

PLCs versus Industrial PCs : just a point of view

Nowadays there is a lot of discussion going on about the effect of Industrial PCs as replacements for PLCs. This article wants to share a vision with you.

Basically you can look at a PLC or PC as consisting of three consecutive layers:

1. Application Software
2. System Software
3. Hardware

APPLICATION SOFTWARE

In the discussion of PC versus PLC, the application software basically has the same functionality. The accessibility of local data, often seen as a benefit of open systems, is related to system software level. Of course, the available tools to create the application program can be different. If focussed to the same type of programmers, IEC 1131-3 can make available the same programmers interface for both types of systems.

SYSTEM SOFTWARE

The discussion of changing this level with standard PC products has three options:

1. Windows NT
2. Windows with real-time extensions (fully in software or with hardware extensions)
3. Real-time kernel (with possibly a Windows API to make it capable of running MS Windows software)

Option 2 and 3 only exists because the real-time performance of Windows NT creates many doubts. Since Windows NT is at real-time kernel level much better developed than other Microsoft versions, there are pure software based extensions to Windows NT possible and proven in praxis. The remaining question is on the compatibility with upcoming NT versions. Some people even seriously doubt that Microsoft as supplier is interested in this relatively very small market called industrial automation system software.

On the PLC level, one often sees proprietary kernels. With the occurrence of the IEC 1131-3 standard, time and event driven tasks have entered the PLC line of products. In order to fulfill these requirements one needs a real-time, multitasking kernel included in the PLC. Because this market already existed for quite some time, a change means a complete new family of the PLC hardware, including all kinds of industrialized I/O products. That takes time, leaving the industrial PC a time window in certain markets.

Actually, the IEC 1131-3 standard can be used on all above-mentioned platforms, thus providing the widest support.

PLC? WHICH PLC?

PLC Software versus Software PLC

Industrial controls are used all over the world. However, there are large differences in the interpretation. One clear example concerns PLCs. In Europe a PLC is hardware inclusive the software, meaning also the programming environment. One can speak about PLC tasks, no matter if that will run on a dedicated PLC hardware, or be included within a controller as a separate task. A good example is the integration within a CNC controller.

In the US however, the term PLC seems directly related to hardware. Hardware in the form of industrialized boards that fit into a rack with backpanel, or dedicated like a micro PLC. A software PLC task, sometimes referred to as Software PLC, SoftPLC or Softlogic, seems to be something completely different. But how about the functionality?

Let's make a parallel to chips, ICs. During the 70s a lot of logic was created with discrete chips, all with limited, fixed functionality (e.g. the famous 7400-series). Nowadays, the same functionality can easily go into a programmable device. This means that there is software involved. Software for the off-line development of the program, and 'firmware' to program the device at power up. The question in this parallel is now: should we see this as hardware or software. Again, is there any difference in functionality?

Is it not true nowadays that hardware without software is 'noware' (read: nowhere).

The functionality of PLCs is largely dependent on software. For IEC 1131-3, this software includes the development environment but also the real-time, preemptive, multi tasking kernel as operating system in the target. In case of the Industrial PC as open platform, one can look at it as a strict hardware replacement. The benefit of IEC 1131-3 is clear in this perspective: the same programming method can be used, independent of the target hardware.

In future let us focus on the functionality versus hard- or software PLC-tasks. This will avoid a lot of confusion and a lot of discussion. Perhaps this gives us more time to focus to the real problems!

INDUSTRIAL PC VS. PLC

HARDWARE LEVEL

So the discussion on PLC versus Industrial PC limits itself to the exchange of hardware.

The conditions when this makes sense are widespread, for example:

1. addition of a PLC task to a system
2. relying heavily on distributed I/O
3. integration into math based manufacturing systems

Ad 1.: There are certain levels of automation that combine different tasks within their application. For instance let us look at a Computer Numerical Control system, CNC. Basically it consists of three tasks (with communication as 4th option):

- I. Motion Control
- II. Human Machine Interface
- III. PLC control

On these kind of systems, it makes a lot of sense to combine these three tasks on one processor / one system. From a performance point of view, an industrial PC is not lagging as much as PLC do, so can be an excellent choice (provided it is supported with the right system software, dealing with the correct scheduling / task management). The PC can provide here an open architecture in software and hardware, and price. Interfaces to the motors and I/O can be done via available intelligent sensor-level networks.

Ad 2: PLCs are well known for there industrialized I/O. Although perhaps not cheap, they are well up to their job. Currently there are other architectures possible, i.e. intelligent distributed networks at sensor/actuator level.

Their biggest advantages lie in the reduction of cable costs (one versus many) and the embedded diagnostics / maintenance information. These architectures are ideal for IPCs: they take away the biggest drawback of limited direct I/O extensions. On the other hand, if the application can be divided in sequential programs capable of running on different systems, perhaps coupled with straightforward synchronisation, low cost PLCs can certainly be an alternative.

Ad 3: In case complex models are used to tune the production systems, PCs can provide the necessary direct link between the modeling systems and even use the same model for their control. In this case standard modeling software, like CAE and CAD, can be used. I/O shall be highly dependent on distribution.

CONCLUSION

Windows-based Industrial PCs, and their spin-offs, will certainly have a large impact on the market of large, modular, rack-based PLCs, In that area they can provide highest performance at relatively low cost.

Two other areas will be difficult:

1. The area of fault-tolerant / fail-safe systems. With the requirements on the integrity of especially the hardware and software, standard Windows-based solutions need so man extensions that they probably will not b viable.
2. The area of micro PLCs: with the current price tags on these industrialized control blocks, there is no competition possible from the industrial PC level, and certainly not in combination with Windows (although a lot of these micro PLCs are based on

IEC 1131-3: A STANDARD PROGRAMMING RESOURCE

IEC 1131-3 is the first real endeavor to standardize programming languages for industrial automation. With its worldwide support, it is independent of any single company.

IEC 1131-3 is the third part of the IEC 1131 family. This consists of:

- Part 1:General Overview
- Part 2 Hardware
- Part 3 Programming Languages
- Part 4 User Guidelines
- Part 5 Communication

There are many ways to look at part 3 of this standard. Just to name a few:

- The result of the Task Force 3, Programming Languages, within IEC TC65 SC65B
- The result of hard work by 7 international companies adding tens of years of experience in the field of industrial automation
- Approx. 200 pages of text, with 60-something tables, including features tables
- The specification of the syntax and semantics of a unified suite of programming languages, including the overall software model and a structuring language.

Another elegant view is by splitting the standard in two parts (see figure 1):

1. Common Elements
2. Programming Languages

CONTINUED ON PAGE 34

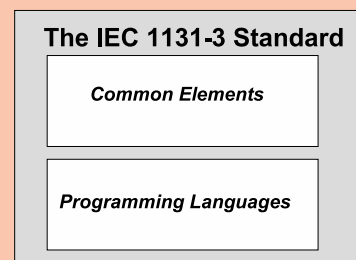


Figure 1. The IEC 1131-3 Standard

**Ad
SysComms**

INDUSTRIAL PC VS. PLC

the Intel line of processors, so potentially be capable of running MS-DOS)

This is an interesting time to live in: the architectures change, the performance of the hardware and software change, new emerging technologies like Java can provide different ways, distributed networks seem to become reality, just to name a view. Looking to the future is always difficult. But looking back certainly will not help this time. Time will tell what technology has in store for us.

Ir. Eelco van der Wal has joined PLCopen as Managing Director in 1995. The growth of the activities of PLCopen as well as its number of members asked for a full-time managing director. The first projects will include further integration of the PLCopen activities in Europe, North-America and Japan, further

development of the promotional activities for IEC 1131-3 and co-ordination of development of higher compliance levels with the standard. Eelco previously held positions in sales, product management, marketing management and business development in the field of real-time industrial automation equipment and VMEbus systems.

COMMON ELEMENTS

Data Typing

Within the common elements, the data types are defined. Data typing prevents errors in an early stage. It is used to define the type of any parameter used. This avoids for instance dividing a Date by an Integer.

Common DataTypes are Boolean, Integer, Real and Byte and Word, but also Date, Time_of_Day and String. Based on these, one can define own personal data types, known as derived data types. In this way one can define an analog input channel as data type, and re-use this over and over again.

Variables

Variables are only assigned to explicit hardware addresses (e.g. input and outputs) in configurations, resources or programs. In this way a high level of hardware independency is created, supporting the reusability of the software.

The scope of the variables is normally limited to the organization unit in which they are declared, e.g. local. This means that their names can be reused in other parts without any conflict, eliminating another source of errors, e.g. the scratchpad. If the variables should have global scope, they have to be declared as such (VAR_GLOBAL). Parameters can be assigned an initial value at start up and cold restart, in order to have

the right setting.

Configuration, Resources and Tasks

To understand these better, let us look at the software model, as defined in the standard (see figure 2).

At the highest level, the entire software required to solve a particular control problem can be formulated as a Configuration. A configuration is specific to a particular type of control system, including the arrangement of the hardware, i.e. processing resources, memory addresses for I/O channels and system capabilities.

Within a configuration one can define one or more Resources. One can look at a resource as a processing facility that is able to execute IEC programs.

Within a resource, one or more Tasks can be defined. Tasks control the execution of a set of programs and/or function blocks. These can either be executed periodically or upon the occurrence of a specified trigger, such as the change of a variable.

Programs are built from a number of different software elements written in any of the IEC defined languages. Typically, a program consists of a network of Functions and Function Blocks, which are able to exchange data. Function and Function Blocks are the basic building blocks, containing a datastructure and an algorithm.

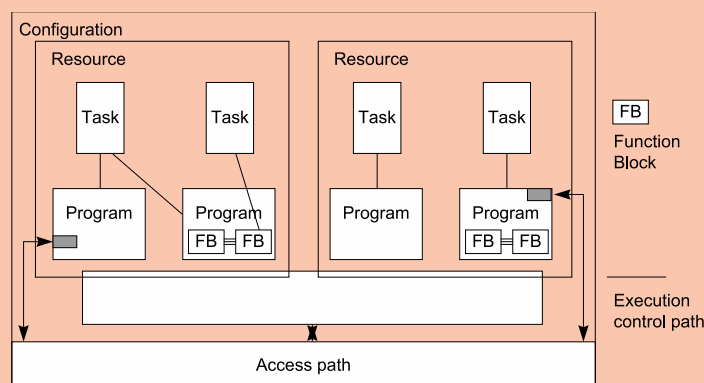


Figure 2. TConfiguration

CONTINUED ON PAGE 35

Let's compare this to a conventional PLC: this contains one resource, running one task, controlling one program, running in a closed loop. IEC 1131-3 adds much to this, making it open to the future. A future that includes multi-processing and event driven programs. And this future is not so far: just look at distributed systems or real-time control systems. IEC 1131-3 is suitable for a broad range of applications, without having to learn additional programming languages.

Program Organization Units

Within IEC 1131-3, the Programs, Function Blocks and Functions are called Program Organization Units, POUs.

Functions

IEC has defined standard functions and user defined functions. Standard functions are for instance ADD(ition), ABS (absolute), SQRT, SINus and COSinus. User defined functions, once defined, can be used over and over again.

Function Blocks, FBs

Function Blocks are the equivalent to Integrated Circuits, ICs, representing a specialized control function. They contain data as well as the algorithm, so they can keep track of the past (which is one of the differences w.r.t. Functions). They have a well-defined interface and hidden internals, like an IC or black box. In this way they give a clear separation between different levels of programmers, or maintenance people.

A temperature control loop, or PID, is an excellent example of a Function Block. Once defined, it can be used over and over again, in the same program, different programs, or even different projects. This makes them highly re-usable.

Function Blocks can be written in any of the IEC languages, and in most cases even in "C". In this way they can be defined by the user. Derived Function Blocks are based on the standard defined FBs, but also completely new, customized FBs are possible within the standard: it just provides the framework.

Programs

With the above-mentioned basic building blocks, one can say that a program is a network of Functions and Function Blocks. A program can be written in any of the defined programming languages.

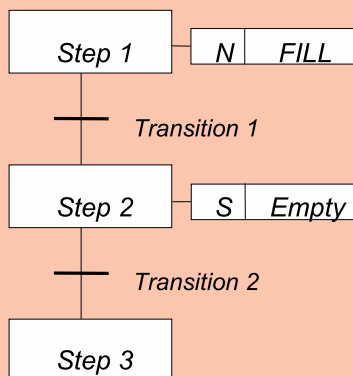


Figure 3. Sequential Function Chart, SFC

Sequential Function Chart, SFC

SFC describes graphically the sequential behaviour of a control program. It is derived from Petri Nets and IEC 848 Grafcet, with the changes necessary to convert the representation from a documentation standard to a set of execution control elements.

SFC structures the internal organization of a program, and helps to decompose a control problem into manageable parts, while maintaining the overview.

SFC consists of Steps, linked with Action Blocks and Transitions. Each step represents a particular state of the systems being controlled. A transition is associated with a condition, which, when true, causes the step before the transition to be deactivated, and the next step to be activated. Steps are linked to action blocks, performing a certain control action. Each element can be programmed in any of the IEC languages, including SFC itself.

One can use alternative sequences and even parallel sequences, such as commonly required in batch applications. For instance, one sequence is used for the primary process, and the second for monitoring the overall operating constraints.

Because of its general structure, SFC provides also a communication tool, combining people of different backgrounds, departments or countries.

PROGRAMMING LANGUAGES

Within the standard four programming languages are defined. This means that their syntax and semantics have been defined, leaving no room for dialects. Once you have learned them, you can use a wide variety of systems based on this standard.

The languages consist of two textual and two graphical versions:

Textual:

- Instruction List, IL
- Structured Text, ST

Graphical:

- Ladder Diagram, LD
- Function Block Diagram, FBD

In figure 4, all four languages describe the same simple program part.

The choice of programming language is dependent on:

- the programmers' background
- the problem at hand
- the level of describing the problem
- the structure of the control system
- the interface to other people / departments

All four languages are interlinked: they provide a common suite, with a link to existing experience. In this way they also provide a communication tool, combining people of different backgrounds.

Ladder Diagram has its roots in the USA. It is based

CONTINUED ON PAGE 36

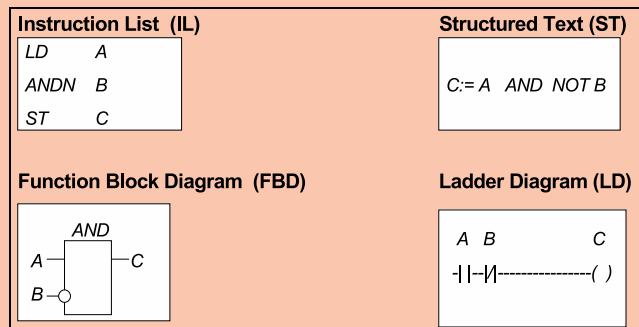


Figure 4.

on the graphical presentation of Relay Ladder Logic. Instruction List is its European counterpart. As textual language, it resembles assembler.

Function Block Diagram is very common to the process industry. It expresses the behaviour of functions, function blocks and programs as a set of interconnected graphical blocks, like in electronic circuit diagrams. It looks at a system in terms of the flow of signals between processing elements.

Structured Text is a very powerful language with its roots in Ada, Pascal and "C". It can be used excellently for the definition of complex function blocks, which

Many current IEC programming environments offer everything you expect from modern environments: mouse operation, pull down menus, graphical programming screens, support for multiple windows, built in hypertext functions, verification during design. Please be aware that this is not specified within the standard itself: it is one of the parts where suppliers can differentiate.

CONCLUSION

The technical implications of the IEC 1131-3 standard are high, leaving enough room for growth and differentiation. This makes this standard suitable to evolve well into the next century.

IEC 1131-3 will have a great impact on the whole industrial control industry. It certainly will not restrict itself to the conventional PLC market. Nowadays, one sees it adopted in the motion control distributed systems and softlogic / PC based control systems, including SCADA packages. And the areas are still growing.

Having a standard over such a broad application area, brings numerous benefits for users / programmers. The benefits for adopting this standard are various, depending on the application areas. Just to name a few for the mindsetting:

- reduced waste of human resources, in training, debugging, maintenance and consultancy
- creating a focus to problem solving via a high level of software reusability
- reduced misunderstanding and errors
- programming techniques usable in a broad environment: general industrial control
- combining different components from different programs, projects, locations, companies and/or countries

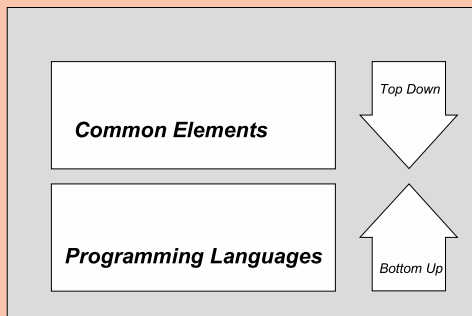


Figure 5. Top-down vs. Bottom-up

can be used within any of the other languages.

TOP-DOWN VS. BOTTOM-UP

Also, the standard allows two ways of developing your program: top down and bottom up. Either you specify your whole application and divide it into sub parts, declare your variables, and so on. Or you start programming your application at the bottom, for instance via derived functions and function blocks. Whichever you choose, the development environment will help you through the whole process.

IMPLEMENTATIONS

The overall requirements of IEC 1131-3 are not easy to fulfill. For that reason, the standard allows partial implementations in various aspects. This covers the number of supported languages, functions and function blocks. This leaves freedom at the supplier side, but a user should be well aware of it during his selection process. Also, a new release can have a dramatically higher level of implementation.