Wolfgang Brendel
infoteam Software GmbH

Component Based IEC 61131-3 Software in Industrial Control

Component Based IEC 61131-3 Software in Industrial Control

Dr. Wolfgang Brendel
infoteam Software GmbH, Germany

Abstract
New directions in the field of industrial control are evolving: large end user companies no longer specify standard off the shelf products for newly designed control systems. Customized tools and products tailored to the specific needs of the application environment set new levels of achievements in cutting the costs of engineering tasks. This change in requirements results in new architectures for component based software, which proves to be more flexible and easier to be adapted to specific needs

1 Requirements for Engineering Tools

A complete methodology covering the lifecycle of control automation includes:

- definition of objects to be controlled by specification of data structures and graphical symbols representing them
- the formal behavioural specification of the objects and the display of visible objects in terms of HMI screens
- implementing the algorithms in terms of textual or graphical programming languages of IEC 61131-3 and describing the user interface with JAVA
- compose the application using already predefined objects in terms of function blocks and HMI-objects
- archive the whole application for later reuse in other projects in an object oriented data base

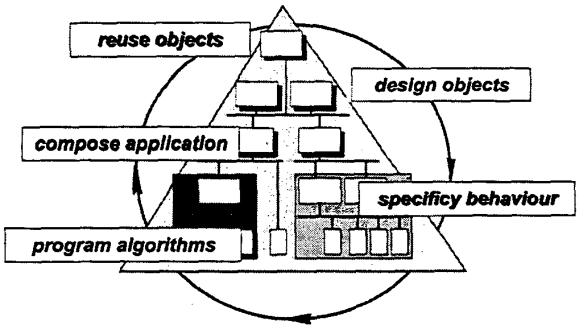


Figure 1: Life cycle in control automation

2 Engineering Tools for Automation Objetcs

Defining an object means decomposing the overall application down to functional units which are characterised by having easy to identify interfaces to other objects and a well understood internal behaviour.
To start with the functional description of an object it is feasible to identify its interface to other objects in several aspects:

- input events that influence the internal behaviour of our object under consideration and output events, which our object emits to influence others

- algorithms that are executed in response to either external events or internal computations, perhaps triggering events to be propagated (function blocks: FB objects)
- data that is needed for the expected performance (input variables) and data that is calculated within the object and propagated to other objects (output variables)
- external representation of the object to the rest of the world especially the operator, the maintenance people or other controller (Operator Interface: OI objects)

Once the process engineer managed the complete cycle of a typical implementation in its domain of applications he built the basic function blocks for similar applications at his hand. Then systems engineering can be seen as the art of combining those building blocks to an engineering solution by several principles involved:

- **selection** of best suitable function blocks for a given set of objects,
- **adding new algorithms** for objects not present in its knowledge base
- **combining** them in a Continuous Function Chart ,
- **system documentation** and maintenance with a graphical engineering tool,
- **enhancing the library** by adding new function blocks for later reuse

This methodology leads to a permanent improvement of the set of available solutions and therefore to a permanent improvement in the overall engineering process.

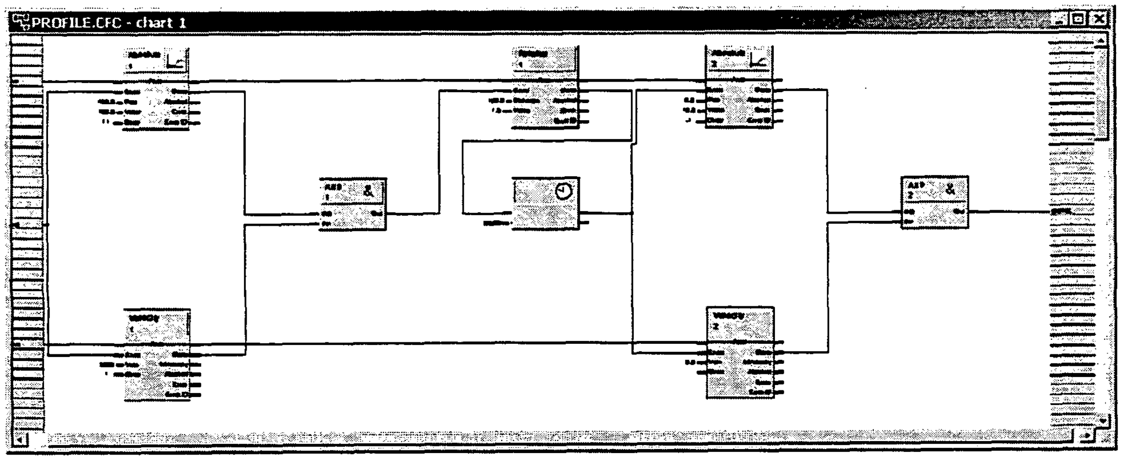


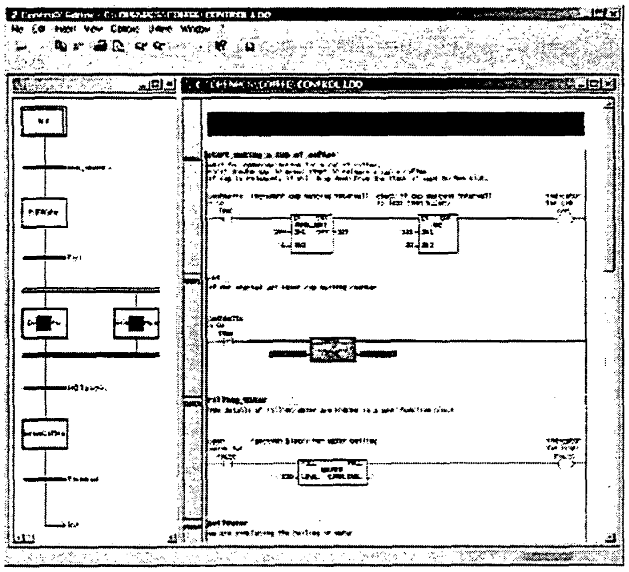
Figure 2: Decomposition into objects using Continuous Function Charts (CFC)

Closing the cycle of control automation design we finally end up with the question how to efficiently reuse already existing designs. The key to success lies in an overall view of what is a design object. It is more than just a function block or a state diagram, it includes also the graphical representation of the part of the machine which it controls, the associated MMI-elements, the names of the interface variables be it events, input/output or parameters and much more.

In the example given above, building a new type of machine by adding a discharge-unit is very similar to select a part from the shelf and combine that to the new machine. By the same way there are function blocks picked and placed and the overall structure of the control algorithm is derived. What remains to be done is the interconnection of the events and variables using the *infoteam CFC* editor.

3 Software Components for IEC 1131-3

The programming requirements of programmable controllers (PLCs) have always been many and various. They are used by people from all kinds of different fields (skilled workers, technical staff and engineers). Programming methods are expected to be application-oriented and geared towards control engineering. At the same time, they have to meet the demands of modern, distributed configurations with increasingly universal, powerful control systems and complex operator systems.



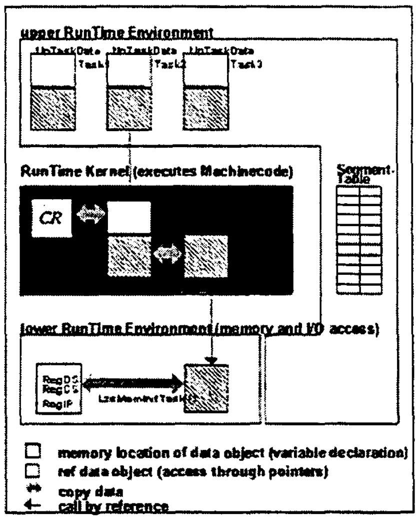
Automation Systems of the future need to be open. Open to the ever changing requirements of your customers and to changes in technology. OpenPCS is designed that way. It consists of a set of tools including editors for all five languages of IEC 61131-3. And not only that. For the special requirements of the North American customer we offer an US Ladder editor, which has the look and feel of the Rockwell Automation tool but has the unique advantage compliant with IEC 61131-3. See the relevant sections of this site to find out more about it.
Despite the plethora of off-the-shelf software products available today for a wide variety of applications, there is still a great demand for custom software development. Catering for the needs of the individual customer is our strong point. This means that OpenPCS is available as:

- Complete IEC 1131-3 programming system
- IEC 1131-3 language extension for your existing programming system
- ActiveX online server with Visual Basic interface for visualisation for all Windows platforms, including Windows CE 3.0.

4 SoftLogic outperforms traditional PLC's

The OpenPCS SoftLogic is based on the runtime model of the SmartPLC. Each implementation which corresponds to this universal machine code model - UCODE for short - is encapsulated by a runtime system, which has the functionality of a SoftLogic and is compliant to IEC 61131-3. The OpenPCS SoftLogic functionality corresponds closely narrowly to the OpenPCS communication protocol and similarities in the interface functions are therefore not accidental, but intentional.
The SmartPLC runtime system is normally part of the firmware of a controller in addition to communication software and operating system. Since there is a wide range of different possible target systems, it was important to implement an easily portable kernel. Platform-specific operations (e.g. memory access routines) were therefore moved into the runtime system. This makes it possible to use the OpenPCS SmartPLC on different controllers and PC based architectures. It comprises functions for task management, a virtual IEC 61131-3 machine code and for the execution of compiler generated code.

Runtime Environment - the yellow shell - and the Runtime Kernel -the red core - can be seen as parallel regular processes. The outer shell encapsulates the core and implements a virtual operating system environment, sheltering the kernel. This makes the kernel totally independent of particular properties of an operating system on the controller. More precisely, it maps the required function for process and memory management to the real-time OS used in this particular implementation. If there are no other tasks beside IEC 61131-3 being performed on the controller, you may use the runtime environment on the bare controller hardware. This is the mode used for a typical PLC.


upper RunTime Environment
UpTaskData Task1
UpTaskData Task2
UpTaskData Task3
RunTime Kernel (executes Machinecode)
CR
Segment-Table
lower RunTime Environment (memory and I/O access)
RegDS
RegCS
RegIP
memory location of data object (variable declaration)
ref data object (access through pointers)
copy data
call by reference

5 Conclusion

Infoteam Software demonstrates that the PLCopen Portability Level specification for IL enables even the portability of graphical languages like Ladder or FBD as well as the code generated from CASE-tools. This gives a far higher importance to PLCopen Portability Level standard than is visible at first. It is the prerequisite for reusing software in control automation.

6 References

[1] Brendel, W.:	*Open Development Kit for IEC 1131-3,* *Technical Notes infoteam Software, March 1995*
[2] Brendel W.:	Principles of CASE Tool design for Automation Control, *IFIP Working Conference „Software Engineering", Chapman & Hall, October 1996*
[3] Feltens, P.:	Continuous Function Chart CFC-Editor, *Benutzerhandbuch, infoteam Software, June 1996*
[4] John K.-H. et.al.:	Portabilty Level for Instruction List, *PLCopen Technical Paper TC3 V0.72 (1997)*
[5] Sperber, M.:	Distributed IEC 1131-3 Programming and CANopen *Industrial Control Programming Conference 96; Oct. 1./2. 1996 Paris*

7 Acknowledgement

The major part of the work reported in this article was done under the framework of the OpenPCS development to which more than 30 people contributed. The results presented partly stem from Michael Sperber, Peter Feltens and Stefan Angele from infoteam Software Bubenreuth and many others not mentioned. Thanks to them all.

Dr. Wolfgang Brendel studied computer science at the „Friedrich Alexander Universität Erlangen - Nürnberg" in the early seventies. Special emphasis was on graph-grammars and incremental compilers. He then joined Siemens and was responsible for the development of the STEP 5 programming software tools until 1981. After rejoining the university he worked on hardware description languages and silicon compilers finishing with the degree of a Ph.D.

In 1983 Dr. Brendel with three colleagues founded „infoteam Software", a company in the software services industry located near Nuremberg, Germany. Nowadays infoteam Software is the leading manufacturer of PLC programming systems with regard to the new standard IEC 1131-3 and is working in a close cooperation with almost all renowned PLC manufacturers. Developing quality software is a main focus as well as commitment in consultation and studies. Dr. Wolfgang Brendel may be reached via mailto:brendel@infoteam.de