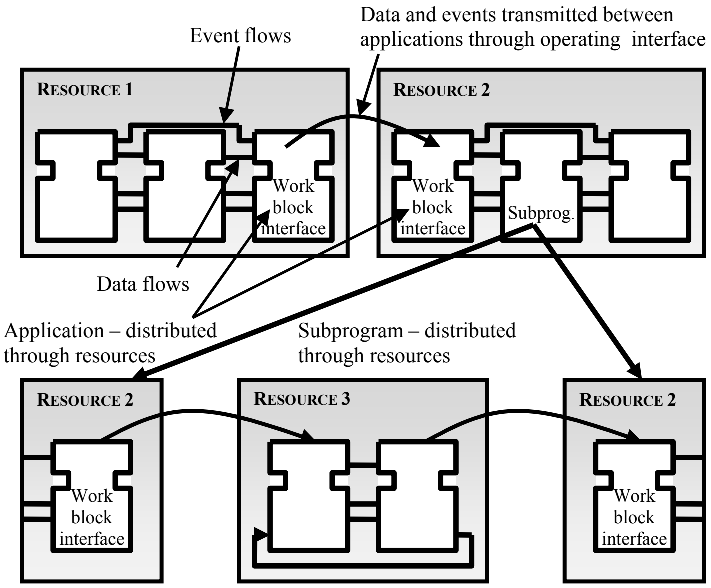

Event flows
Data and events transmitted between applications through operating interface
RESOURCE 1
RESOURCE 2
Work block interface
Work block interface
Subprog.
Data flows
Application – distributed through resources
Subprogram – distributed through resources
RESOURCE 2
RESOURCE 3
RESOURCE 2
Work block interface
Work block interface

***Fig. 14** Application model*

Function block model

Definition of the function block type provides formal description of the data structure and data assigned algorithms, which exists in different instances. Function block consists of two parts:

- "Execution management" – top part of the function block records events in algorithms, provides information about the status between input, output and algorithm execution events.
- Bottom part of the function block contains algorithms and internal data, which are hidden inside of the function block. Function block is

a software component type and if it is properly developed, it should not request detailed understanding of its internal design from the user.

- Function block operates in connection with its resource, which supplies resources for algorithms scheduling and requests recording for communication and process interfaces.

An important concept in IEC 61499 is the option to define the function block type [24], which determines the behaviour and function block instance interfaces, which are built of that type.

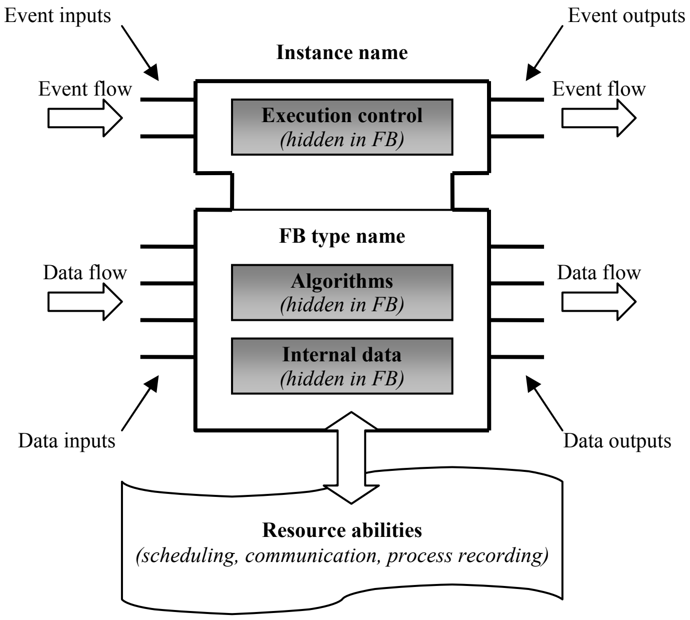


Fig. 15 *Function block model*

2.3 PROFINET CBA

It can be described as a transfer of the IEC 61499 standard elements into the world of traditional PLC systems. The main element of PROFINET CBA [2] [35] [36] [37] [60] is a component. Component represents control functionalities and communicates with other components through its interface. In the PROFINET environment, each component equals the control device. PROFINET CBA enables the creation of component interfaces and relations between them. Individual components are programmed with tools supplied by device manufacturers. Although several companies adopted PROFINET CBA (Hilscher, KW Sotfware), the market accepted PROFINET CBA only on a limited scale, mainly because this system is complicated to use and some key features of IEC 61499 are missing (e.g. application model and composite function blocks).

PROFINET is an open standard for automation based on industrial Ethernet. PROFINET CBA is a part of this standard and is based on IEC 61499-1 – describes technologies for modular and distributed system implementation on the base of pre-defined components.

"Architecture Description and Profibus Specification" issued by Profibus International, defines the communication between devices from different manufacturers, automation and design model based on IEC 61499-1. The aim is to implement the solutions of distributed management by use of unified communication on Ethernet network and industrial networks using the open standards.

Distributed control systems in accordance with IEC 61499-1 standard have a hierarchical structure and are not based on components. Basic terms are "System", "Device", "Resource", "Application" and "Function block".

Specification of PROFINET CBA is largely in line with the model described in IEC 61449-1.

BASIC EXPRESSIONS IN IEC 61499-1
AND PROFINET CBA Table

IEC 61499-1	**PROFINET CBA**
System	System
Device	Physical device
Resource	Logic device
Function block	Object/Control function
Application	Application
Conjunction	Connection

System

System model describes distributed control system as an entity. System is the top level in architecture hierarchy. It consists of physical devices that are interconnected through communication network. Current process is controlled by several applications, which are located in a single device (Application C), or are divided among several devices (Application A and B).

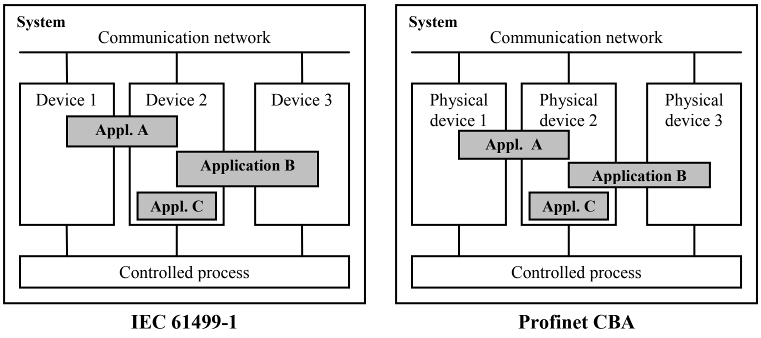


Fig. 16 *System model according to IEC 61499-1 and PROFINET CBA*

Device

Device performs within the complex control task as an independent function. It is a general term for various device types, starting with PLC, from personal computers to strong programmed devices (embedded regulators), or intelligent fieldbus devices with specific firmware. All of them contain a process and communication interface. Devices consist of single or multiple resources.

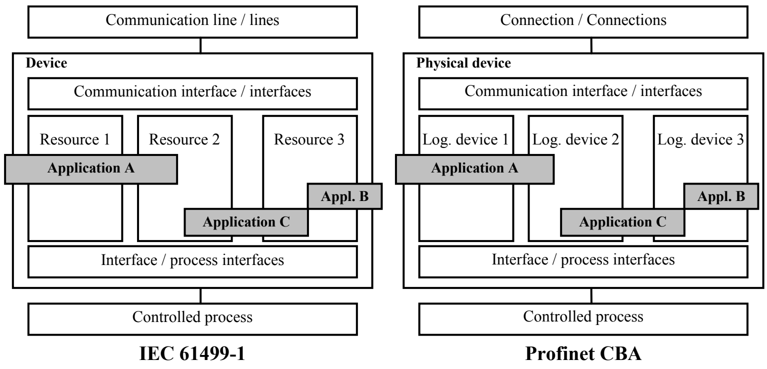


***Fig. 17** Device model according IEC 61449-1 and PROFINET CBA*

Resource

The resource performs local applications, also known as function blocks. These represent the wrap around software needed to run the equipment. This software includes firmware as well as the control program, which can be freely programmable. Functional blocks are capable through the connection of exchanging data with other function blocks. Communication partners can be in the same device, or located in other devices in the communication network.

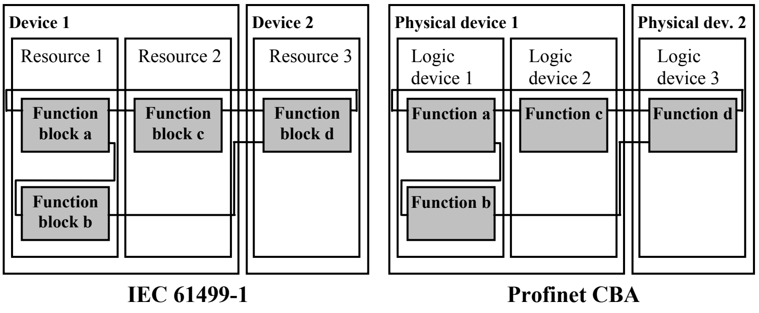


Fig. 18 *Resource model according to IEC 61449-1 and PROFINET CBA*

Function blocks

Function blocks are the basic architecture elements of IEC 61499-1. They contain interfaces for data receiving and transmitting, as well as containing internal data and executable algorithms invisible from the outside. Technological functions of PROFINET CBA are based on the application oriented Service Interface Function Blocks (SIFB) described in IEC 61499-1 standard.

Function blocks [32] have a fixed structure, or they are freely programmable.

As a result of that, these devices can have either:

- fixed functions (e.g. fieldbus devices, drivers, sensors), or
- programmable and recordable functions (e.g. PLC, personal computers).

2.4 Future development of IEC 61499 standard

Considering the entire life cycle of the automation systems, current work on IEC 61499 deals with solving only its small part, which are the creating and activating aspects. These are very important aspects, but for pioneers of industrial automation technology it is also important that this system is fully functional, being easily kept in operating status and in case of any problems (e.g. breakdown), the system can return back to operating mode immediately. To overcome these problems, research and development will be necessary in the following areas:

- How to perform the activation and deactivation phase of the distributed system.
- Distributed control systems monitoring and debugging – how to gain data, how to present them in the tools and how to detect defects.
- So far, all sample applications have from one up to four devices. How will the system behave with many nodes (thousands of nodes)?
- Analysis, behaviour tests and system stability tests (e.g. one node will always be inactive).
- Managing the complexity of distributed systems. Distributed systems are more complex than centralized systems. They need methods and tools to help the system designers manage their complexity.

Not completely related to the device operation are the intended heterogeneous targets of the IEC61499 system. Subject to the requirements of accuracy, interoperability and configurability appear as a problem of how to activate control programs on devices from different producers. The reason for these gaps and deficiencies in the standard can be interpreted differently.

The world of industrial automation and management has changed a lot since works on the IEC 61499 standard began. Distributed devices, programmable input/output modules and intelligent drives are more used in the field of automation systems. Increasing their computing power and increasing memory allows the control programs to maintain a reasonable size. Ethernet in industrial and automation practice is another popular technology of recent years. This technology allows any member of the communication system to communicate with another. Another element is the growing complexity of the control programs. Large modular devices consist of 60 or smaller control devices (such as woodworking and printing machinery). There are even more devices in the case of the building automation. Currently available development methodologies and tools cannot handle such a complex device. The main reason is that the IEC 61131-3 standard is designed for central tightly- bound control systems. Therefore, it is difficult to build and keep the distributed systems in operation. Software costs grow rapidly and exceed the price of mechanical parts of the device.

IEC 61499, as an architecture focused on distributed control systems, can solve the majority of such problems. Basic knowledge of IEC 61499 must be put under constant technology change and development in area of industrial automation and control systems. Otherwise the IEC 61499 standard would be a nice concept without longevity. Vyatkin came to the same conclusion in his essay on the use of IEC 61499 [57]. He also mentioned the large increase in the number of published works on this subject in recent years. This may be a sign of future recognition of the IEC 61499 standard.

One possibility for how to get the products supporting IEC 61499 on the market is to help the producers integrate this technology into their products. As a result, an open source initiative named Open Distributed Embedded Control Environment (ODECE) [60], currently operating under the auspices of the OOONEIDA society, was founded.

The next chapter lists the best known programming tools implementing the IEC 61499 standard, as well as their brief descriptions and future development prognosis for this standard.

3. PROGRAMMING TOOLS SUPPORTING IEC 61499

IEC 61499 has been practiced in research projects for years. Several research groups [55] around the world participated in case studies research as well as supporting tools prototypes.

Several programming tools are used according to IEC 61499 standard (e.g. FBDK, ISaGRAF, Fbench).

3.1 FBDK

FBDK is the first programming tool which follows IEC 61499. It is written in the Java language and function blocks are implemented as Java classes.

FBDK [40] consists of a development environment (FBEditor) and tool for triggering (FBRT). However, usage of Java, even if it is beneficial for portability, is not in any way conditioned by actual standards. FBDK was developed by one of the main suppliers of the industrial automation systems, Rockwell Automation, and played a very important role in the evaluation, development and teaching of the IEC 61499 standard. It was never intended to be used in "field" and it´s missing many functions of a modern automation tool. FBDK is mainly used in academic environments and development communities.

This development software [52] allows a management engineer to create and test data types, function blocks types, resources, devices and system configurations according to IEC 61499 standard. It also offers application–oriented development, has an extensible component software library and allows the recording of the control program to different devices.

FBDK is at present managed by the company, Holobloc Inc., which provides user training, references and industrial process, measuring and regulatory systems based on IEC 61499 standard.

3.2 ISaGRAF

The last ISaGRAF 5 version was in addition to its opportunities with IEC 61131-3, ISaGRAF [55] is the first full-valued automation product that supports the entire design process. The vendor of this product, ICS Triplex ISaGRAF, actively promotes this product on the market. It is based on a well-known IEC 61131 approach.

ISaGRAF was the first development environment that supported all 5 programming languages [30] following the IEC 61131-3 standard for Windows. Moreover, to achieve the highest performance and flexibility, it supports ISaGRAF functions and function blocks written in C language and languages listed in IEC 61131-3.

Extended to support the first set of IEC 61499 models. We speak about basic and composite function blocks. These blocks can be used in programs that are present in resources of various ISaGRAF devices [52] within a distributed automation system. While offering so-called application overview that shows where the individual application parts are located in the system, these must be programmed directly on individual devices. Communication between the program parts is done by using network variables. Another issue is that the events triggered function blocks of IEC 61499 are executed at the end of the cyclically triggered IEC 61131-3 system, resulting in long delays in triggering and that is why the IEC 61499 application performance is rather low.

3.3 FBench

FBench [6] is an open source project initiated by the Canadian non-profit organization, OOONEIDA. FBench is able to design, develop, debug, activate and verify the IEC 61499 application. The research group at the University of Auckland continues with the project. FBench aims to be a complex tool, which supports programming languages of both IEC 61131 and IEC 61499 standards.

As defined in IEC 61499, to program the function block internal logic any high-level programming language may be used.

In contrast to design and simulation functions contained in FBDK, FBench contains tools for local and remote program debugging and project environment to accelerate function blocks development. In addition, FBench project development architecture contains the possibility of using plugins [9], which allows third party developers easily expanding on FBench abilities without knowing details of its implementation. FBench is developed in Java as an open-source project according to CPL (Common Public Licence).

The aim for the future is to add new functions in FBench [8], which will create a research environment of an industrial scale. Among them is a possibility of a formal validation of the designed function block system by using NCES (Net Condition/Event Systems). There is no function block editor so far that would be supported by this technology.

3.4 Tool implementing PROFINET CBA

Component Based Automation is an implementation of Profibus CBA [32] for control systems Simatic S7 and Simatic NET. These are the products according to the configuration:

- Simatic Step7 as a development tool for configuration and programming of Simatic S7 and Simatic NET control systems and also for creating PROFINET components.
- Simatic IMap as a development tool for distributed systems configuration and integration for specific program devices, configuration and diagnostic tools into the PROFINET CBA development environment.
- Simatic NET OPC Server PN for access to process and HMI data by using OPC interface.

Engineering tool Simatic IMap is designed for communication configuration between components from different manufacturers. This means a great advantage for distributed automation solutions with smart, programmable fieldbus devices. Program encapsulation or their parts is performed by Simatic Step 7. Simatic IMap meets the PROFINET standard, which means it can be used by different manufacturers. PROFINET components, which were created by programming tools from other manufacturers and which communicate through ethernet, can be imported by using open interfaces.

Characteristics of Simatic IMap:

- Simatic IMap built on an open component-based PROFINET architecture.

- Each intelligent machine/operation is represented by PROFINET component in connection editor. It appears in software function form.
- Simatic IMap combines elements of technology - oriented libraries, regardless of the manufacturer or functionality.
- On-line functions and communication function diagnostic possibilities make the startup more simple.
- PROFINET components can be used several times in Simatic IMap (library components reuse) but only once, if they were designated as so-called "unique".
- Machine/operation can be hierarchically structured to any depth.
- All variables which are necessary for general data access (e.g. visualisation requirements) are generated automatically from technical information.

Simatic IMap in principle includes the following views:

- Project tree – is used to manage all project resources (technological functions and devices) and to navigate within automated hierarchy of the operations.
- Technological library – technological software functions, which are necessary for the project. Library elements must be supplied by device manufacturers.
- Connection editor – is used to specify the data exchange between individual technological modules.
- Network and topology view – is used to assign each hardware devices in typology and for system diagnostics (communication and device status diagnostics).

- Project view – is used for project management and survey of software features used in the project.
- Graphic view – structured representation of the configurated operation. Procedure for project designing and launching with IMap [41] proceeds as follows:
- Creating the software components for each module of device/operation.
- Linking of the technology software components using the link editor.
- Configuration of assigned devices in the network typology.
- Recording the control programs and communication data into devices.

3.5 Future development of tools implementing IEC 61499

The first commercial implementations of the IEC 61499 standard raise the question of their subordination to the standard. Here are the outlines of the issues [55], which need to be cecked:

1. Syntactic consistency can be proved by direct comparison of standard syntactic elements with individual implementation structure.
2. Semantic consistency can be demonstrated by comparison with standard definitions, and if they are not sufficient, then with reference implementations.
3. Compliance with the design model application and workflow.

Although the IEC 61499 standard was implemented almost exclusively in research laboratories, a number of design models were designed and examined.

At the present time, a majority of these tools is used mainly for research purposes [6] and not for commercial use because they do not have sufficient support and are not compatible with most of the PLCs.

Because each implementation of the IEC 61131 or IEC 61499 standards has some weak points for distribution control system designing (IEC 61131 was primarily not intended for distributed systems designing and implementation of the IEC 61499 standard are still almost exclusively intended for research purposes), it is necessary to consider all important aspects and decide, which standard would be better to use for a particular distributed system. The issue of choosing the suitable standard is discussed in the next chapters.

4. METHODOLOGY PROPOSAL FOR COMPARISON OF THE APPLICATION STANDARDS AND CRITERIA SELECTION

IEC 61131 standard is still a base, on which all previously used control systems are built. This does not include creation of distributed control systems. Although individual producers solve communication between their own control systems, it is not a final solution for the problem with distributed systems design. Although such systems can be created, the programmer must solve many actions during software creation through creating their own program blocks and must monitor many aspects [3] [4] [19] [21] [22] [23] [29], which are ordinarily solved by the IEC 61499 standard, and increase demands on used components and provide much more work for the programmer.

The IEC 61499 standard is directly dedicated to creating distribution systems. But this is a new standard (work group, which started with the standard development was established in year 1990), which is not implemented in many cases by hardware manufacturers. Some companies have already started with the implementation of this standard, but so far we speak only about a few attempts, which do not solve all problems, and can occur during distributed systems creation. Also the use of software tools according to the IEC 61131 standard will increase total costs for system creation because the software tools of IEC 61131 are widespread and long used. This raises a question of which approach to use when creating the distributed control systems. By comparing various criteria from both systems, it is possible to determine which individual approach is suitable for distributed systems creation of different types and sizes. This chapter is focused on methodology constitution, which enables the comparison of both mentioned approaches during creation of distributed control systems and following a selection of suitable approaches for a particular system.

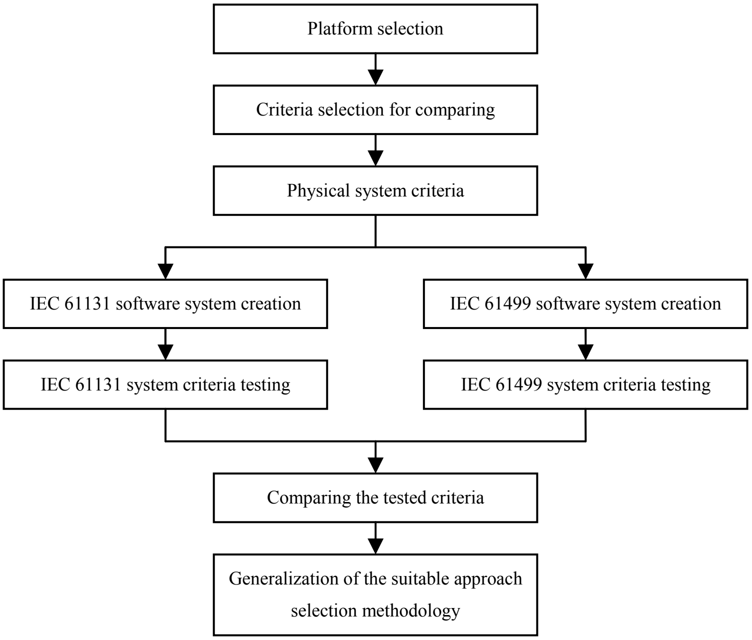

Platform selection
Criteria selection for comparing
Physical system criteria
IEC 61131 software system creation
IEC 61499 software system creation
IEC 61131 system criteria testing
IEC 61499 system criteria testing
Comparing the tested criteria
Generalization of the suitable approach selection methodology

Fig. 19 *Procedure for methodology formation*

4.1 Platform selection

One basic condition for an objective parameter assessment of the designed system is to use the same hardware components for both systems. That will assure that parameter can be influenced only by using different approaches during distributed system designing and programming. The majority of the tools for system development according to IEC 61499 are still dedicated to research purposes only and are not compatible with the devices operating under the IEC 61131 standard. It is necessary to choose system components, which are able to operate under the IEC 61131 standard or under the new IEC 61449 standard.

4.2 Criteria selection for comparing

Since identical devices and communication networks are used for both cases, the assumption is that the hardware performance of an entire distributed system will be the same (whether IEC 61131, or IEC 61499). Selection of programming approach will influence the communication speed and difficulty and amount of work as well, which must be carried out by the programmer in order to design the system. The following criteria will be influenced by approach selection for system creation:

a) **Response time of remote system** - response time represents the time when the remote station responds to a call from a local station.

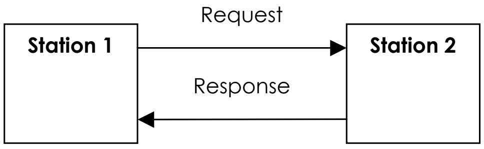


Fig. 20 *Communication between stations*

Can be expressed as a sum of communication time and time at which the remote station processes the request:

$$T_{celk} = T_{poz} + T_{sprac} + T_{odp} \qquad (1)$$

It is an important data piece, which corresponds with fast system response to the status change. It is affected by various hardware aspects, such as network complexity, used transmission medium, network utilization influenced by other devices, distance between stations, line speed and others. In addition, response time can be largely influenced by used programming approach and selection of software tools and technologies.

b) **Network utilization** – Network utilization will vary depending on the selected system. It represents amount of data transmitted through the whole network in one second. It is expressed in kB. Various technologies use various communication transmission protocols that are reflected in the data frame composition and in the quantity of transmitted data. Desired status is to reduce network utilization to a minimum possible value.

c) **Program influence on the PLC cycle time** - as visible in Figure 21, PLC executes all operations [7] cyclically.

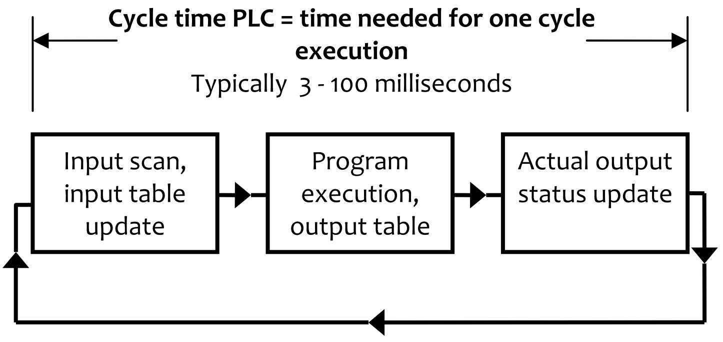

Cycle time PLC = time needed for one cycle execution
Typically 3 - 100 milliseconds
Input scan, input table update
Program execution, output table
Actual output status update

***Fig. 21** PLC cycle time*

First it sequentially scans input devices and updates the image of their status in the memory. Then all program blocks will be executed. After processing the PLC program blocks, the output device status image is updated in the memory. Finally, it will use the output status image for a real change in the status of outcome devices.

Cycle time can be expressed as:

$$T_{cycle} = T_{input} + T_{program} + T_{output} \tag{2}$$

Cycle time is an important piece of data that the programmers try to reduce to the lowest possible value. This is the time in which PLC will process all its inputs, program blocks and outputs. If this time is increased to too high of a value, PLC would not respond soon enough. It could even be possible that it does not respond to the input state change [33], when this change takes place during program block processing in one PLC cycle (Figure 22).

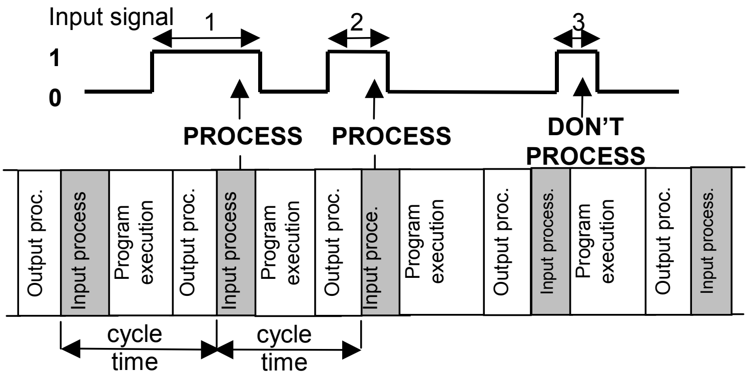


Fig. 22 *Processing the PLC input signals*

The assumption is that all blocks and service communication blocks will increase the processing PLC cycle time.

d) **Program influence on the size of PLC consumed memory** – PLC memory [39] consists of three basic parts :

1. Program memory – is used to save programs without any associated names or comments.
2. Work memory – contains all data needed for program running.
3. System memory – contains memory elements, which provide processor for user program.

PLC Memory

Program memory	**Work memory**	**System memory**
Programs without associated symbolic names and comments	Data necessary for program running	- binary memory - counters - timers - program buffer - interrupts buffer - local data buffer

Fig. 23 *PLC memory organization*

Total memory consumption PLC can be expressed as:

$$P_{total} = P_{prog} + P_{work} + P_{syst} \tag{3}$$

Because the memory PLC is limited to a few hundred kilobytes [34] and also any unnecessary program code prolongs the PLC processing cycle time, it is important to reduce the program size to a minimum possible value. This data also represents the amount of the program code that was necessary by the programmer to generate and implement within the system.

4.3 Creating the distributed system

The first step is a physical creation of the control system consisting of individual stations and creation of the communication network. First, each station must be compiled from required modules, then correctly interconnected and physically configured. If necessary, other devices can be used during creation of the communication network, such as control stations, which ensure network functionality. Then follows the software project setup, in particular the programming tool.

4.3.1 Creating the distributed IEC 61131 system

When creating the software project, the following steps are necessary:

- insert all stations into the programming tool,
- set hardware configuration of all stations and all their modules,
- configuration of the communication network and station communication interfaces,
- creating the communication program blocks [13] and review of all events that can occur when communicating,
- creation of station control system itself .

4.3.2 Creating the distributed IEC 61499 system

Steps for software project creating:

- Putting a station in the programming tool,
- Setting of the hardware configuration of all stations and their modules,
- Creating the function blocks of IEC 61499,
- Compiling the control program ,
- Interconnection configuration of individual function blocks ,
- Communication network configuration.

4.4 Testing of the selected criteria

4.4.1 Procedure for response time measuring

Measurement is based on transfer of 1000 values between master station and each of the slave stations. Transmission time of one value is calculated as an arithmetic average of the total transmission time. To ensure the most accurate results, each measurement is repeated 10 times. During this measurement, blocks must be implemented within PLC, which ensure data transmission between master and slave stations and blocks. These will record the number of transmitted values and transmission start and end.

4.4.2 Procedure for measuring network utilization

Measuring consists of measuring the device connection within the distributed system. This device will detect the amount of transmitted data between master system and slave. PLC stations will have implemented program blocks to ensure continuous data exchange between stations at the maximum possible speed. Each measurement must last a certain time in order to evaluate the results as an average value transferred over a period of time. This excludes short-term deviations from the standard status.

4.4.3 Procedure for measuring the program impact on the PLC cycle time

Measuring proceeds as follows:

- Each PLC will be recorded with a program with the number of active connections – stations will communicate continuously.
- Only necessary communication program blocks will be located in PLC, so the processor is not loaded with any other processes.
- Stations will be restarted.
- Master system cycle time will then be read from the programming tool through connection to the master station (because it is mostly loaded by the communication).

4.4.4 Procedure for measuring the program impact on the size of PLC consumed memory

Measuring will run as follows:

- Program with number of active connections will be recorded into each PLC, stations will communicate continuously.
- Only necessary communication program blocks will be located in PLC, so the processor is not loaded with any other process.
- Stations will be restarted.
- Memory consumption of PLC will be then read from the programming tool through connection to the master station (because it is most loaded by the communication).

4.5 Comparing the tested criteria

Based on the results of performed tests, it is necessary to compare the impact of important system parameters on the monitored criteria change. Among the important parameters that influence the selection of more suitable programming approaches are mainly the number of stations and number of communication channels between the stations. Therefore, it is important to carry out the tests for various numbers of control stations and corresponding communication channels. The suitability of each standard will largely depend on the nature of the designed system. There are systems for which the most important parameter is speed of communication between individual stations and processing speed of the control program. In some systems these parameters are not so important, where more importantly is the time, or the financial costs necessary to design the control system. Therefore, it is necessary to consider all these aspects and the selection of a better approach subject to these aspects.

4.6 Generalizing the methodology for suitable approach selection

The results of the comparison will serve as a basis for methodology, establishing use for suitable approach selection. This methodology is designed to recommend the programming approach, thanks to which the final system meets the requirements for distributed control system parameters.

5. METHODOLOGY VERIFICATION

5.1 Platform selection

One of the first companies that enables the creation of systems in several ways is Siemens. Programmable logic controllers Simatic S7 were originally able to only program according to IEC 61131 standard (Simatic Step7), but Siemens extended the programming tools offered by supplying iMap, which implements PROFINET CBA based on the principles of IEC 61499. Accordingly, it is possible to use PLC Simatic S7 to design distributed systems based on the IEC 61131 platform, or on the PROFINET CBA platform. Although PROFINET CBA does not implement all features of IEC 61499, basic philosophy of the communication and perception of individual devices as function blocks, is maintained.

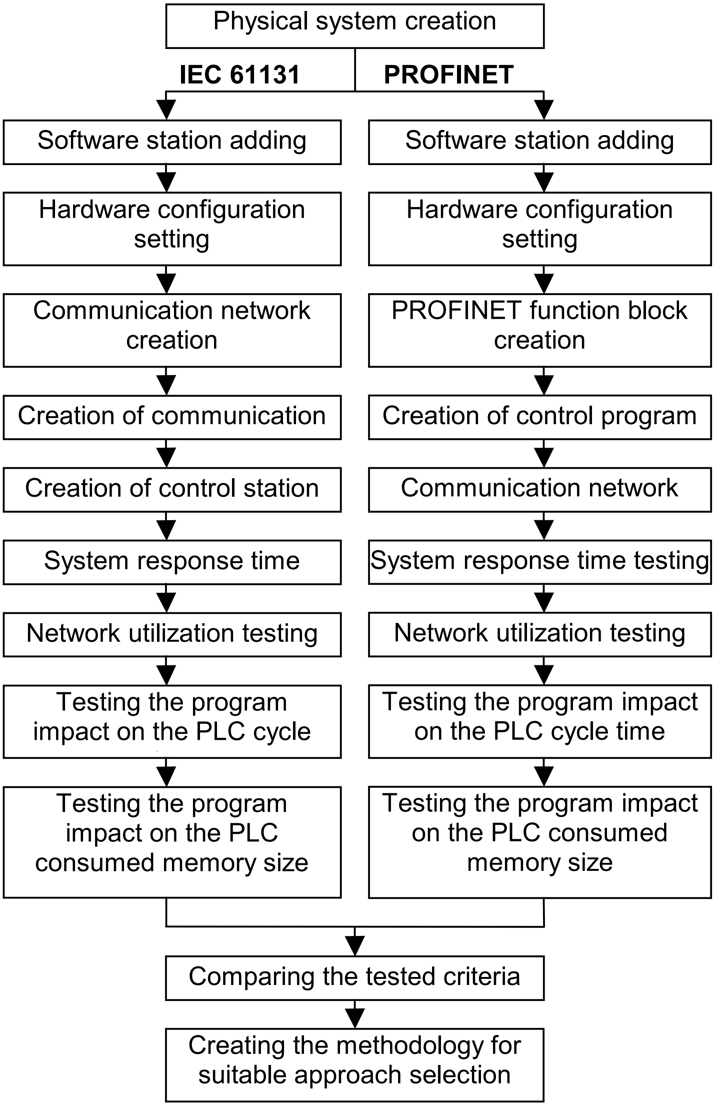

Physical system creation
IEC 61131
PROFINET
Software station adding
Software station adding
Hardware configuration setting
Hardware configuration setting
Communication network creation
PROFINET function block creation
Creation of communication
Creation of control program
Creation of control station
Communication network
System response time
System response time testing
Network utilization testing
Network utilization testing
Testing the program impact on the PLC cycle
Testing the program impact on the PLC cycle time
Testing the program impact on the PLC consumed memory size
Testing the program impact on the PLC consumed memory size
Comparing the tested criteria
Creating the methodology for suitable approach selection

Fig. 24 *Procedure for methodology establishment based on Siemens Simatic platform*

5.2 Creating the experimental control system

Model system consists of four identical PLC Siemens Simatic S7-300 that are connected through PROFINET network.

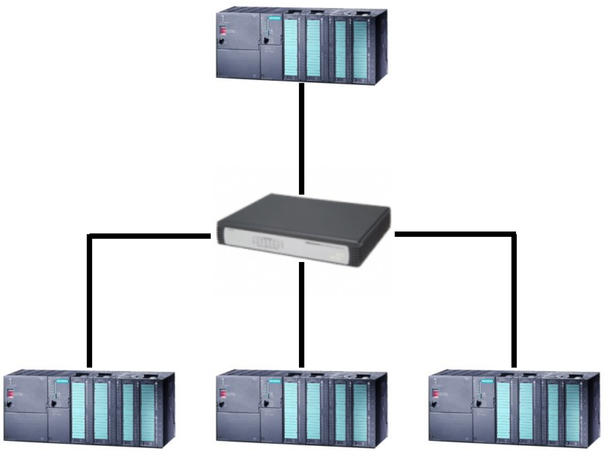


Fig. 25 *Model system*

PLC Configuration:

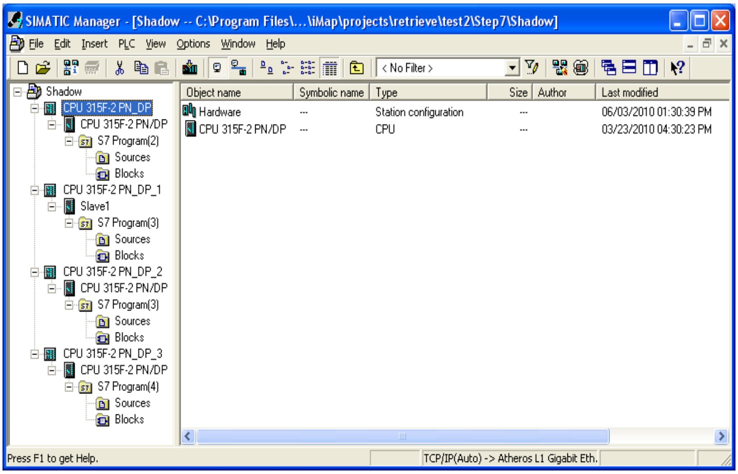
- power supply PS 307 5A,
- CPU 315F-2 PN/DP,
- memory card 512 kB,
- digital input module DI16xDC24V,
- digital input module DI32xDC24V,
- digital input / output module DI16/DO16x24V/0,5A,
- digital output module DO16xDC24V/0,5A,
- analog input / output module AI4/AO2x8/8Bit.

Each station contains interface PROFINET and ProfiBUS. One station acts within the system as a "Master" (transmits requirements and processes responses), while three remaining stations act as a "Slave" (responds to requests from the Master station).

The initial project design phase of the distributed system is identical for both approaches. First, you need to create a project by using Step7, to where all the stations and their hardware sets are configurated. This is done using the component of Step7 [48], Simatic Manager. The common procedure ends are discussed in the coming sections of this paper.

5.3 Creating IEC 61131 system

The entire system is created within a single software package Step7. First, it was necessary to start a new project in Simatic Manager tool. All four stations were sequentially added to the project.



SIMATIC Manager - [Shadow -- C:\Program Files\...\iMap\projects\retrieve\test2\Step7\Shadow]

```
Shadow
  CPU 315F-2 PN_DP
    CPU 315F-2 PN/DP
      S7 Program(2)
        Sources
        Blocks
  CPU 315F-2 PN_DP_1
    Slave1
      S7 Program(3)
        Sources
        Blocks
  CPU 315F-2 PN_DP_2
    CPU 315F-2 PN/DP
      S7 Program(3)
        Sources
        Blocks
  CPU 315F-2 PN_DP_3
    CPU 315F-2 PN/DP
      S7 Program(4)
        Sources
        Blocks
```

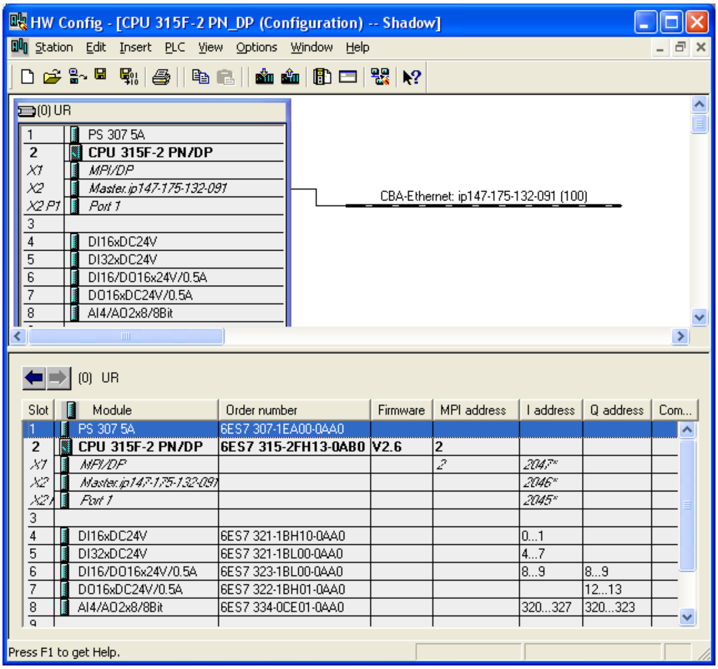
Object name	Symbolic name	Type	Size	Author	Last modified
Hardware	---	Station configuration	---		06/03/2010 01:30:39 PM
CPU 315F-2 PN/DP	---	CPU	---		03/23/2010 04:30:23 PM

Press F1 to get Help. TCP/IP(Auto) -> Atheros L1 Gigabit Eth.

Fig. 26 Project with added stations

5.3.1 Creating the hardware configuration

Each station is a modular system, which base consists of a metal frame [46], known in English as a "rail". All remaining modules are fixed on this frame. During design of the hardware configuration, the frame was also created, and gradually all station modules were added. Subsequently, each station was assigned an IP address and configuration was saved and recorded into particular PLC. Hardware configuration is created in HW Config tool.


HW Config - [CPU 315F-2 PN_DP (Configuration) -- Shadow]
Station Edit Insert PLC View Options Window Help

(0) UR

1	PS 307 5A
2	**CPU 315F-2 PN/DP**
X1	*MPI/DP*
X2	*Master.ip147-175-132-091*
X2 P1	*Port 1*
3	
4	DI16xDC24V
5	DI32xDC24V
6	DI16/DO16x24V/0.5A
7	DO16xDC24V/0.5A
8	AI4/AO2x8/8Bit

CBA-Ethernet: ip147-175-132-091 (100)

(0) UR

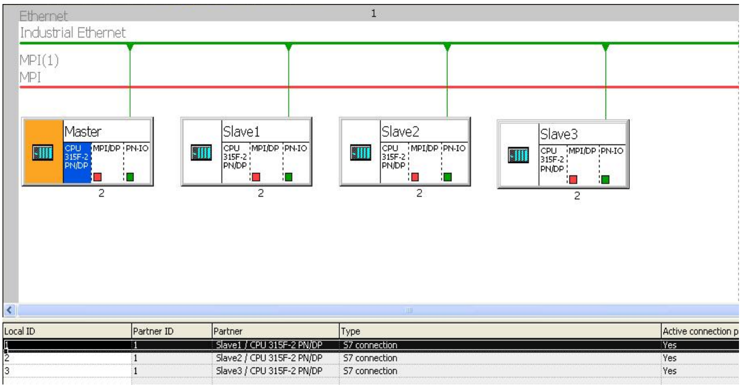
Slot	Module	Order number	Firmware	MPI address	I address	Q address	Com...
1	PS 307 5A	6ES7 307-1EA00-0AA0					
2	**CPU 315F-2 PN/DP**	**6ES7 315-2FH13-0AB0**	**V2.6**	**2**			
X1	*MPI/DP*			*2*	*2047**		
X2	*Master.ip147-175-132-091*				*2046**		
X2 P1	*Port 1*				*2045**		
3							
4	DI16xDC24V	6ES7 321-1BH10-0AA0			0...1		
5	DI32xDC24V	6ES7 321-1BL00-0AA0			4...7		
6	DI16/DO16x24V/0.5A	6ES7 323-1BL00-0AA0			8...9	8...9	
7	DO16xDC24V/0.5A	6ES7 322-1BH01-0AA0				12...13	
8	AI4/AO2x8/8Bit	6ES7 334-0CE01-0AA0			320...327	320...323	
9							

Press F1 to get Help.

***Fig.** 27 Master system hardware configuration*

5.3.2 Creating the network interface system

NetPro tool is used to create the network interface system according to IEC 61131. In this case it was necessary to create individual connections between stations by using this tool, it means one in each connection. Six connections were created in total (3 in Master–Slave direction and three in Slave–Master direction). The maximum you can create are 16 communication channels, so in the case of two-way communication, the number of stations limited is to 8. Of course, final configuration was recorded in all system stations.

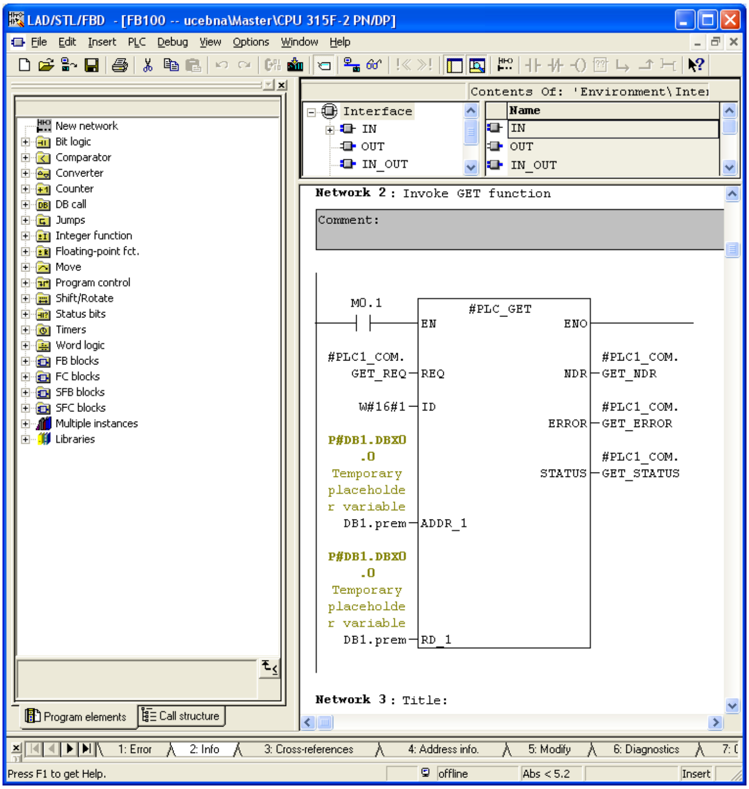


Local ID	Partner ID	Partner	Type	Active connection p
1	1	Slave1 / CPU 315F-2 PN/DP	S7 connection	Yes
2	1	Slave2 / CPU 315F-2 PN/DP	S7 connection	Yes
3	1	Slave3 / CPU 315F-2 PN/DP	S7 connection	Yes

Fig. 28 *Network designed by NetPro tool*

5.3.3 Creating the communication blocks

Communication in Step7 is not cyclic, and is realised by using system functions PUT and GET. Function PUT [44] allows data to be sent to a remote system. It is called with the parameters such as communication channel number, data address in local system and data address in remote system. After calling the function, the system waits for a response, it can be sending a confirmation or an error message. GET function is used to receive data from a remote system and similar parameters are being called, like for PUT function. Each of these system functions occupies one communication channel.



```
Network 2 : Invoke GET function
Comment:

M0.1                  #PLC_GET
         EN                         ENO
#PLC1_COM.
  GET_REQ -REQ                      NDR- #PLC1_COM.
                                         GET_NDR
  W#16#1 -ID                      ERROR- #PLC1_COM.
                                         GET_ERROR
P#DB1.DBX0
  .0                             STATUS- #PLC1_COM.
Temporary                                GET_STATUS
placeholde
r variable
  DB1.prem -ADDR_1

P#DB1.DBX0
  .0
Temporary
placeholde
r variable
  DB1.prem -RD_1

Network 3 : Title:
```

Fig. 29 Calling the GET function

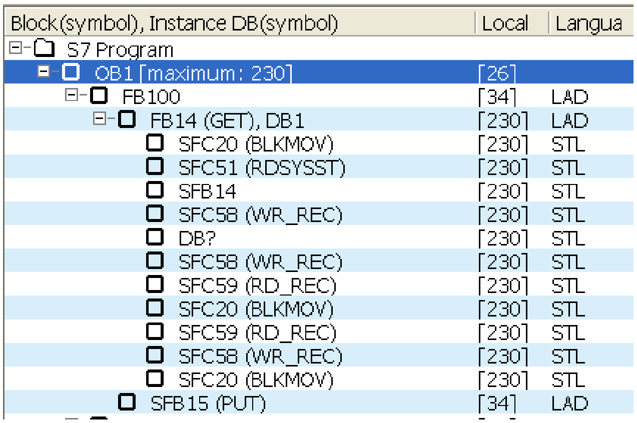
Measuring program in master station consists of:

- ***Program blocks***
  - OB1 – within this block all function blocks ensuring the communication are cyclically called; it also monitors the achieved required number of transmitted data,
  - OB100 – initialize the communication by system startup.
- ***Function blocks***
  - FB14 – GET communication function,
  - FB100 – function block provides data sending and receiving from the first slave, station, it also provides counting of transmitted data and saving the information about transmission start and finish,
  - FB101 – identical to FB100, supports the second slave station,
  - FB102 – identical to FB100, supports the third slave station.
- ***Functions***
  - FC1 – function detecting actual system time,
  - FC8 – system time conversion.
- ***System function blocks***
  - SFB14 – GET communication function,
  - SFB15 – PUT communication function.
- ***System functions***
  - SFC20 – BLKMOV, called by function GET and PUT,
  - SFC51 – RDSYSST, called by function GET and PUT,
  - SFC58 – WR_REC, called by function GET and PUT,
  - SFC59 – RD_REC, called by function GET and PUT.

- ***Data blocks***
    - DB1 – store transmitted data values,
    - DB2 – store time values for transmission start and finish,
    - DB100 – instance data block automatically created for communication purposes with first slave station,
    - DB101 – instance data block automatically created for communication purposes, with second slave station,
    - DB102 – instance data block automatically created for communication purposes with third slave station.

Slave stations contain only two blocks:

- program block OB1 – cyclically called, must exist in PLC,
- data block DB1 – save values of the transmitted data.

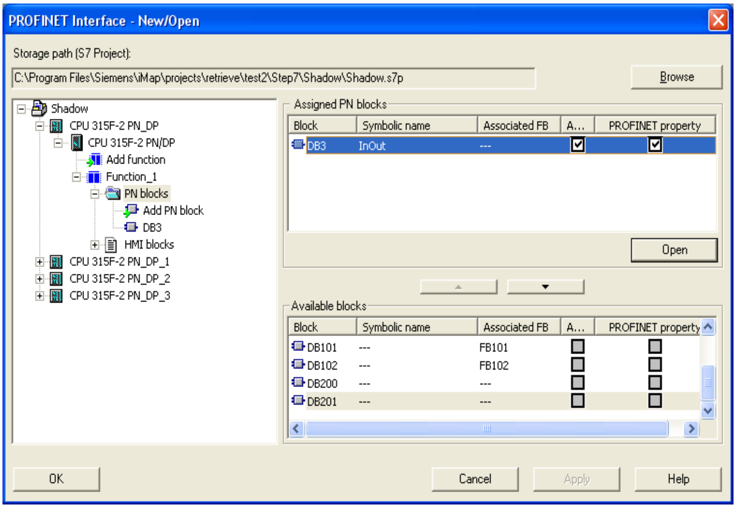


Block(symbol), Instance DB(symbol)	Local	Langua
S7 Program		
OB1 [maximum: 230]	[26]	
FB100	[34]	LAD
FB14 (GET), DB1	[230]	LAD
SFC20 (BLKMOV)	[230]	STL
SFC51 (RDSYSST)	[230]	STL
SFB14	[230]	STL
SFC58 (WR_REC)	[230]	STL
DB?	[230]	STL
SFC58 (WR_REC)	[230]	STL
SFC59 (RD_REC)	[230]	STL
SFC20 (BLKMOV)	[230]	STL
SFC59 (RD_REC)	[230]	STL
SFC58 (WR_REC)	[230]	STL
SFC20 (BLKMOV)	[230]	STL
SFB15 (PUT)	[34]	LAD

***Fig. 30** Blocks called by GET system function*

5.4 Creating PROFINET CBA system

The first steps when creating PROFINET CBA system are identical to creating the IEC 61131 system. The project was designed by using Step7 [42], four stations were added and their hardware configuration set. Unlike the connection NetPro, the PROFINET CBA works differently. First step is to create function blocks of PROFINET CBA [41] and particular shared data blocks. One PROFINET CBA function block was created within each station, which was associated with a shared DB3 data block. Master station is in DB3 three input and three output variables (Slave1In, Slave2In, Slave3In, Slave1Out, Slave2Out, Slave3Out). Slave stations store only one output variable in DB3.


PROFINET Interface - New/Open

Storage path (S7 Project): C:\Program Files\Siemens\iMap\projects\retrieve\test2\Step7\Shadow\Shadow.s7p

Assigned PN blocks

Block	Symbolic name	Associated FB	A...	PROFINET property
DB3	InOut	---	☑	☑

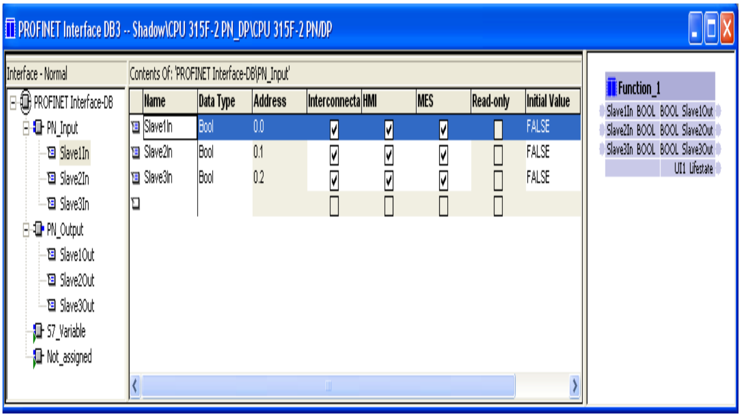
Available blocks

Block	Symbolic name	Associated FB	A...	PROFINET property
DB101	---	FB101	☐	☐
DB102	---	FB102	☐	☐
DB200	---	---	☐	☐
DB201	---	---	☐	☐

Fig. 31 Creating PROFINET function and communication interface

Master station contains following blocks:

- ***program blocks***
  - OB1 – communication variable status is cyclically monitored within this block, change of their values will increase the value of transferred data counter. Measuring startup and end time is also recorded here;
- ***functions***
  - FC1 – function detecting correct system time,
  - FC8 – system time conversion;
- ***data blocks***
  - DB3 – save values of the transmitted data,
  - DB2 – save time values of the transmission start and finish.



PROFINET Interface DB3 -- Shadow\CPU 315F-2 PN_DP\CPU 315F-2 PN/DP

Interface - Normal

- PROFINET Interface-DB
  - PN_Input
    - Slave1In
    - Slave2In
    - Slave3In
  - PN_Output
    - Slave1Out
    - Slave2Out
    - Slave3Out
  - S7_Variable
  - Not_assigned

Contents Of: 'PROFINET Interface-DB\PN_Input'

Name	Data Type	Address	Interconnecta	HMI	MES	Read-only	Initial Value
Slave1In	Bool	0.0	☑	☑	☑	☐	FALSE
Slave2In	Bool	0.1	☑	☑	☑	☐	FALSE
Slave3In	Bool	0.2	☑	☑	☑	☐	FALSE
			☐	☐	☐	☐	

Function_1

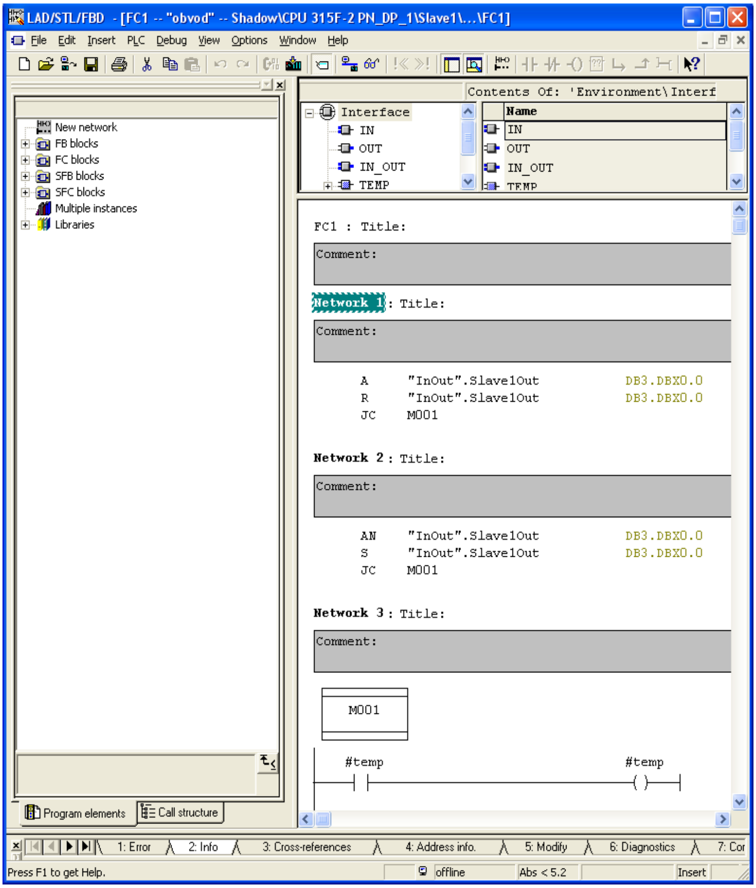
Slave1In	BOOL	BOOL	Slave1Out
Slave2In	BOOL	BOOL	Slave2Out
Slave3In	BOOL	BOOL	Slave3Out
		UI1	Lifestate

Fig. 32 *Variables assignment for the communication interface*

As seen, compared to the classic style of programming [45] without using iMap, the number of program blocks is much smaller.

Slave stations contain the following blocks:

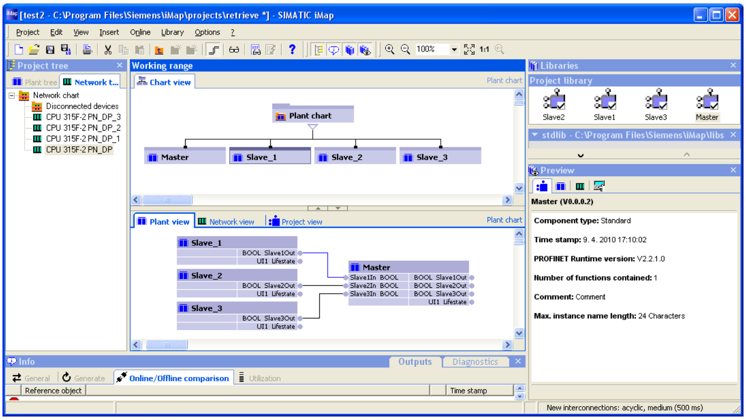
- program block OB1 – cyclically called data generator FC1,
- function FC1 – ensures cyclical values change of transmitted date,
- data block DB3 – save values of the transmitted data.


LAD/STL/FBD - [FC1 -- "obvod" -- Shadow\CPU 315F-2 PN_DP_1\Slave1\...\FC1]

```
FC1 : Title:
Comment:

Network 1: Title:
Comment:

      A      "InOut".Slave1Out              DB3.DBX0.0
      R      "InOut".Slave1Out              DB3.DBX0.0
      JC     M001

Network 2 : Title:
Comment:

      AN     "InOut".Slave1Out              DB3.DBX0.0
      S      "InOut".Slave1Out              DB3.DBX0.0
      JC     M001

Network 3 : Title:
Comment:

   M001

   #temp                                    #temp
```

Fig. 33 FC1 function

PROFINET components [47] were subsequently created from PROFINET CBA function blocks, which contain inputs and outputs (elements of shared data blocks). These were later implemented into the Simatic iMap tool. Individual components are in iMap [43], graphically connected, and that created a network interface for distributed system. Cyclic communication of each channel with minimum refresh time was adjusted, which is 8 ms. Then, communication of the whole system was recorded into each station.



Master (V0.0.0.2)

Component type: Standard

Time stamp: 9. 4. 2010 17:10:02

PROFINET Runtime version: V2.2.1.0

Number of functions contained: 1

Comment: Comment

Max. instance name length: 24 Characters

New interconnections: acyclic, medium (500 ms)

Fig. 34 *Project iMap with added components and formed links*

5.5 Comparing the response time of the remote stations

The time in which the remote station transferred the information about system status change was determined in this measurement.

5.5.1 Measuring the system response time by Step7 and NetPro

When measuring the response of one slave station, FB101 and FB102 were block calls excluded; when measuring the response of two slave stations, FB102 block call was excluded. When the measuring started, current start time was written to DB2 block, when the data transmission finished, finish time was written individually for each station. The last time is considered to be the total time of communication.

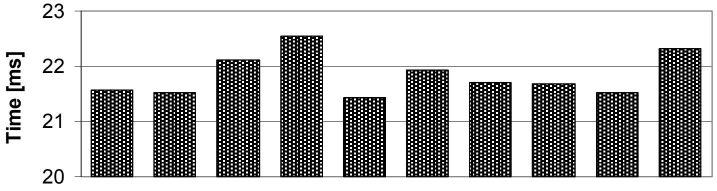


Fig. 35 *Response time for communication with a single station, one binary value*

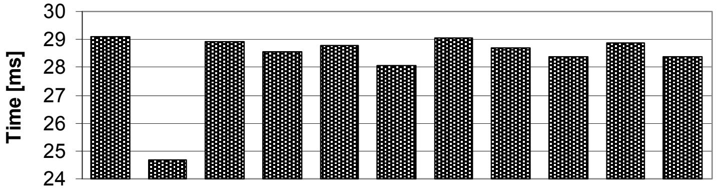


Fig. 36 *Response time for communication with two stations, one binary value*

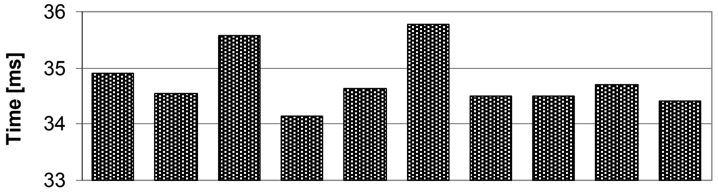


Fig. 37 *Response time for communication with three stations, one binary value*

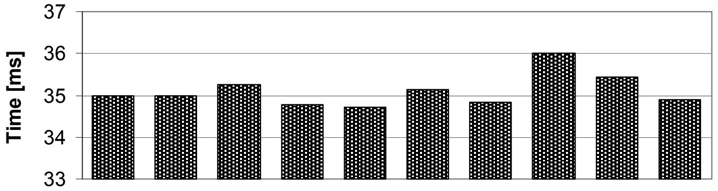


Fig. 38 *Response time for communication with three stations, 20 integer values*

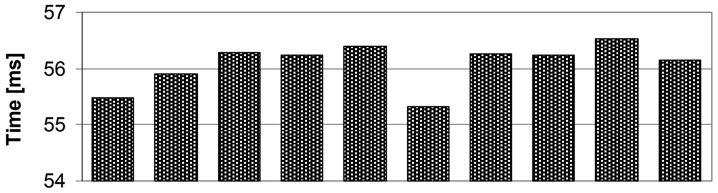
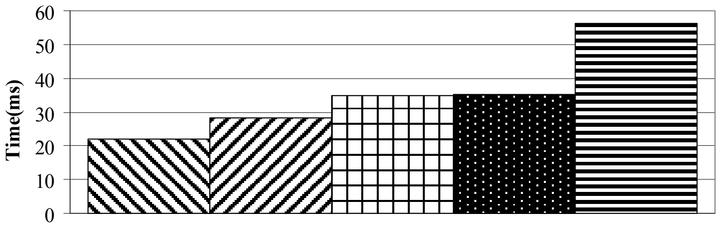


Fig. 39 *Response time, two-way communication with three stations, 20 values*

As can be seen in Figure 39, the system measurement summary of the Step7 system, during transmission of one value through one channel, the average response time was nearly 22 ms. When another communication system was added, response time raised by 7 ms. Change of the transmitted data types from binary values to integer values did not influence the response time.



▧ Response time for communication with a single station, one binary value is transmitted

▨ Response time for communication with two stations, one binary value is transmitted

⊞ Response time for communication with three stations, one binary value is transmitted

■ Response time for communication with three stations, 20 integer values are transmitted

☰ Response time for two-way communication with three stations, 20 integer values are transmitted

Fig. 40 *Overview of the average response times by using Step7 and NetPro*

5.5.2 Measuring the system response time by iMap

When begun, current start time was written into DB2 block, when finished with data transmission, finish time was written for each station individually. The last time is considered to be the total communication time. Cyclic communication with cycle time of 8 ms was set for the system.

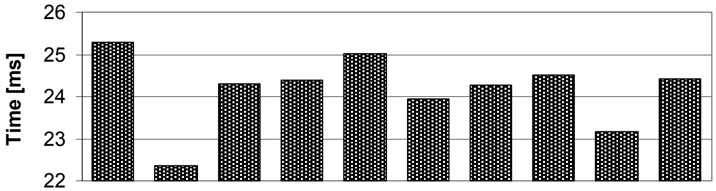


Fig. 41 *Response time for communication with a single station, one binary value*

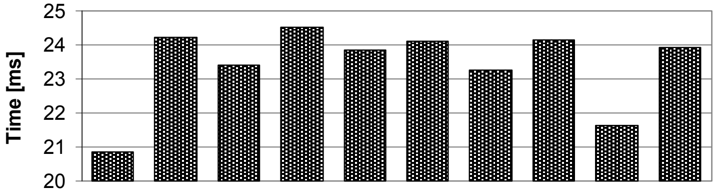


Fig. 42 *Response time for communication with two stations, one binary value*

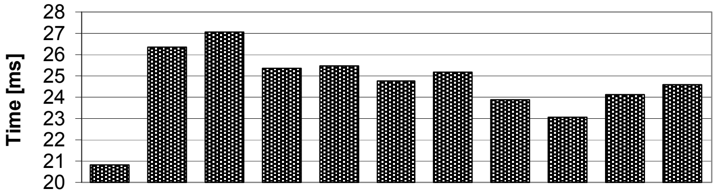


Fig. 43 *Response time for communication with three stations, one binary value*

As possible to see from measurement results for the iMap system, change in the number of communication channels has no major impact on response time, as it still fluctuates around the value of 24 ms. Set cycle value of 8 ms was not reached, PLC reached for both communication types, minimum time 22 to 24 ms. This is probably the lowest communication cycle time reached with this type of processor.

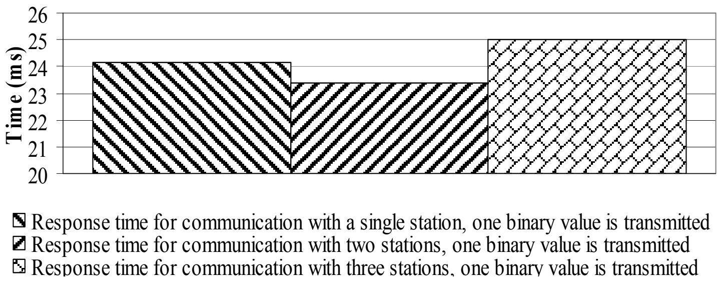


Fig. 44 *Overview of the average response times by using iMap*

As seen in the results, a distributed system created only by using Step7 is influenced by number of active connections. Any additional connection increased the response time of about 8 ms. Changing the transmitted data type does not affect response time. Number of connections is limited to 16.

A distributed system created by using iMap is not affected by the number of active connections.

Response time is kept to a limit of 24 ms and number of connections is not limited.

5.6 Comparing the network utilization

Since individual stations are connected to the network by a network switch, which operates on the principle of sending the address data to the end device, it is not possible to determine the utilization through connection of measuring device (in our case, computer with installed software for network monitoring and catching of data packets) to the network by the same way. An alternative is to replace the network switch by hub, which sends received data to all connected devices, but PROFINET CBA network cannot communicate when using a hub. One functional solution offered was to connect measuring computer between computer and switch and master station as a transparent bridge. For this role, it is necessary to equip the PC with two network interfaces.

To catch the packets, clear operation system Windows XP with all unnecessary services turned off and Wireahark software, was used. Each measuring lasted 5 minutes.

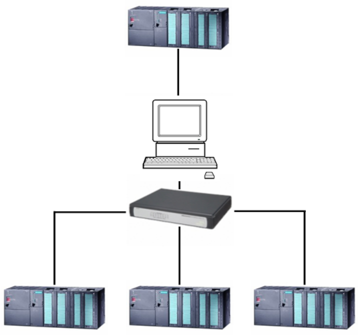


Fig. 45 *System scheme with connected transmittance meter*

First measuring was performed with distributed system, which was created by Step7.

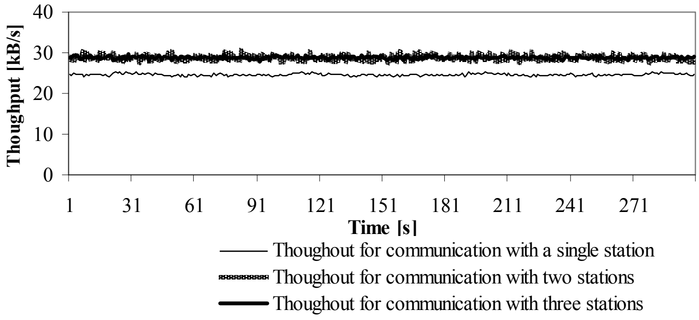


Fig. 46 Measuring the network utilization by using Step7 and NetPro

As confirmed by measurements, network utilization during master system communication with one of the slaves was about 25kB/s. When adding another communication channel, utilization raised to approximately 29kB/s and stayed stable afterwards. Based on response speed measuring, the addition of each communication channel was increasing the response time, therefore, with each channel, data was transmitted with less frequency. From this we can conclude that utilization of 29kB/s is probably the maximum reachable value. When reaching this value it is compensated by decrease in number of transmitted data by increasing the remote system response time.

Another two measurements were performed by distributed system created using Simatic iMap.

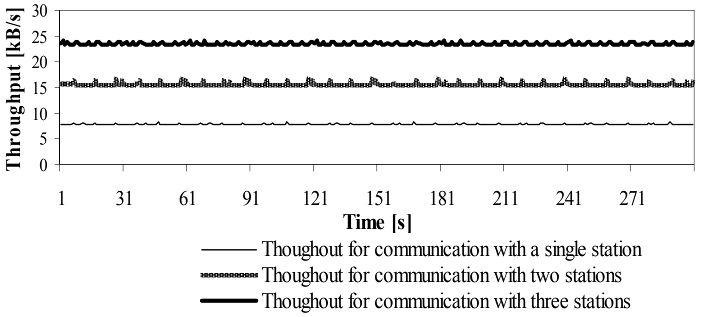


Fig. 47 *Measuring the network utilization by using iMap, 8 ms cycle time*

As seen in Figure 47, network utilization was increasing linearly when adding more communication channels, each channel caused a utilization increase about 8kB/s. When communicating with three stations, the utilization was approximately 24kB/s when using iMap, when using only Step7 it was approximately 29kB/s. However, the response time of the system created by iMap was approximately 24 ms and for the system created by Step7 it was approximately 35 ms. This shows that a system created by iMap can transmit smaller data packets. For that reason, another measurement was done by using a system designed by iMap, and the cycle time increased to 32 ms.

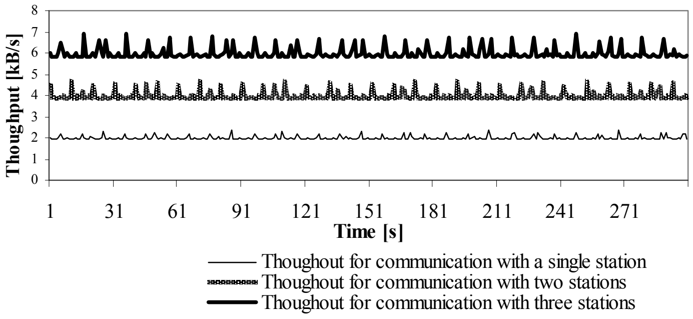
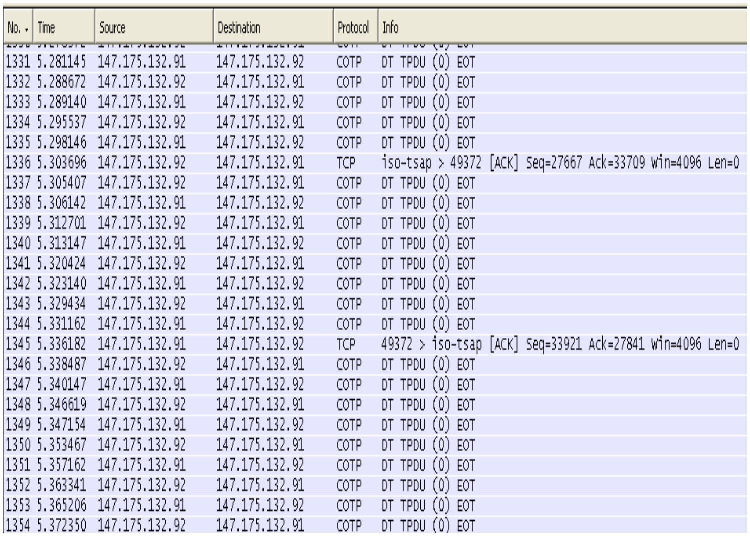


***Fig. 48** Measuring the network utilization by using iMap, 32 ms cycle time*

As seen in Figure 48, change of the cycle time radically decreased data transfers between the stations, and the actual response time was approximately 50 ms. Every communication channel to follow increased the data transfer by 2kB/s.

Because of the apparent difference in number of transmitted data, an analysis of transmitted packets was performed by using both communication methods.

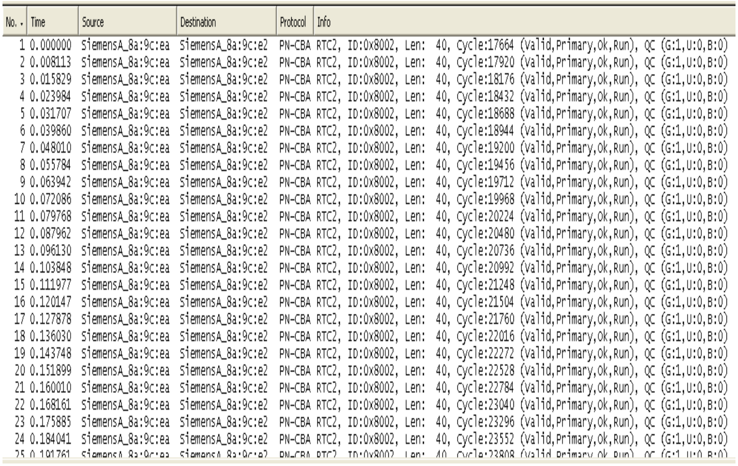
There is a record of transmitted datas between two system stations created in Step7 on Figure 49. As shown, two-way communication TCP/IP is running between the stations. Master system is cyclically sending data packets to the slave system. The slave system responds, and after successful sending of the whole data frame, it confirms by sending ACK packet to the master system. Then the master system confirms that the frame was received.



```
No. -  Time      Source          Destination     Protocol Info
1331 5.281145 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1332 5.288672 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1333 5.289140 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1334 5.295537 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1335 5.298146 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1336 5.303696 147.175.132.92  147.175.132.91  TCP   iso-tsap > 49372 [ACK] Seq=27667 Ack=33709 Win=4096 Len=0
1337 5.305407 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1338 5.306142 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1339 5.312701 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1340 5.313147 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1341 5.320424 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1342 5.323140 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1343 5.329434 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1344 5.331162 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1345 5.336182 147.175.132.91  147.175.132.92  TCP   49372 > iso-tsap [ACK] Seq=33921 Ack=27841 Win=4096 Len=0
1346 5.338487 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1347 5.340147 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1348 5.346619 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1349 5.347154 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1350 5.353467 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1351 5.357162 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1352 5.363341 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
1353 5.365206 147.175.132.91  147.175.132.92  COTP  DT TPDU (0) EOT
1354 5.372350 147.175.132.92  147.175.132.91  COTP  DT TPDU (0) EOT
```

Fig. 49 *System communication record generated in Step7*

The record of transmitted data between two system stations, created in iMap is shown in Figure 50. As shown, it is a one-way communication. When the network communication is created and recorded in all PLCs, data are automatically sent in specified intervals to the end stations. In this particular case, the slave system is sending the information to the master system every 8 ms. They do not have to communicate using TCP/IP protocol, but rather PROFINET CBA protocol.



```
No. Time     Source            Destination       Protocol Info
  1 0.000000 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:17664 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  2 0.008113 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:17920 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  3 0.015829 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:18176 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  4 0.023984 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:18432 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  5 0.031707 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:18688 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  6 0.039860 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:18944 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  7 0.048010 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:19200 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  8 0.055784 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:19456 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
  9 0.063942 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:19712 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 10 0.072086 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:19968 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 11 0.079768 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:20224 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 12 0.087962 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:20480 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 13 0.096130 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:20736 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 14 0.103848 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:20992 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 15 0.111977 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:21248 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 16 0.120147 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:21504 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 17 0.127878 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:21760 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 18 0.136030 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:22016 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 19 0.143748 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:22272 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 20 0.151899 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:22528 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 21 0.160010 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:22784 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 22 0.168161 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:23040 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 23 0.175885 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:23296 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 24 0.184041 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:23552 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
 25 0.191761 SiemensA_8a:9c:ea SiemensA_8a:9c:e2 PN-CBA RTC2, ID:0x8002, Len:  40, Cycle:23808 (Valid,Primary,Ok,Run), QC (G:1,U:0,B:0)
```

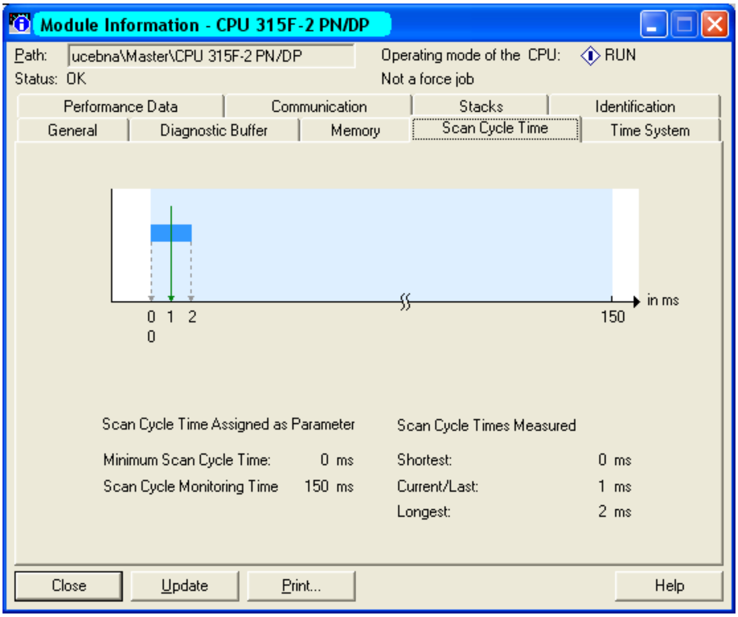
Fig. 50 *System communication record generated in iMap*

The system created in Step7 communicated in a bidirectional manner in each processor cycle. This time is not affected by number of operations, which must be executed in one cycle by PLC. This value range is standardly from 1 to 10 ms. That is why the amount of transferred data in this way is higher, even though the response times are higher than by the iMap system. In contrast, iMap system transmissions are executed only at specific times, therefore the total data flow is lower even for lower response time.

An interesting fact is that although the slave station is sending information every 8ms, they are processed by the master system approximately every 24 ms. Also, the minimum response value reached by the Step7 system was 22 ms, although the processor processed the program every 3 ms. This difference of about 16-19 ms is probably the time required for processing the processor network interface data.

5.7 Comparing the program impact on the PLC cycle time

Data were collected through the Hardware Config tool. This enables the monitoring of current PLC status, including the cycle time.



Path: ucebna\Master\CPU 315F-2 PN/DP
Operating mode of the CPU: RUN
Status: OK
Not a force job

Scan Cycle Time Assigned as Parameter

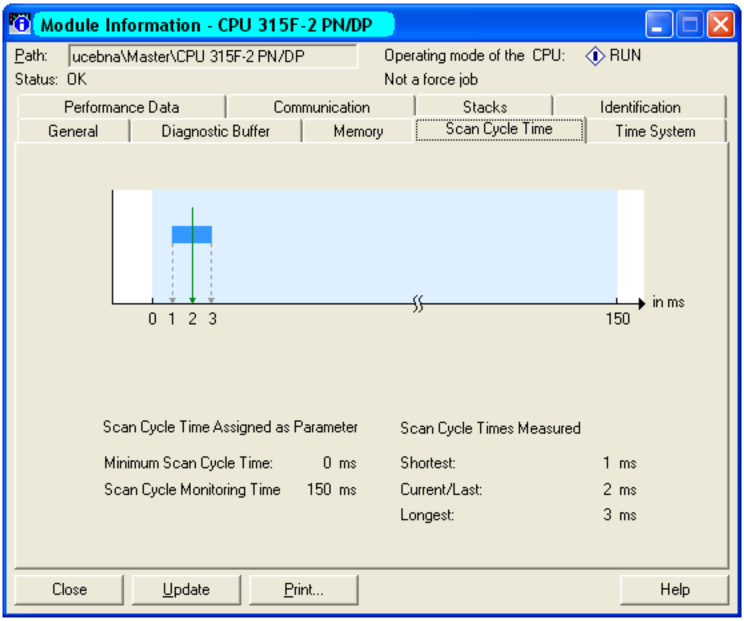
Minimum Scan Cycle Time: 0 ms
Scan Cycle Monitoring Time 150 ms

Scan Cycle Times Measured

Shortest: 0 ms
Current/Last: 1 ms
Longest: 2 ms

Fig. 51 *Cycle time for Step7 system, communication with a single station*

During communication of Step7 system with one station, the longest cycle lasted 2ms, average was 1 ms and lowest value was 0 ms.


Module Information - CPU 315F-2 PN/DP

Path: ucebna\Master\CPU 315F-2 PN/DP
Operating mode of the CPU: RUN
Status: OK
Not a force job

Scan Cycle Time Assigned as Parameter

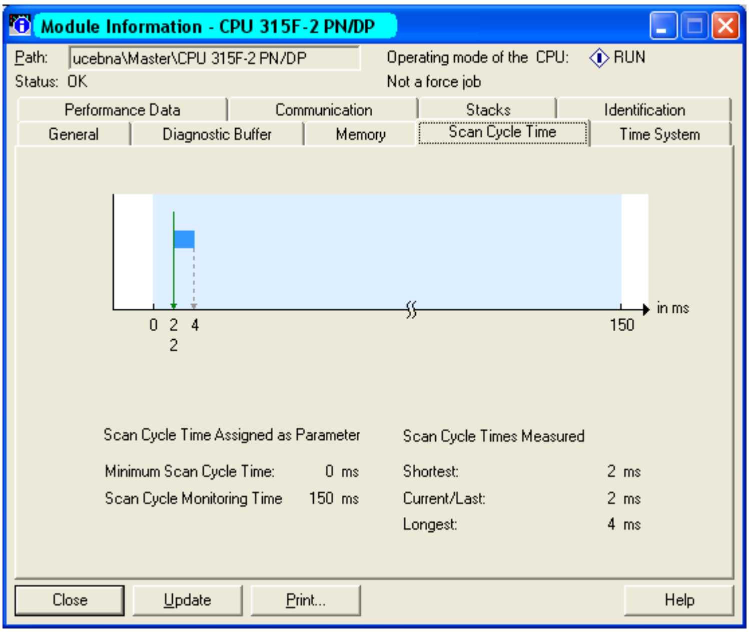
Minimum Scan Cycle Time:	0 ms
Scan Cycle Monitoring Time	150 ms

Scan Cycle Times Measured

Shortest:	1 ms
Current/Last:	2 ms
Longest:	3 ms

***Fig. 52** Cycle time for Step7 system, communication with two stations*

During communication of Step7 system with two stations, the longest cycle lasted 3ms, average was 2 ms and lowest value was 1 ms.



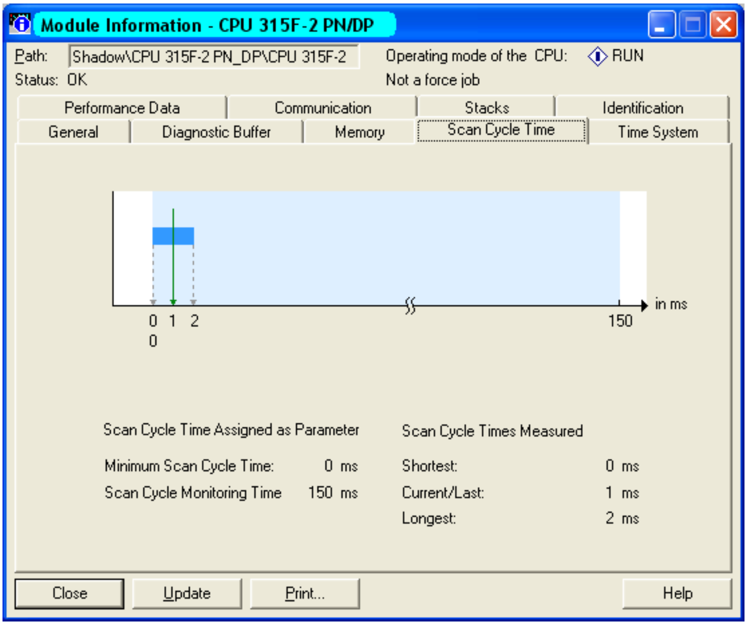
Path: ucebna\Master\CPU 315F-2 PN/DP   Operating mode of the CPU: RUN
Status: OK   Not a force job

Scan Cycle Time Assigned as Parameter		Scan Cycle Times Measured	
Minimum Scan Cycle Time:	0 ms	Shortest:	2 ms
Scan Cycle Monitoring Time	150 ms	Current/Last:	2 ms
		Longest:	4 ms

Fig. 53 cycle time for Step7 system, communication with three stations

During communication of Step7 system with three stations the longest cycle lasted 4ms, average was 3 ms and lowest value was 2 ms.

The results show that each active PUT or GET communication block increases the cycle processing time by approximately 1ms; this means for maximum number of connections it can be up to 16 ms more.



Path: Shadow\CPU 315F-2 PN_DP\CPU 315F-2
Operating mode of the CPU: RUN
Status: OK
Not a force job

Scan Cycle Time Assigned as Parameter

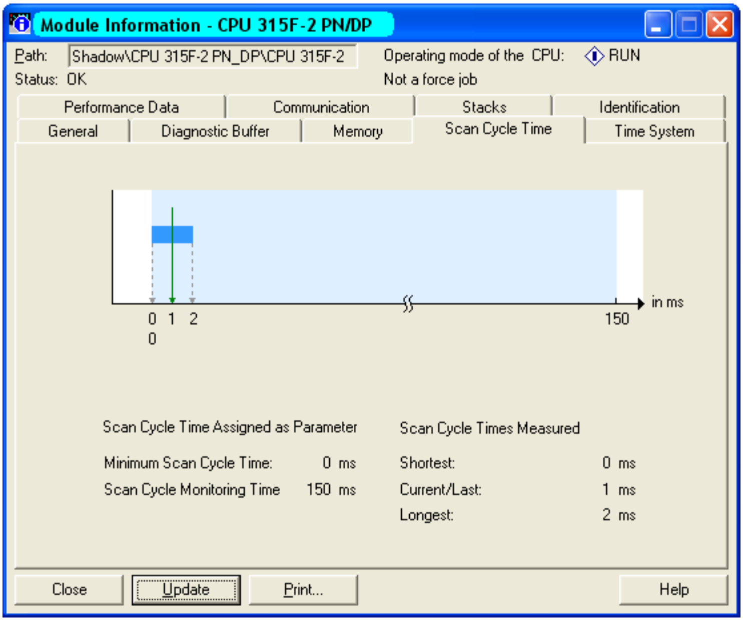
Minimum Scan Cycle Time: 0 ms
Scan Cycle Monitoring Time 150 ms

Scan Cycle Times Measured

Shortest: 0 ms
Current/Last: 1 ms
Longest: 2 ms

Fig. 54 *Cycle time for iMap system, communication with a single station*

During communication of iMap system with one station the longest cycle lasted 2ms, average was 1 ms and lowest value was 0 ms.


Module Information - CPU 315F-2 PN/DP

Path: Shadow\CPU 315F-2 PN_DP\CPU 315F-2   Operating mode of the CPU: RUN

Status: OK   Not a force job

Scan Cycle Time Assigned as Parameter

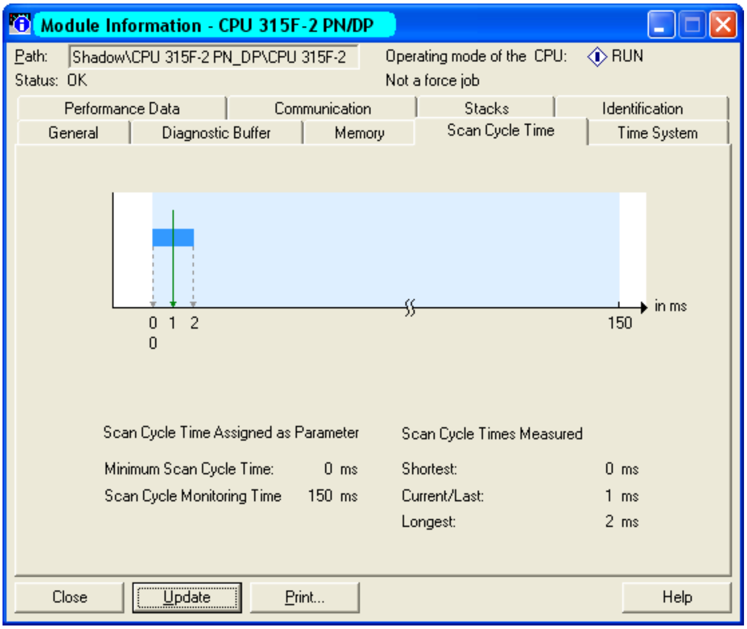
Minimum Scan Cycle Time:	0 ms
Scan Cycle Monitoring Time	150 ms

Scan Cycle Times Measured

Shortest:	0 ms
Current/Last:	1 ms
Longest:	2 ms

***Fig. 55** Cycle time for iMap system, communication with two stations*

During communication of iMap system with two stations the longest cycle lasted 2ms, average was 1 ms and lowest value was 0 ms.


Module Information - CPU 315F-2 PN/DP

Path: Shadow\CPU 315F-2 PN_DP\CPU 315F-2 Operating mode of the CPU: RUN

Status: OK Not a force job

Scan Cycle Time Assigned as Parameter

Minimum Scan Cycle Time: 0 ms

Scan Cycle Monitoring Time 150 ms

Scan Cycle Times Measured

Shortest: 0 ms

Current/Last: 1 ms

Longest: 2 ms

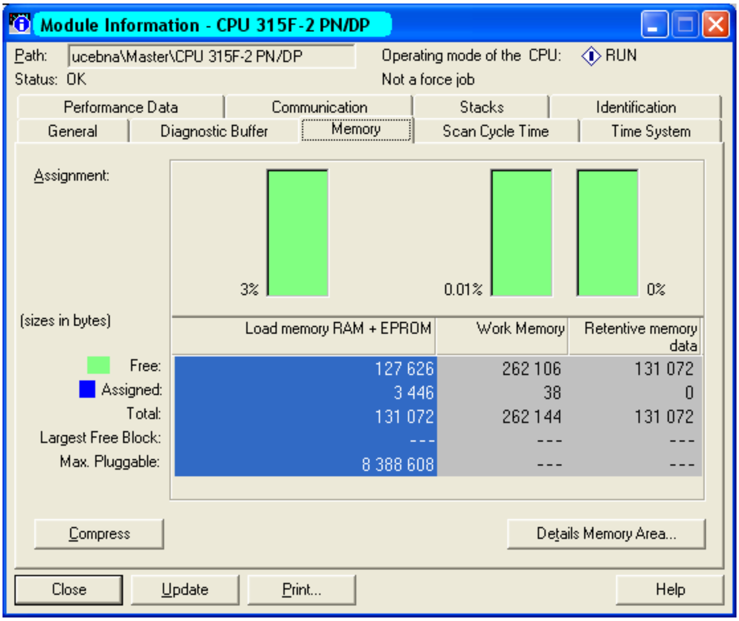
***Fig. 56** Cycle time for iMap system, communication with three stations*

During communication of iMap system with three stations the longest cycle lasted 2ms, average was 1 ms and lowest value was 0 ms.

The analysis results show that communication interface and channels created by using iMap do not affect program processing speed in PLC. This is because PLC is not burdened by communication function blocks, as was the case for a system created by Step7.

5.8 Comparing the program impact on the PLC memory size

Data was collected using the Hardware Config tool. It allows for the monitoring of the current state of PLC, including the size of memory consumption.


Module Information - CPU 315F-2 PN/DP

Path: ucebna\Master\CPU 315F-2 PN/DP   Operating mode of the CPU: RUN
Status: OK   Not a force job

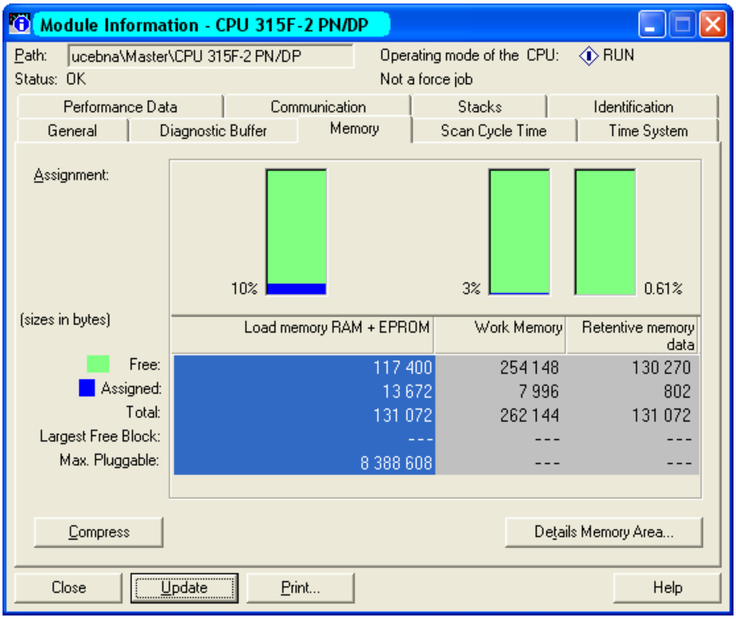
Assignment: 3% 0.01% 0%

(sizes in bytes)

	Load memory RAM + EPROM	Work Memory	Retentive memory data
🟩 Free:	127 626	262 106	131 072
🟦 Assigned:	3 446	38	0
Total:	131 072	262 144	131 072
Largest Free Block:	---	---	---
Max. Pluggable:	8 388 608	---	---

Fig. 57 PLC Memory consumption without the program, only with recorded HW configuration

As shown in Figure 57, even PLC in basic status contains information saved in memory. It is mainly about the device hardware configuration (Load memory RAM + EPROM). Also, an empty block QB1, which is necessary to activate PLC, takes few bytes (38 bytes of work memory).



Module Information - CPU 315F-2 PN/DP

Path: ucebna\Master\CPU 315F-2 PN/DP   Operating mode of the CPU: RUN

Status: OK   Not a force job

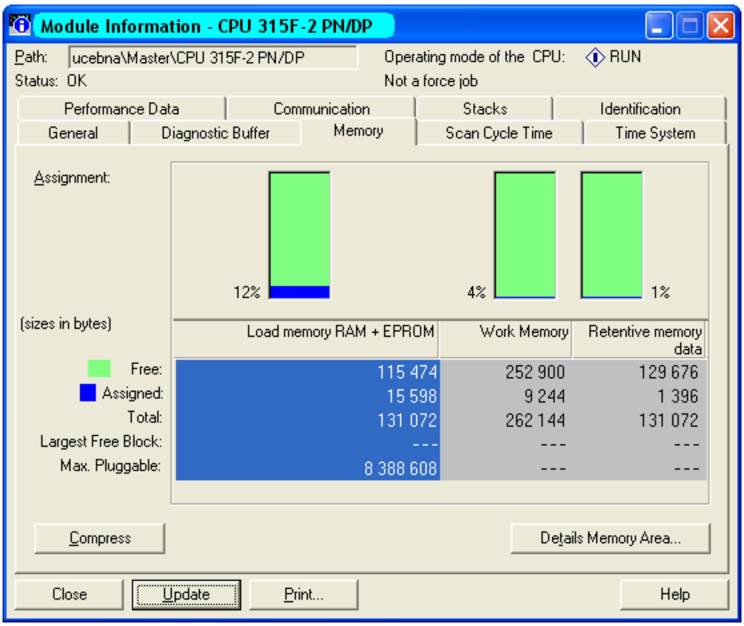
Assignment:

(sizes in bytes)

	Load memory RAM + EPROM	Work Memory	Retentive memory data
	10%	3%	0.61%
🟩 Free:	117 400	254 148	130 270
🟦 Assigned:	13 672	7 996	802
Total:	131 072	262 144	131 072
Largest Free Block:	---	---	---
Max. Pluggable:	8 388 608	---	---

Fig. 58 *PLC memory consumption programmed by Step7, one connection*

Creating one communication channel using NetPro and all necessary program blocks in Step7 increased the memory consumption EPROM to 10% (13672 B), work memory to 3% (7996 b) and retentive memory to 0,61% (802 b).



Path: ucebna\Master\CPU 315F-2 PN/DP

Operating mode of the CPU: RUN

Status: OK

Not a force job

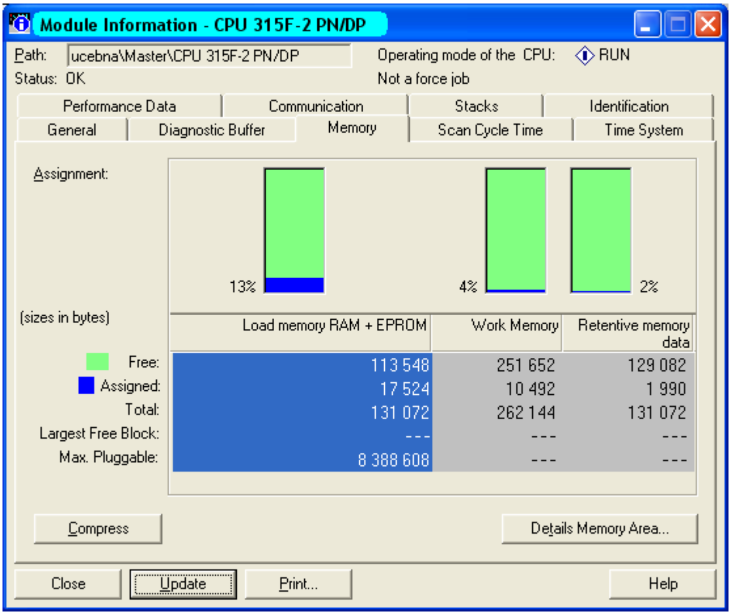
Assignment:

(sizes in bytes)

	Load memory RAM + EPROM	Work Memory	Retentive memory data
🟩 Free:	115 474	252 900	129 676
🟦 Assigned:	15 598	9 244	1 396
Total:	131 072	262 144	131 072
Largest Free Block:	---	---	---
Max. Pluggable:	8 388 608	---	---

Fig. 59 *PLC memory consumption programmed by Step7, two connections*

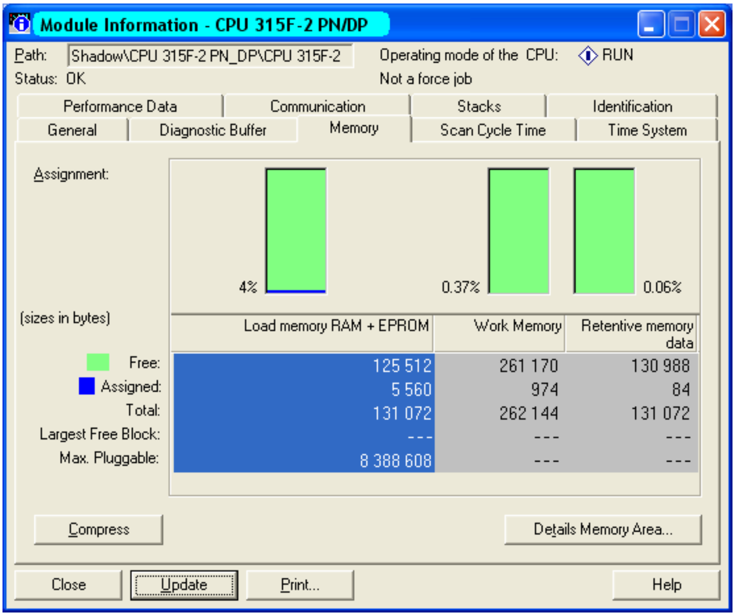
Creating two communication channels using NetPro and all necessary program blocks in Step7 increased the memory consumption EPROM to 12% (15598 b), work memory to 4% (9244 b) and retentive memory to 1% (1396 b).



(sizes in bytes)	Load memory RAM + EPROM	Work Memory	Retentive memory data
🟩 Free:	113 548	251 652	129 082
🟦 Assigned:	17 524	10 492	1 990
Total:	131 072	262 144	131 072
Largest Free Block:	---	---	---
Max. Pluggable:	8 388 608	---	---

Fig. 60 *PLC memory consumption programmed by Step7, three connections*

Creating three communication channels using NetPro and all necessary program blocks in Step7 increased the memory consumption EPROM to 13% (17524 b), work memory to 4% (10492 b) and retentive memory to 2% (1990 b).


Module Information - CPU 315F-2 PN/DP

Path: Shadow\CPU 315F-2 PN_DP\CPU 315F-2
Operating mode of the CPU: RUN
Status: OK
Not a force job

Assignment:

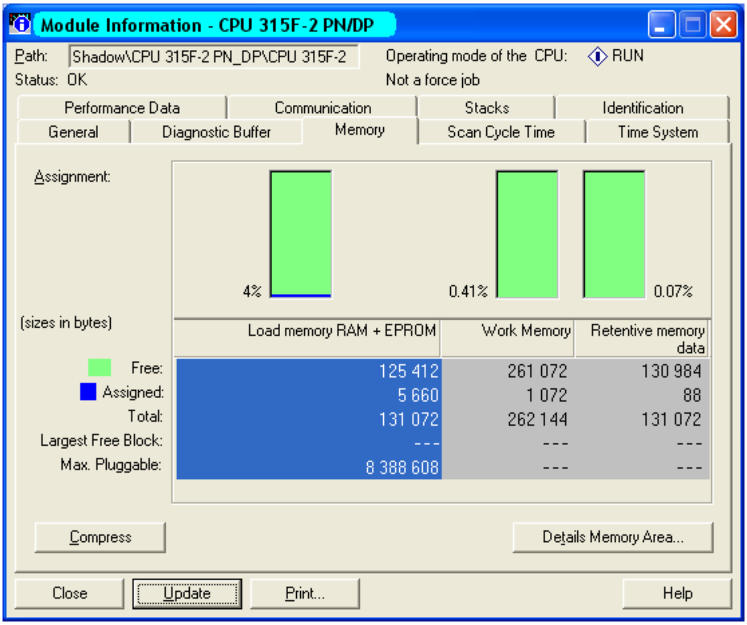
4% 0.37% 0.06%

(sizes in bytes)

	Load memory RAM + EPROM	Work Memory	Retentive memory data
🟩 Free:	125 512	261 170	130 988
🟦 Assigned:	5 560	974	84
Total:	131 072	262 144	131 072
Largest Free Block:	---	---	---
Max. Pluggable:	8 388 608	---	---

***Fig. 61** PLC memory consumption, iMap, one connection*

Creating one communication channel and all necessary program blocks in iMap increased the memory consumption EPROM to 4% (5560 b), work memory to 0,37% (974 b) and retentive memory to 0,06% (84 b).



Path: Shadow\CPU 315F-2 PN_DP\CPU 315F-2

Status: OK

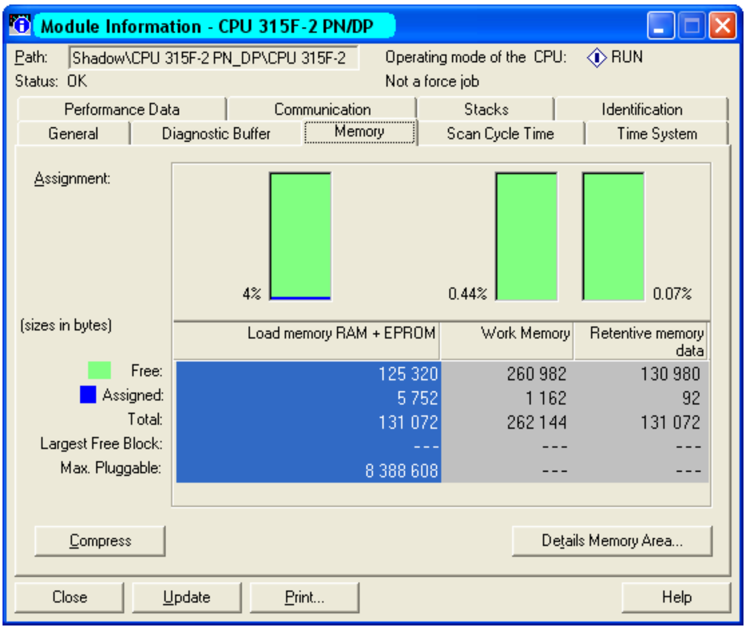
Operating mode of the CPU: RUN

Not a force job

(sizes in bytes)	Load memory RAM + EPROM	Work Memory	Retentive memory data
🟩 Free:	125 412	261 072	130 984
🟦 Assigned:	5 660	1 072	88
Total:	131 072	262 144	131 072
Largest Free Block:	---	---	---
Max. Pluggable:	8 388 608	---	---

Fig. 62 *PLC memory consumption, programmed by iMap, two connections*

Creating two communication channels and all necessary program blocks in iMap increased the memory consumption EPROM to 4% (5660 b), work memory to 0,41% (1072 b) and retentive memory to 0,07% (88 b).


Module Information - CPU 315F-2 PN/DP

Path: Shadow\CPU 315F-2 PN_DP\CPU 315F-2 Operating mode of the CPU: RUN

Status: OK Not a force job

(sizes in bytes)

	Load memory RAM + EPROM	Work Memory	Retentive memory data
🟩 Free:	125 320	260 982	130 980
🟦 Assigned:	5 752	1 162	92
Total:	131 072	262 144	131 072
Largest Free Block:	---	---	---
Max. Pluggable:	8 388 608	---	---

***Fig. 63** PLC memory consumption, programmed by iMap, three connections*

Creating three communication channels and all necessary program blocks in iMap increased the memory consumption EPROM to 4% (5752 b), work memory to 0,44% (1162 b) and retentive memory to 0,07% (92 b).

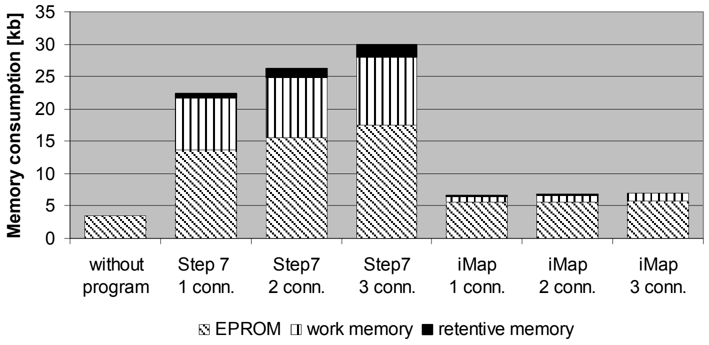


Fig. 64 *Comparing the memory consumption*

The comparing results show that a system created by Step7 is substantially larger than a system created by iMap. iMap communication system elements occupy the memory in size of bytes. Creating a communication interface with a single connection required approximately 3 kb of memory, each additional channel increased memory occupation by approximately 200 bytes. Communication elements and blocks of the Step7 system increase memory requirements by whole kilobytes. Creating communication blocks and one channel increased the program by nearly 19 kb, every other channel by 4 kb. When using the maximum number of channels, communication would need some 80 kb of the memory, only 6 kb is needed when using iMap.

The next chapter takes into account all comparison criteria and methodology generalization for selection of a suitable approach used for distributed control system design.

6. METHODOLOGY GENERALIZATION FOR SUITABLE APPROACH SELECTION

When deciding which standard to use in designing of distributed control system, it is necessary to consider several aspects. Very important are required parameters of the designed system.

6.1 Size of the distributed system

Basic parameter during decision making regarding the use of IEC 61131 standard, or PROFINET CBA is the size of designed distributed system. When using IEC 61131 approach and programming tool Step7, it is possible to create a distributed system, which has a maximum of 16 communication channels. It means in practice:

e) Creation of the system with maximum number of 16 channels, which are using one-way communication;

f) Creation of the system with maximum number of 8 stations, which are using two-way communication.

Each sizable system must be created by PROFINET CBA standard and iMap tool.

Size of the distributed system affects other system parameters, which are critical factors in choosing a better approach to design of the system.

These include:

g) remote system response time,

h) network utilization,

i) program impact on PLC cycle time,

j) program impact on PLC memory consumption,

k) system hardware network resources,

l) costs.

6.2 Remote system response time

Remote system response time is data that describes station reaction time to change of the system status. Systems designed by both methods behave as follows:

- System designed according to IEC 61131 standard has minimum reaction time (in case of single communication channel) of approximately 22 ms. Any other communication channel increases the response time constantly, approximately about 7 ms. When achieving the maximum number of communication channels, the reaction time of the stations will be approximately 120 ms. This time is not dependent on the volume or data transmitted through the communication channel.
- System designed according PROFINET CBA standard has a reaction time of the remote stations of approximately 24 ms. Reaction time is not dependent on the number of communication channels. This time is not affected by the volume or transmitted data type through the communication channel.

When the system is small (consisting of max. of 5 stations), then the system response time is according to IEC 61131, still on the acceptable level. For a system with multiple stations, this time increases and therefore it is better to design systems according to the PROFINET CBA standard. In the case of any large distributed system, in which one of the most important parameters represents the system reaction time, it is necessary to use the PROFINET CBA standard.

6.3 Network utilization

This is a parameter that determines the data amount transmitted through the communication network. The aim is to minimize this quantity to a minimum value. Smaller amount of transmitted data means less chance for error occurrence during transmission (even if the transmission protocols deal directly with the errors) and that shorten the system reaction time.

- System designed according to the IEC 61131 standard has the network utilization on the level of 25 – 30 kB/s, which is low enough data, but it is a maximal number of data that the system can transfer per second. Every other communication channel represents more data to be delivered to the end station. Because the transmission band is limited, it will adversely affect the system reaction time.
- System designed according to the PROFINET CBA standard has for comparable communication time much lower data transmission (transmission of each channel is approximately 2 kB/s). Although the data transfer is much lower than for a system according to the IEC 61131 standard, for both approaches we speak about relatively minor data transfers. Increasing the number of communication channels for the PROFINET CBA system by higher number of channels means small data transfer. However, it does not negatively influence the system reaction time.

6.4 Program impact on the cycle time PLC

It is data that indicates the time for which the PLC process its entire program only one time. Of course, increasing the cycle time means delayed station reactions to system status change. By default, this time varies within milliseconds. The following lines describe the impact of communication method between the stations on the outcome PLC cycle time. That time does not include the impact of the control program itself, for both systems it is approximately the same time.

- For a system designed by IEC 61131 standard the minimum achievable time for one communication channel is approximately 1 ms. Every next communication channel will increase the response time about 1 millisecond. For a system with the maximum number o communication channels, the cycle time will be increased to about 16 ms.
- Increasing the number of communication channels does not have any impact on PLC cycle time for systems designed with PROFINET CBA standard. Communication takes 1 ms for any number of stations and communication channels in the system.

For systems where the cycle time is not very important, a system designed by IEC 61131 is acceptable. For systems where the reaction time is an important parameter, it is better to use a system designed by the PROFINET CBA standard.

6.5 Program impact on memory consumption for PLC

This is a parameter that determines how much memory of PLC will be taken by program blocks, which are necessary for distributed system designing. The more memory communication blocks utilize, the less memory remains for the control program itself and obviously, the total performance of the system decreases. The PLC memory capacity is usually limited to a few MB and memory cards are standardly supplied in the size of few hundred kilobytes.

- Blocks necessary to ensure communication with one station and blocks necessary for PLC running will take some 23 kB of memory for the test system designed by the IEC 61131 standard. Each subsequent communication channel increases the memory consumption by approximately 4 Kb. For sixteen communication channels it will be some 80 Kb of memory consumed only by communication blocks. For example, using 256 kB of memory card means a significant memory reduction for the control program itself.
- Blocks necessary to ensure communication with one station and blocks necessary for PLC running will take some 6 KB of memory for a system designed by PROFINET CBA standard. Each communication channel to follow increases the memory consumption by about 200 bytes. For example, for sixteen communication channels it is some 9 kB of consumed memory, which is roughly nine times less than by using the IEC 61131 standard.

If it is a distributed system where the control program is simple and takes only a small part of the station memory, or when large memory cards are available in the stations, it is possible to use the system according to the IEC 61131standard as well. If it is a high-volume program, or where the hardware memory size is limited, it is necessary to design the system under the PROFINET CBA standard.

6.6 System hardware network resources

The decision of which standard to use in distributed system development is influenced by hardware resources used in the system communication network.

When using the IEC 61131 standard, the communication is through TCP/IP protocol. This enables the building of such a system on any network resource that meet TCP/IP criteria.

- When using the PROFINET CBA standard, it is not a standard communication under TCP/IP protocol, but the PROFINET CBA industrial protocol is used. It has specially evaluated safety risks, arising from the nature of the control systems (require maximum possible system reliability), but it needs devices able to work with PROFINET CBA for its running. This protocol is not supported, for example, by network hubs. WiFi routers are functioning on the hub principle as well. When the network is using such elements, the PROFINET CBA system will not work. One option is to use either the IEC 61131 standard or network element replacement.

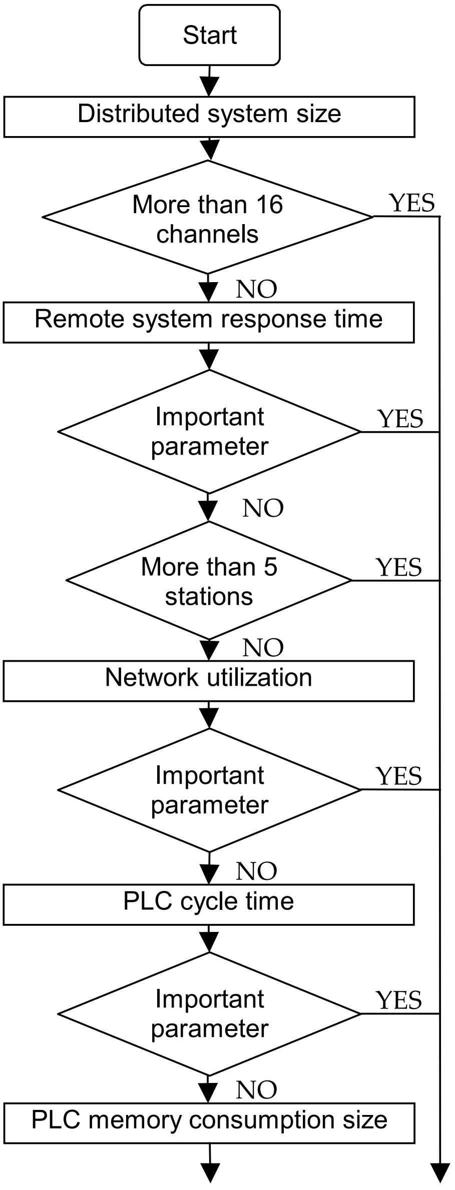
6.7 Costs

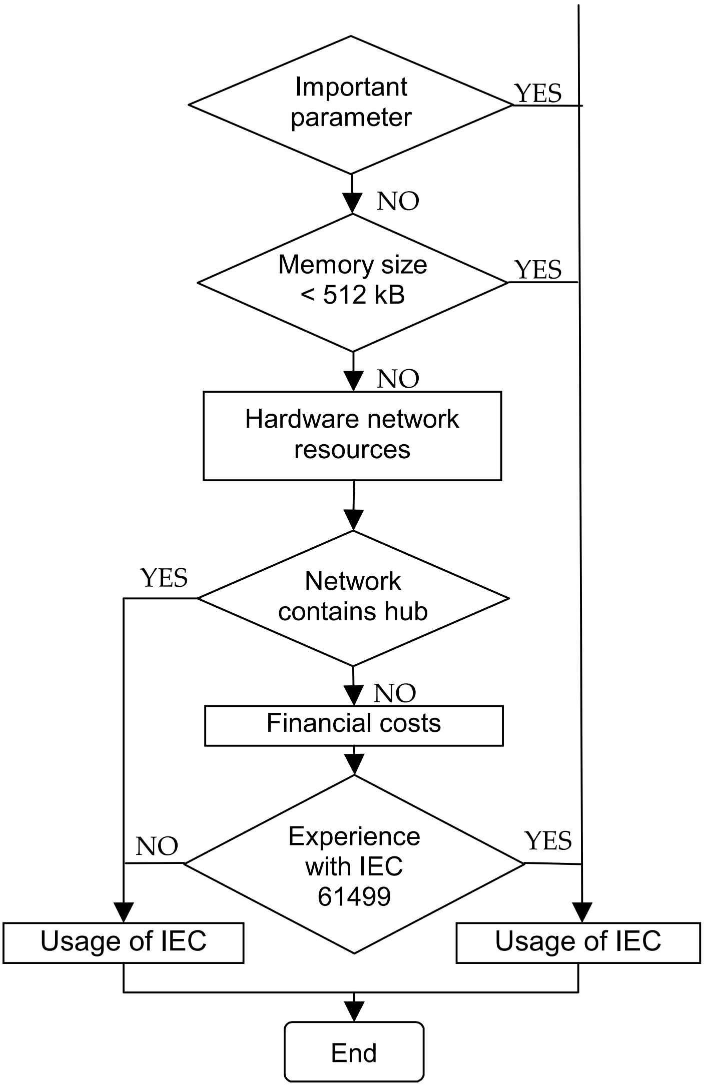
Using a different approach for system design means that financial costs may vary:

- When using IEC 61131 standard, it is necessary to buy a software package, Step7, that has been used for many years and each company working with Simatic devices must have already purchased this package. Additional investments in other program packages are not necessary. However, increased complexity of system design means higher labour costs.
- When using the PROFINET CBA standard, it is necessary in addition to the Step7 package to buy another software package, iMap. Its price ranges within hundreds of euros. Because this is a relatively new tool, most companies do not own it as of yet. It must be remembered that only a few people are skilled in using this tool and all others must be trained. In contrast, the resulting program is much simpler and therefore, the development costs are lower than those generated by a system designed according to the IEC 61131 standard.

6.8 Selecting the suitable standard

The entire decision-making process can be depicted as follows:


Start
Distributed system size
More than 16 channels
YES
NO
Remote system response time
Important parameter
YES
NO
More than 5 stations
YES
NO
Network utilization
Important parameter
YES
NO
PLC cycle time
Important parameter
YES
NO
PLC memory consumption size


Important parameter
YES
NO
Memory size < 512 kB
YES
NO
Hardware network resources
YES
Network contains hub
NO
Financial costs
NO
Experience with IEC 61499
YES
Usage of IEC
Usage of IEC
End

As shown, in most of the criteria the IEC 61499 standard achieves better results. It is not always possible to use one of the latest approaches when using some of the older network components. Also, in the case of small systems, the financial costs related to purchasing of important development tools and developers' training may determine if the IEC 61131 standard should be used.

CONCLUSION

This publication deals with norms and standards for industrial automation, mainly regarding the IEC 61131 and IEC 61449 standards and their application in practice by distributed control systems design. In the beginning, these standards were described in detail and their further development was indicated. Subsequently, the criteria for each standard was looked at as it plays a significant role in determining which standard is better to use. Based on the criteria proposal, the methodology for application comparison for both standards used by distributed control system design in practice, was formed. This methodology was verified on a real control system that was created by using both systems. Based on the system test results, methodology for suitable approach selection was generalized and a procedure for suitable standard selection was determined.

The main criteria taken into account were: remote station response time, network utilization, affect of the approach on consumed memory size and processor cycle time. During the verification, another factor occurred when hardware incompatibility was detected. It is the selection of hardware network components. The last evaluation criterion was the costs of both systems.

As a model system of interconnected devices the Simatic S7-300 devices made by Siemens were chosen because these can work under the IEC 61131 standard as well as under the PROFINET CBA standard, which represents implementation of the IEC 61499 standard for Siemens. All mentioned criteria were tested for the chosen system. IEC 61499 standard showed better results in almost all areas. When older network components were used within the network, it was not possible to use this standard in some cases. Differences were not great when small distributed control systems were created. With the increase in the number of stations, the IEC 61131 standard still showed worse results. The disadvantage of the IEC 61499 is that it is still not widespread. System developers need to be trained and gain experience, which requires considerable financial costs and a large amount of time. Also, the majority of developers do not own the tools needed for system development according to the IEC 61499 standard. That is why it is necessary to purchase them and their prices vary from hundreds to thousands of euros. Therefore, for small systems with low output requirements, it still seems better to use the IEC 61131 standard. Once the IEC 61499 standard is widely used on a global scale, use of the IEC 61131 standard for development of distributed control systems will taper off. This is not expected to happen for several years.

This is just one view of the complex problem with distributed control systems. To make the comparison more simpler, only components made by single manufacturer were used. Distributed systems may be formed from the devices made by different manufacturers and it does not necessary to only be programmable logic controller. Also, robustness of the distributed control systems can be much greater. Theoretically, it can consist of hundreds of cooperating control stations located in multiple areas. This also offers the opportunity for future research on this topic. For that reason, the next activities should focus on:

- distributed systems consisting of devices from multiple manufacturers and multiple types
- distributed systems consisting of more devices,

- distributed systems where individual stations are divided into multiple communication networks.

REFERENCES

[1] BAKSHI, V. U. *Control Systems*. Pune: Technical Publications. 2008. ISBN 81-8431-289-X

[2] BÉLAI, I., DRAHOŠ, P. The Industrial Communication Systems PROFIBUS and PROFINET. In *Proceedings of Applied Natural Sciences 2009*. Trnava: Univerzita Sv. Cyrila a Metoda, 2009. ISBN 978-80-8105-127-2

[3] BOLTON, W. *Programmable Logic Controllers.* Burlington: Newnes, Elsevier Ltd., 2009. ISBN 978-85617-751-1

[4] BOLTON, W. *Instrumentation and Control Systems*. Burlington: Newnes, Elsevier Ltd., 2004. ISBN 978-7506-6432-5

[5] BOSCH REXROTH CORPORATION. Understanding the IEC61131-3 Programming Languages [online] [cit. 18.7.2010] Available online at <http://www.automation.com/pdf_articles/IEC_Programming_Thayer_L.pdf>

[6] DAI, W.W., VYATKIN V. A Case Study on Migration from IEC 61131 PLC to IEC 61499 Function Block Control. In *Proceedings of Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference.* Cardiff, Wales, 2009. pp. 79–84. ISBN 978-1-4244-3759-7

[7] ELECTRO CAM CORP. *Scan Times: PLC vs. PLuS*. [online] [cit. 19.7.2010] available online at <http://www.electrocam.com/pdf/tech/SCANTIME.PDF>

[8] *FBench description,* [online] [cit. 19.7.2010] Available on online at <http://fbench.wikispaces.com/>

[9] FBENCH PROJECT TEAM. *FBench Project homepage.* [online] [cit. 19.7.2010] Available online at <http://www.ece.auckland.ac.nz/~vyatkin/fbench/index.html>

[10] *Good Automated Manufacturing Practice*, [online] [cit. 19.7.2010] Available online at <http://www.appliedintegration.co.uk/services/gamp/>

[11] IEC. IEC 61131-1 Programmable Controllers – Part 1: General Information. Geneva: International Electrotechnical Commision, 2003. ISBN 2-8318-7039-9

[12] IEC. IEC 61131-3 Programmable Controllers - Programming Languages. Geneva: International Electrotechnical Commision, 2003. ISBN 2-8318-6653-7

[13] IEC. IEC 61131-5 Programmable Controllers – Part 5: Communications. Geneva: International Electrotechnical Commision, 2000. ISBN 2-8318-5510-1

[14] IEC. IEC 61131-8 Programmable Controllers – Part 8: Guidelines for the application and implementation of programming languages. Geneva: International Electrotechnical Commision, 2003. ISBN 2-8318-7210-3

[15] IEC. IEC 61499-1: Function blocks - Part 1: Architecture. Geneva: International Electrotechnical Commision, 2005. ISBN 2-12-4869111-6

[16] IEC. IEC 61499-1: Function blocks - Part 2: Software tool requirements. Geneva: International Electrotechnical Commision, 2005.

[17] *ISPE launch GAMP 5 Good Automated Manufacturing Practice*, [online] [cit. 19.7.2010] Available online at <http://tern-quay.com/Docs/GAMP5_is_here.pdf>

[18] JOHN, K-H., TIEGELKAMP M. *IEC 61131-3: Programming Industrial Automation Systems*. Regensburg, German: Springer-Verlag, 2001. ISBN 3-540-67752-6

[19] JOHNSON, C. D. *Process Control Instrumentation Technology*. Pearson/Prentice Hall, 2006. ISBN 978-0131194571

[20] KALUŽA, F. Methodologies, Metrics and Evaluation criteria for effective ISMS. Part II. In *Security revue*, 2008. ISSN 1336-9717

[21] KALSI, H. S. *Electronic Instrumentation*. New Delhi: Tata McGraw-Hill Publishing Company Limited, 2004. ISBN 978-0-07-058370-2

[22] KAMEN, E. W. *Industrial Controls and Manufacturing*. San Diego: Academic Press, 1999, ISBN 0-12-394850-9

[23] LEONARD, J. *Systems Engineering Fundamentals: Supplementary Text*. Fort Belvoir, Virginia: Defense Systems Management College Press, DIANE Publishing, 2001. ISBN 9781428996113

[24] LEWIS, R. *Modelling control systems using IEC 61499.* London: The Institution of Engineering and Technology, 2001. ISBN 0852967969

[25] LEWIS, R. *Programming industrial control systems using IEC 1131-3 Revised edition*. London: The Institution of Electrical Engineers, 1998. ISBN 0852969503

[26] MAJOR, L. Standardisation and Quality management systems in software and systems engineering. In *ATP Journal* 7/2004, pp. 100-103. Bratislava: HMH. ISSN 1335-2237

[27] MILLER, R., MILLER, M. R. Industrial Electricity and Motor Controls. New York: McGraw-Hill Professional, 2007. ISBN 978-07-154476-4

[28] MORAVČÍK, J. *Directive GAMP – Forms processing and presentation*. The Thesis, 2010. Trnava: UIAM MTF STU, pp. 10-11.

[29] NAGRATH, I. J. *Control Systems Engineering*. New Delhi: New Age International, 2006. ISBN 81-224-1775-2

[30] OEM Technology Solutions PTY LTD. *ISaGRAF v3.5 overview*, [online] [cit. 19.7.2010] Available online at <http://www.rabbit.com/documentation/docs/appkits/EmbeddedPLC/ISaGRAF35_Overview_V1.pdf>

[31] PESSEN, D. W. *Industrial automation: circuit design and components.* Wiley-IEEE, 1989. ISBN 0-471-60071-7

[32] PIGAN, R., METTER. M. *Automating with PROFINET: Industrial communication based on Industrial Ethernet.* Erlangen: Publicis Publishing, 2008. ISBN 978-3-89578-294-7

[33] PLC LADDER. *Scan Time of PLC.* [online] [cit. 19.7.2010] Available online at <http://program-plc.blogspot.com/2010/02/scan-time-of-plc.html>

[34] PHIPPS, C. A. *Fundamentals of electrical control.* The Fairmont Press, Inc. Lilburn, GA, USA. 1998. pp. 119-124. ISBN 0-88173-312-1

[35] PROFIBUS Working Group "Communication Function Blocks". PROFIBUS Communication and Proxy Function Blocks acc. To IEC 61131-3. Karlsruhe: PROFIBUS Nutzerorganisation e.V., 2001

[36] PROFIBUS Working Group 10 "PROFINET CBA". PROFINET CBA – Architecture Description and Specification, Karlsruhe: PROFIBUS Nutzerorganisation e.V., 2004

[37] REAL TIME AUTOMATION. PROFINET CBA – An Innovative Distributed Automation Solution, [online] [cit. 18.7.2010] Available online at <http://www.rtaautomation.com/profinetcba/ProfiNet_CBA_Overview_R2.pdf>

[38] ROHNER, P. *Automation With Programmable Logic Controllers*. Sydney: UNSW Press, 1996. ISBN 86840-287-7

[39] SAMK. *S7 Memory Areas,* [online] [cit. 19.7.2010] Available online at <http://www.tp.spt.fi/~salabra/ha/Automaatiotekniikka/S7-Memory.pdf>

[40] SARDESAI, A. R., MAZHARULLAH, O., VYATKIN, V. *Reconfiguration of Mechatronic Systems enabled by IEC 61499 Function Blocks.* [online] [cit. 2010-07-19] Available online at <http://www.araa.asn.au/acra/acra2006/papers/paper_5_24.pdf>

[41] SIEMENS AG. Component based Automation Creating PROFINET Components. Nürnberg: Siemens AG Nürnberg. 2003. A5E00248719-01

[42] SIEMENS AG. Component based Automation Commissioning Systems. Nürnberg: Siemens AG Nürnberg. 2003. A5E00248818-01

[43] SIEMENS AG. Component based Automation Configuring Plants with SIMATIC iMap. Nürnberg : Siemens AG Nürnberg. 2003. A5E00248797-01

[44] SIEMENS AG. Programming with STEP 7 - Manual. Nürnberg: Siemens AG Nürnberg. 2006. A5E00706944-01

[45] SIEMENS AG. Statement List (STL) for S7-300 and S7-400 Programming - Reference Manual. Nürnberg: Siemens AG Nürnberg. 2006. A5E00706960-01

[46] SIEMENS AG. Configuring Hardware and Communication Connections with STEP7 - Manual. Nürnberg : Siemens AG Nürnberg. 2006. A5E00706939-01

[47] SIEMENS AG. SIMATIC iMap STEP 7 AddOn Creating PROFINET Components. Nürnberg : Siemens AG Nürnberg. 2008. A5E00716547-02

[48] SIEMENS AG. Tools for configuring and programming SIMATIC Controllers.

[49] Brochure. Nürnberg: Siemens AG Nürnberg. 2009. 6ZB5310-0MM02-0BA6

[50] SIEMENS AG. *Engineering communications – across all boundaries: SIMATIC iMap.* [online] [cit. 2010-07-19]. Available online at <http://www.automation.siemens.com/>

[51] *SPE Glossary of Pharmaceutical and Biotechnology Terminology Glossary of Applied Terminology for the Pharmaceutical Industry*, [online] [cit. 2010-07-19]. Available online at <http://www.ispe.org/glossary?term=Good+Automated+Manufacturi ng +Practice+%28GAMP%29>

[52] STRASSER, T., MULLER, I., SCHUPANY, M., EBENHOFER, G., MUNGENAST, R., SUNDER, C., ZOITL, A., HUMMER, O., THOMAS, S., STEININGER, H. *An Advanced Engineering Environment for Distributed & Reconfigurable Industrial Automation & Control Systems based on IEC 61499.* [online] [cit. 2010-07-19]. Available online at <http://conference.iproms.org/sites/conference.iproms.org/files/PID155260.pdf>

[53] SWAINSTON, F. *A Systems Approach to Programmable Controllers*. Albany, New York: Cengage Learning, 1992. ISBN 0-8273-4670-0

[54] TICKIT SCHEME, [online] [cit. 19.7.2010] Available online at <http://www.tickit.org/scheme.htm>

[55] VYATKIN, V. *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design.* Ottawa, Canada: O3NEIDA – Instrumentation Society of America, ISBN 978-0-9792343-0-9

[56] VYATKIN, V., CHOUINARD, J. On comparisons of the ISaGRAF implementation of IEC 61499 with FBDK and other implementations. In *Proceedings of Industrial* June 2008, pp. 289–294. ISBN 978-1-4244-2170-1

[57] VYATKIN, V. The Potential Impact of the IEC 61499 Standard on the Progress of Distributed Intelligent Automation. In *International Journal of Manufacturing Technology and Management,* 2006, *vol. 8*, pp. 107-125. ISSN 1368-2148

[58] YANG, W. *Implementation of IEC61499 Distributed Function Block Architecture for Industrial Measurement and Control Systems (IPMCS).* [online] [cit 2010-07-19] Available online at <http://www.holobloc.com/stds/iec/tc65wg6/meetings/fla03/yangwei.pdf>

[59] ZIMMERMAN, G. P. Programmable Logic Controllers and Ladder Logic. Rapid City: Dr. Alfred R. Boysen, Department of Humanities, South Dakota School of Mines and Technology, 2008

[60] ZOITL, A. *Real-Time Execution for IEC 61499*. Ottawa, Canada: O3NEIDA – Instrumentation Society of America. ISBN 978193439-4274

[61] ZOITL, A., STRASSER, T., HALL, K., STARON, R., SUNDER, CH., FAVRE-BULLE, B. The Past, Present, and Future of IEC 61499, Holonic and Multi-Agent Systems for Manufacturing. In *Third International Conference on Industrial Applications of Holonic and Multi-Agent Systems.* Regensburg, German: Springer-Verlag, 2007. pp 1-15. ISBN-10 3-540-74478-9

CONTENTS

Abstract	5
LIST OF ACRONYMS	6
LIST OF SYMBOLS	8
INTRODUCTION	9
1. CURRENT STATUS IN INDUSTRIAL AUTOMATION STANDARDS	11
1.1 IEC standards	11
1.1.1 IEC 61131 standard	13
1.1.2 IEC 61499 standard	14
2. IEC 61131 AND IEC 61499 STANDARDS AND THEIR FUTURE DEVELOPMENT	17
2.1 IEC 61131	19
2.2 IEC 61499	28
2.2.1 System design	31
2.2.2 Models and concepts	33
2.3 PROFINET CBA	38
2.4 Future development of IEC 61499 standard	43
3. PROGRAMMING TOOLS SUPPORTING IEC 61499	46
3.1 FBDK	46
3.2 ISaGRAF	47
3.3 Fbench	48
3.4 Tool implementing PROFINET CBA	49
3.5 Future development of tools implementing IEC 6149951	54
4. METHODOLOGY PROPOSAL FOR COMPARISON OF THE APPLICATION STANDARDS AND CRITERIA SELECTION	53
4.1 Platform selection	54
4.2 Criteria selection for comparing	55
4.3 Creating the distributed system	59
4.3.1 Creating the distributed IEC 61131 system	63
4.3.2 Creating the distributed IEC 61499 system	60
4.4 Testing of the selected criteria	60
4.4.1 Procedure for response time measuring	60

4.4.2 Procedure for measuring network utilization 60
4.4.3 Procedure for measuring the program impact on the PLC cycle time 61
4.4.4 Procedure for measuring the program impact on the size of PLC consumed memory 61
4.5 Comparing the tested criteria 62
4.6 Generalizing the methodology for suitable approach selection 62
5. METHODOLOGY VERIFICATION 63
5.1 Platform selection 63
5.2 Creating the experimental control system 65
5.3 Creating IEC 61131 system 66
5.3.1 Creating the hardware configuration 67
5.3.2 Creating the network interface system 68
5.3.3 Creating the communication blocks 68
5.4 Creating PROFINET CBA system 72
5.5 Comparing the response time of the remote stations 75
5.5.1 Measuring the system response time by Step7 and NetPro 76
5.5.2 Measuring the system response time by iMap 78
5.6 Comparing the network utilization 80
5.7 Comparing the program impact on the PLC cycle time 87
5.8 Comparing the program impact on the PLC memory size 93
6. METHODOLOGY GENERALIZATION FOR SUITABLE APPROACH SELECTION 101
6.1 Size of the distributed system 101
6.2 Remote system response time 102
6.3 Network utilization 103
6.4 Program impact on the cycle time PLC 104
6.5 Program impact on memory consumption for PLC 105
6.6 System hardware network resources 106
6.7 Costs 107
6.8 Selecting the suitable standard 107
CONCLUSION 111
REFERENCES 114