Integration of P-NET into IEC 1131

Martin Wollschlaeger

Abstract

The international standard IEC1131 is widely used in programming of PLC-based automation systems. While a PLC mostly is a concentrated device, a fieldbus system appears as a distributed automation system. Since IEC1131 covers data exchanging between distributed modules, a number of fieldbus system developers offer tools and environments to handle the bus system in accordance to IEC 1131. Following a short description of the main principles defined in IEC 1131, the specific features of P-NET are characterised. A way for the integration of P-NET into IEC 1131 is offered. The prerequisites necessary for the integration are described, as well as the integration of a programming environment into existing P-NET tools.

1. Introduction

Programming of a PLC according to IEC1131 has become an industrial standard. Nearly every developer of a PLC offers tools and strategies for programming his brand of a PLC following the rools defined in IEC1131. Using this way, a common view to PLCs was established. The customer may program different PLCs of different vendors in the same way. Especially the graphical programming languages offer a convenient programming environment.

Fieldbus systems may be treated in a exactly the same way. A complex fieldbus system is just a „distributed PLC“ with a set of hard- and software components and a built-in communication system. Like a PLC, a fieldbus system has to be configured in order to solve an automation problem. There are a lot of different tools with totally different user interfaces, even for modules of the same bus system. But, if a bus system is similar to a PLC, why shouldn't the customer use models, definitions, programming languages and development tools according to IEC1131? If he does so, he will get a view to a bus system, where nothing of explicit communication is to be seen. All communication processes are performed hidden for the user. So the real user interface are the automation functions, their handling is part of an automation specialist's knowledge.

In P-NET, there are some special features that provide starting points for an integration of this fieldbus system into IEC1131. They are discussed below.

2. Overview of the IEC 1131

2.1. Basics and terms

The software model

Programming a PLC according to IEC1131 is based on a standardised software model. Figure 1 illustrates the model and the related data flow. The elements of the model may be devided into two groups - function related and organisation related.

Dr.-Ing. Martin Wollschlaeger
Institut für Prozeßmeßtechnik und Elektronik (IPE)
Otto-von-Guericke-University Magdeburg
P.O. Box 4120
39016 Magdeburg
GERMANY
Tel.: +49 (391) 67-1 46 53
Fax: +49 (391) 5 61 63 58
e-mail: mw@ipe.et.uni-magdeburg.de

Function related elements are:

- **programs**,
- **functions** and
- **function blocks**

These elements are *programmed* using the languages defined in the standard.

Elements related to organisation tasks are

- **configurations**,
- **resources**,
- **tasks**,
- **global variables** and
- **data access paths**,

that are *declared* using defined declaration terms.

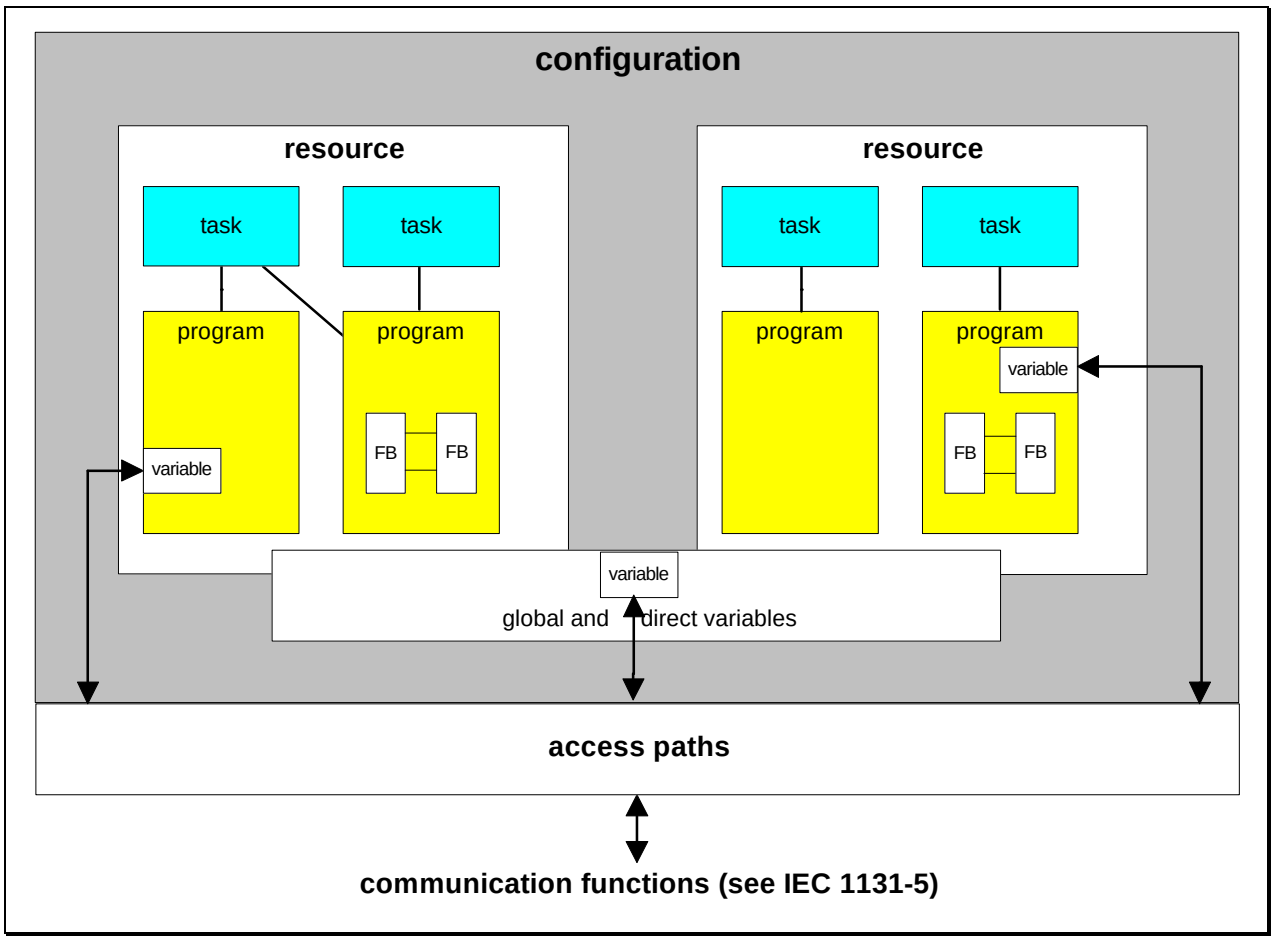

Figure 1: The software model of IEC1131

Configurations and resources

The entire structure of an automation system may be broken down into smaller units, that are easier to be managed. So a hierarchical structure of the system may be designed (figure 2). Each of these units represents a configuration. A configuration can communicate with others via interfaces (see figure1). Typically a configuration describes a PLC with its specific features, including the software implemented as firmware and that implemented by the user. The configuration consists of one or several resources.

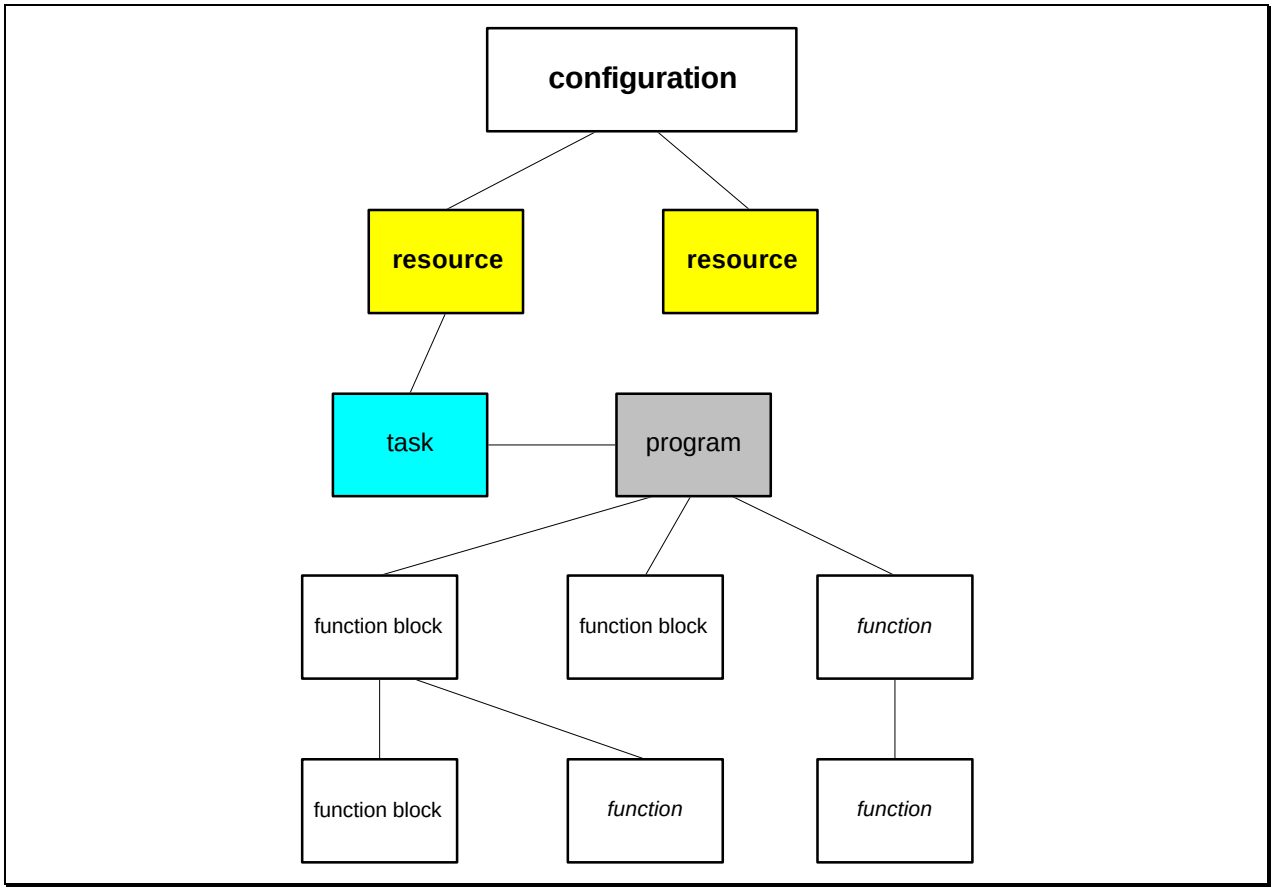

Figure 2: Hierarchical software structure of an IEC1131-according system

A resource usually describes a processor-based functionality. It is equivalent to signal processing functions as well as functions of the sensor/actuator interfaces.

Tasks and programs

As a resource typically consists of a processor-based function, an appropriate operating system allows the execution of several tasks. These tasks may have specific attributes influencing their operation - cyclic, interrupted, assigned priority, starting conditions and so on. Tasks can invoke programs to do the necessary calculation. Programs themselves can activate function blocks and functions. A part of a program may be the declaration of variables for data exchange between function blocks, functions and the hardware.

Function blocks and functions

Function blocks and functions are the units, that do the explicit automation functions. A function is used to calculate a result on given input values without having internal store capabilities. So the

result only depends on the input values and is every time the same, when the function is called with the same inputs.

A function block has the ability to store values in an own memory area. So the result of a function block may depend on the inputs as well as on previous calls of the block. Using this way time dependent calculations and iterative functions may be performed (e.g. PID-calculators).

Data representation

The data representation is similar to that used in modern high level languages. There are simple data types (integer, word, byte, Boolean and so on) as well as structured types like arrays and records. The declaration of types and variables is nearly the same like in other languages. Its notation is oriented to Pascal. Some special characters in a declaration statement describe the physical representation of the data, like position in input or output registers or absolute addresses.

2.2. Programming Languages

The focus of IEC1131-3 is the definition of five „programming languages“ used to describe the automation problem, which is to be solved using a PLC-based system. The programming-lan guages can bedevided into textual and graphical languages. Graphical languages are Sequential Function Chart (SFC), Ladder Diagram (LD) and Function Block Diagram (FBD), while textual languages are Instruction List (IL) and Structured Text (ST). Details of the different languages are described below.

Sequential Function Chart

Sequential Function Chart (SFC) allows the programming of sequence-oriented problems. Therefor the problems are described using steps and transitions. Steps represent actions, while transitions define conditions. These conditions have to be fulfilled before moving from one step to another. Steps can be performed in parallel and priority may be assigned to them. The definition of steps and transitions may be done using the other four languages. Figure shows a fragment of a program defined with SFC.

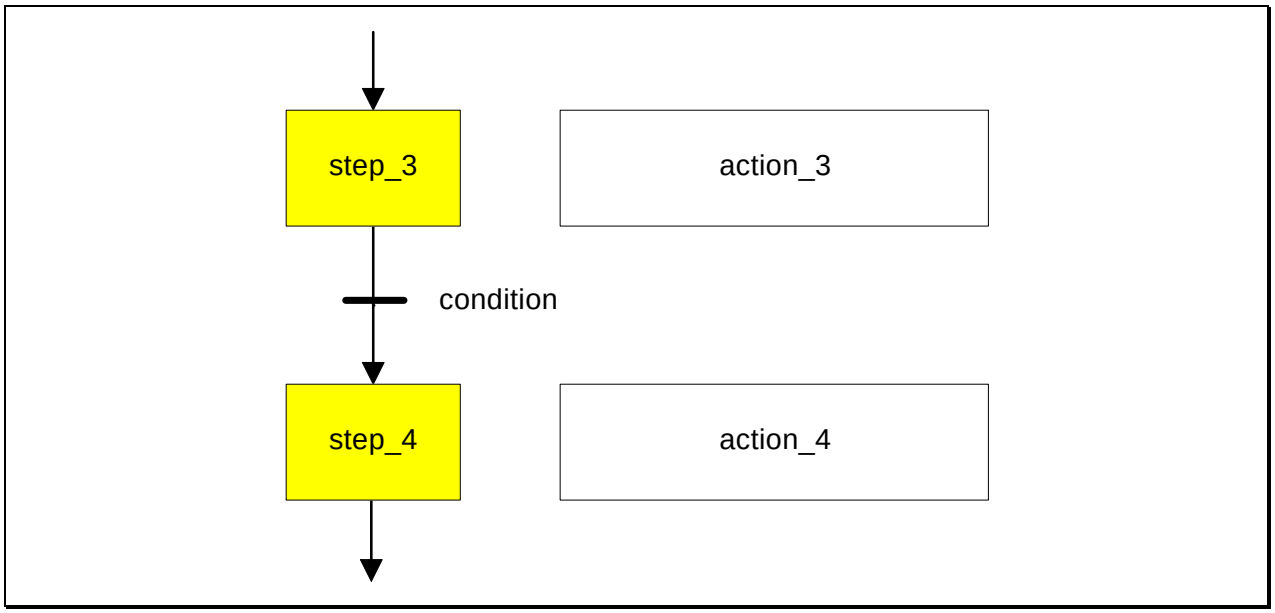

Figure 3: Fragment of an SFC-coded program

Ladder Diagram

A Ladder Diagram (LD) is often used for binary logic coding. Figure 4 shows a small example that could be part of a complex logic sequence in a loop controller. Elements of this language are typically binary contacts, like normally opened or normally closed contacts, and coils. The elements are placed on a drawing sheet connecting vertical lines representing a current flow.

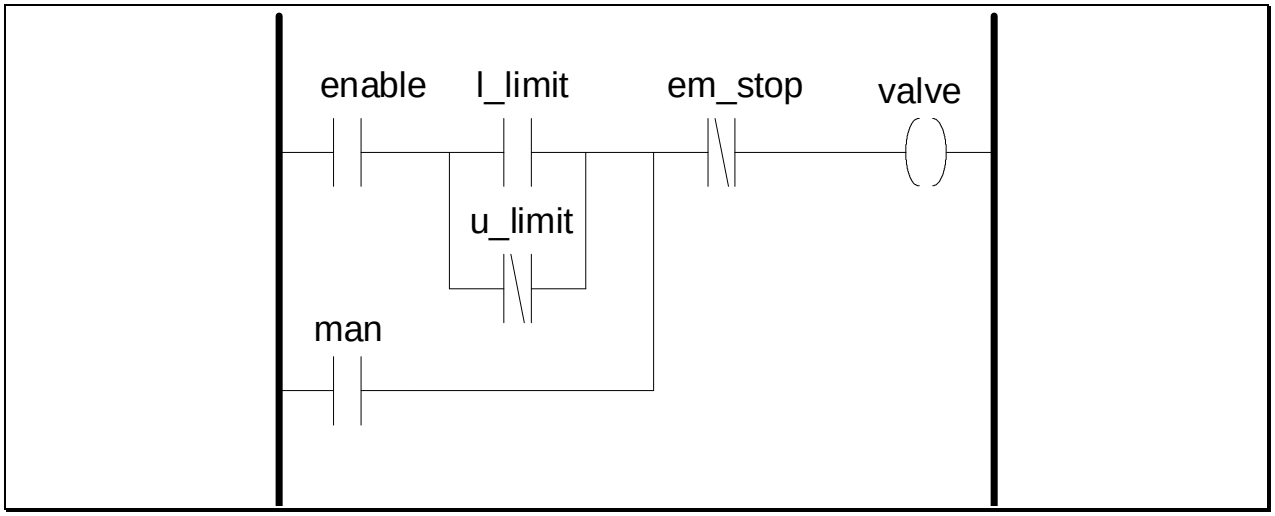

Figure 4: Example for a ladder diagram

Function Block Diagram

The Function Block Diagram (FBD) is based on function blocks, whether predefined (standard function blocks) or self defined (hierarchical blocks or macros). These function blocks represent automation functions operating on both digital and analogue variables. A number of predefined function blocks are implemented in most of different PLC systems. A program coded with FBD looks like a circuit diagram (figure 5), which is widely used.

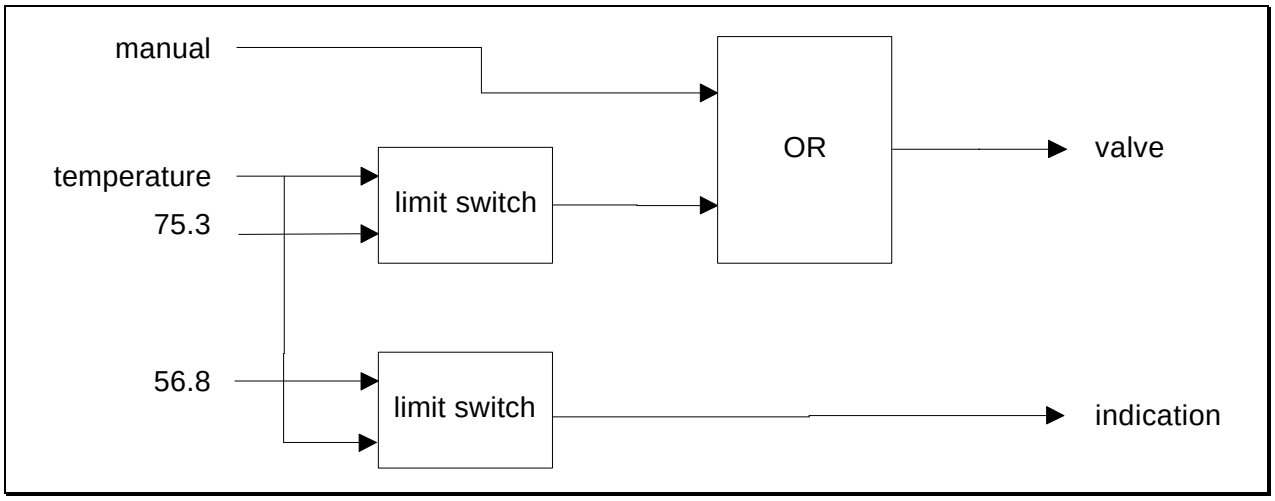

Figure 5: Program coded using FBD

Instruction List

The structure of programs coded with Instruction List (IL) looks like an assembler program using standardised operators (figure 6), expanded with symbolic identifiers and some structuring information like functions and record-like notation for variables in function blocks.

```
LD      %IW4        (* load word from direct input *)
ST      CALC.X      (* store to parameter X of function block CALC *)
CAL     CALC        (* call function block *)
...
```

Figure 6: Program coded using Instruction List

Structured Text

Programming in Structured Text (ST) is, as an opposite to Instruction List, coding of algorithms using a high-level language (figure 7). This language contains all essential elements of a modern programming language, including instructions for branch and iteration.

```
ALARM:=FALSE;               (* initial state *)

TEMP:=%IW4;                 (* read temperature value *)

IF TEMP<5.0 THEN
        ALARM:=TRUE;        (* value less min level *)
ENDIF

%OX1:=ALARM;                (* handle alarm output contact *)
...
```

Figure 7: Program coded using Structured Text

3. Integration of P-NET

3.1. Prerequisites

3.1.1. Fundamental relations between P-NET and IEC 1131

A P-NET system can be considered as a distributed automation system consisting of different modules with different functionality. Every module with its specific features can be treated as a configuration. The configurations are characterised by specific features, e.g. timing conditions, implemented functionality, operating system and so on. Depending on the kind of firmware implementation, modules may be devided into those with predefined sets of functions (slave modules) and those programmable by the user (masters). Variables may reside in every module and are exchanged by means of the bus system. Tasks and programs are performed within the modules, pre-programmed by the manufacturer or even user-programmable.

3.1.2. Features of P-NET similar to IEC 1131

There are some features in P-NET, that are similar to the terms of IEC 1131. The most important feature is the channel definition in P-NET. A channel can be treated as a function block. It represents a predefined automation function with a set of input and output values, with an internal memory and with parameters influencing the calculation of the result represented by the output values. Figure 8 shows on its upper part a PID-calculation circuit built of P-NET channels, on its lower part one built by function blocks.

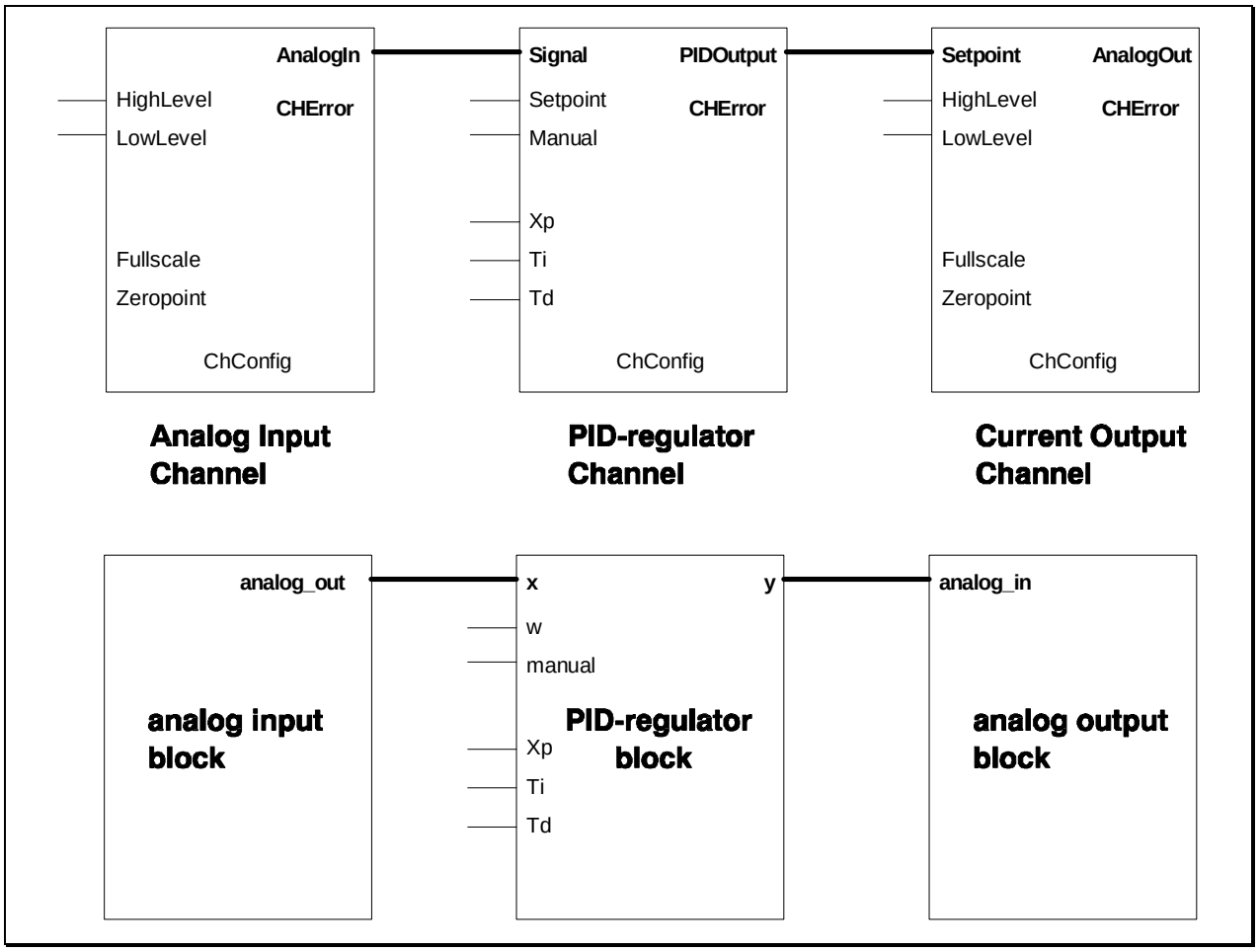

Figure 8: Calculation circuit with PNET channels and function blocks

There are channels for performing organisation functions. First one is the program channel for handling a program in a module following the definitions in MMS. Using this channel as a „description- and control-unit“ of a program allows a flexible control of a program. The second channel is the data channel, which can be treated as communication function block for receiving and transmitting data as described in IEC1131-5.

Data representation in PNET is in most cases according to IEC1131. The concept of thesoft-wire numbers provides hardware independent use of variables, but allows the use of direct ad dressing for special purposes.

Master modules in PNET can be programmed with PROCESS PASCAL in order to perform the application specific automation functions. Part of the syntax of PROCESS PASCAL are decla ration statements for physical representation of variables as well as for declaration of tasks. The task statements allow cyclic and interrupted tasks with a wide range of starting conditions. Priorities are used and communication between tasks is done using global variables.

Another similarity can be found by comparing the syntax of the programming language for the calculator channel in a UPI and a program coded by Instruction List.

3.2. Steps of an integration

The tasks that have to be performed for an integration of PROFINET may be devided into two parts. The first one is a description of the features implemented in PROFINET, the second one is the integration of these definitions into tools.

3.2.1. Description of the features

A mandatory task in order to make a user able for programming PROFINET systems according to IEC 1131 is to build up a description of the capabilities and features of PROFINET and its modules. Therefor at least the following actions have to be performed:

- **definition and implementation of predefined functions**
- **description of the capabilities of the modules**
- **definition of data access paths**

The definition of predefined functions covers the description of the channels' capabilities. For this task two different methods have to be applied. The first one is the description of the „fixed" channels implemented in slave modules. The term „fixed" stands for channels, which are not user-programmable. So an analogue input channel of a UPI is a fixed channel, while the calculator channel can be programmed by the user. Because of the channels' similarity to function blocks in most cases only a set of initialisation values has to be defined, depending on the internal structure of a channel. Figure 9 illustrates the definition of different function blocks based on the same channel.

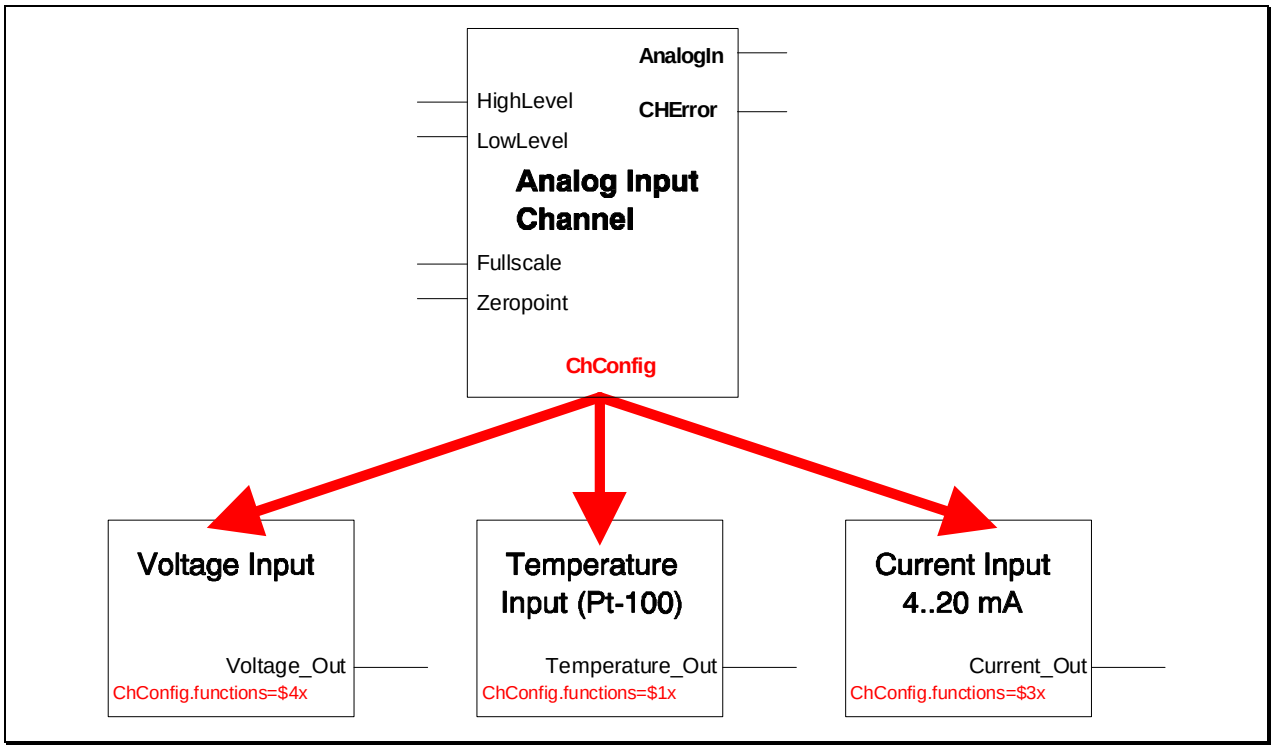

Figure 9: Definition of function blocks in slave modules

The second method is the definition and implementation of function blocks in programmable channels or master modules. Those function blocks not only have to be described, but also have to be implemented. That means, that some code has to be written using existing tools like PROCESS PASCAL for the master modules or the CALCASM-utility for the calculator channel

of the UPI. It depends on the integration of the function blocks into tools, in which way the implementation is performed. The easiest way is to define statements in PROCESS PASCAL, which have to be included in a compilation process after the configuration of the whole P-NET system. Another method is to produce pre-compiled or downloadable code, that - in conjunction with some additional software - can be linked or download into the modules. A third way would be to implement the function blocks or functions into the firmware.

Design and implementation of functions and function blocks include the creation of a machine readable description. The description should be done using existing standards of device description languages (DDL), and will lead to a library of functions and function blocks. In addition, the textual and graphical representation of functions and function blocks may be defined here.

The description of the modules' capabilities has to be performed in order to build up a library of existing configurations. This includes specific features of the different modules, like timing restrictions, maximum number of tasks, amount of usable memory and so on. Included into this description should be a module identification. In conjunction with that step the description of organisation tools like the program channel have to be performed.

The definition of access paths to variables covers the description of existing global variables in the modules with their data types and access rights as well as a the implementation of function blocks for explicit receiving and transmitting of data. Therefor the data channel is to be considered in order to build up SEND and RECV function blocks defined in IEC1131-5.

All activities above are necessary, if a user wants to program a P-NET system using the languages defined in IEC1131. The information provided with the described libraries can be treated as a firmware extension and is the basic input for any kind of programming tools. It has to be assembled once and has to be updated for every new module, what is a task of the developer of that module.

3.2.2. Integration into tools

It is the intention to use the definitions described above for the programming of a P-NET system using the described languages. Therefor some prerequisites have to be established. Different ways can be considered here, too.

The first one is to use the defined libraries in existing, IEC1131 specific development environments. This has the advantage, that tested and certified software packages can be used, which are fully compliant to IEC1131. As a prerequisite in that way an interface layer between the P-NET and the P-NET related libraries at one side and the programming tool at the other side has to be designed and implemented. An interesting way to implement such an interface could be a solution based on VIGO. Then the interface can use previously made definitions of data types, channels and so on, that are part of VIGO. In addition, the exchange of data between the programming environment and the manager information base or existing VIGO tools can be made using the interface.

The second way of providing programming tools is to build up specific tools. These tools are concentrated around the manager information base. Since it is not mandatory to provide translation between the different languages, this can be done step by step. In order to use the libraries with definitions, a special layer may be necessary or the libraries have to be integrated into the manager information base. Using this way, a closer integration of existing tools like monitor, download-utility, PROCESS PASCAL and so on is possible. The disadvantage is, that all the tools have to be developed and tested.

Figure 10 shows the implementation of programming tools using both ways described above.

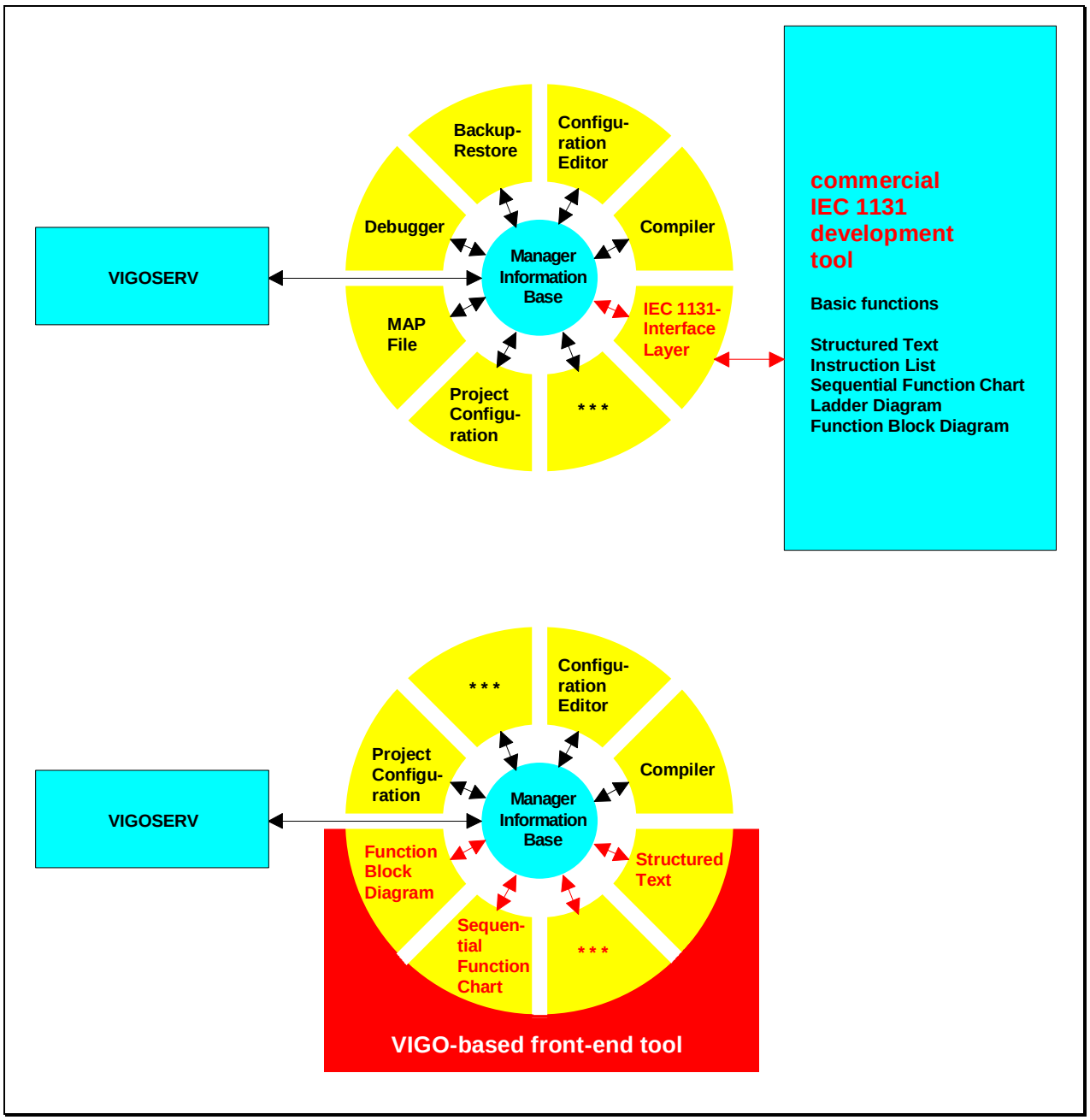

Figure 10: Integration into tools

When performing the second integration method, a very interesting idea is to use the internal VIGO-controls and expand this library with controls specific to IEC 1131-components. This would be a good solution for building dialogues in order to specify function block parameters, too.

4. Summary

As a conclusion, there are a lot of similarities between P-NET and IEC 1131. So it should be possible to integrate P-NET into this standard and enable the user to produce re-usable programs not only in a well known way, but also in a standardised, convenient environment. On the other

hand he does not need to know internal details of the system's components. Transparency and modularity of the bus system and its components increase, efforts in creating specific configured systems, their technical support and documentation will decrease.

All these facts will lead to an unified view on serial bus systems, which will increase the acceptance of those systems by the customer.

Literature

/1/ Johansen, J.:
New P-NET Channels.
3rd International conference on the P-NET Fieldbus system, Silkeborg, 25th-26th April 1995,
proceedings.

/2/ n.n.:
DIN IEC 1131: Speicherprogrammierbare Steuerungen.
Teil 3: Programmiersprachen.
Beuth Verlag GmbH Berlin, 1992.

/3/ n.n.:
Standardization in PLC programming.
Material of the PLCopen.
Zaltbommel, 1994.

/4/ Neumann, P.; Grötsch, E.; Lubkoll, C.; Simon, R.:
SPS-Standard: IEC 1131. Programmierung in verteilten Automatisierungssystemen.
R. Oldenbourg Verlag München Wien, 1995.

/5/ Wollschlaeger, M.:
Graphical Programming of P-NET modules.
3rd International conference on the P-NET Fieldbus system, Silkeborg, 25th-26th April 1995,
proceedings.