Parallelization of engineering processes and reusability of software modules reduce engineering effort.

Efficient engineering with TwinCAT 3

The proportion of engineering costs that is attributable to the control software for machine and plant automation is considerable. As a result, shortening the engineering time for the control software promises enormous savings potential. Through the parallelization of engineering processes, which is supported by TwinCAT 3, as well as the re-usability of already existing software modules, engineering expenditures can be significantly reduced while increasing the quality of the control code at the same time. Knut Güttel and Dr. Josef Papenfort, Product Managers TwinCAT at Beckhoff, give an overview of the possibilities with TwinCAT 3.

In addition, object-oriented technologies universally supported by TwinCAT 3 as well as the provision of additional language resources allow the process knowledge of machine and plant manufacturers to be implemented quickly and efficiently with the aid of modern software technologies. Not only that, embedding in the framework of Microsoft Visual Studio® makes a development infrastructure available that has long been regarded as state of the art in the IT world, but which is currently unique in the field of automation.

	1999	2011
CPU	1 x Intel® Pentium® 4	1 x Intel® Core™2 Duo
Number of units	30	> 80
Number of I/O points	400	2000
Number of servo axes	20	> 100
Velocity	30 m/min	> 80 m/min

Table 1

Regarding productivity, the demands placed on machines have grown tremendously in recent years. As a consequence of this, machines are becoming increasingly complex. Whereas, for example, a woodworking machine in 1999 consisted of 30 units with 400 I/O points and produced 25 to 30 parts per minute, a newer type machine already has 80 units, over 2,000 I/O points and produces more than 80 parts per minute (see Table 1). In order to do justice to this reality, demands have increased not only where the control hardware is concerned, but also in relation to the control software. PC-based control technology imposes no limits here with regard to computer performance, but new solutions are also required in the field of control software. The primary requirements that need to be considered with new software solutions include the following:

– The performance of the control software must be increased.

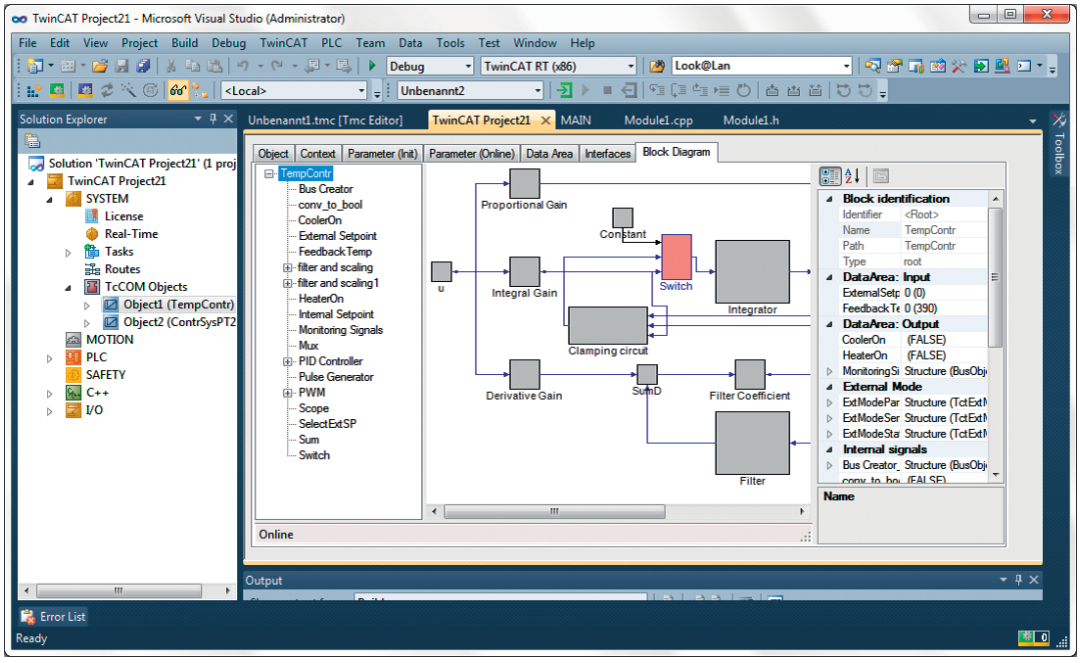
- The implementation of complex control concepts must be possible with the most efficient language resources available.
- Support of object orientation, resulting in re-usable software modules that are simple to maintain
- The software modules must be freely scalable: from a controller through to a complete run-time system.
- The universal use of interfaces in all description languages guaranties the interoperability of the software modules.
- The software modules must be portable to other platforms with little or no effort.
- The software concepts must support current IT technologies such as multi-core CPUs.

In order to answer to these demands as well as to the fact that less and less time is available for planning machines and plants, simple and universal solutions are required in the engineering field. Important factors for achieving this are:

(1) Support of current IT technologies (multi-core, 64-bit operating systems)
(2) Support of various language resources (e.g. IEC 611631-3, C++, Matlab®/ Simulink®)
(3) Provision of a uniform framework for all available language resources
(4) Universal support of object-oriented technologies for the generation of re-usable modules that are simple to extend and maintain
(5) Provision of integrated simulation options that permit switchover to the normal operating mode of the plant without additional expenditure (e.g. re-compilation of modules in the case of address changes)
(6) Provision of universal and efficient debugging functions that facilitate the efficient debugging of the generated software modules without additional tools
(7) Provision of simple options that allow the automatic generation of plant configurations or of control code
(8) Provision of the complete development infrastructure (e.g. tie-in to source code management systems) in the same control environment

Implementation with TwinCAT 3

In order to achieve significant performance increases of the control software, current IT technologies must be supported as stated in point 1. TwinCAT 3 therefore enables the active utilization of resources on multi-core and 64-bit architectures. This means, for example, that software modules created in TwinCAT 3 can be freely distributed to the cores of a multi-core system. A dedicated base time can be defined for each core, independent of the other cores. Based on this, the 65,000 tasks available to the system can be assigned to the individual cores.



Marking of an error in the block diagram

The support of various language resources as demanded in point 2 is intended to solve tasks with the language that is most efficient for the respective problem. For this reason the C/C++ languages as well as Matlab®/Simulink® are available in TwinCAT 3 in addition to the popular IEC 61131-3 languages for the flexible creation of software modules. In practice this means that it is possible to describe a process with the aid of IEC 61131-3, to describe a complex algorithm using C++ and to execute a controller created on the basis of Maltab®/Simulink® in one and the same software project. These modules are executed in the same run-time in TwinCAT 3. In addition, the use of a universal framework as demanded in (point 3) enables the optimum interaction of the modules with each other; i.e. they can exchange data, mutually call one another, address interfaces of the other modules, etc.

As stated in point 4, universal object-oriented technologies are used in TwinCAT 3 for the creation of these modules. The aim is to encapsulate objects in such a way that expandable modules are created that are re-usable and simple to maintain. Among other things this enables the multiple instancing and linking of a controller generated by Matlab®/Simulink® in the same software project (without having to compile it several times). New object-oriented language sources such as interfaces, inheritance or methods are also available in the IEC 61131-3 languages. On top of that, the IEC 61131-3 languages now also permit the dynamic generation of objects at run-time.

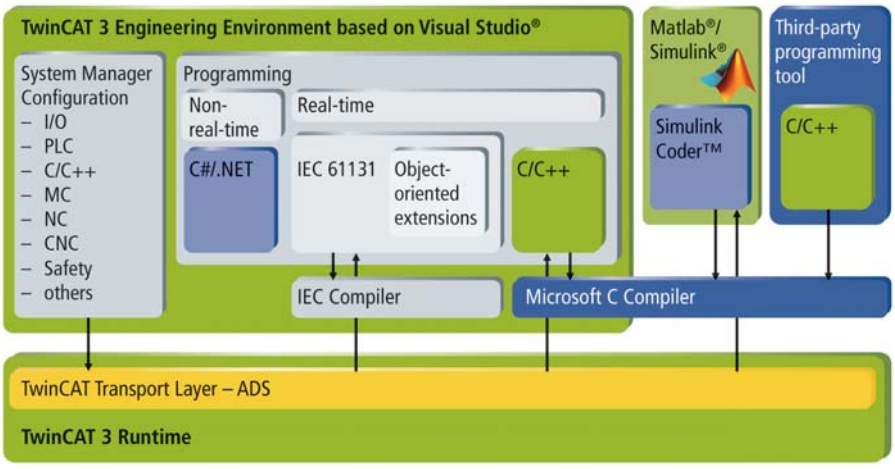
For efficient development or for the testing and commissioning of software modules, it is necessary, as stated in point 5, to embed various simulation options. This is also made possible in TwinCAT 3 thanks to the modular concept. In addition to the software modules which contain the actual control logic, further modules can be integrated into the project in order to simulate entire plant areas. These modules can be switched on or off as required, so that a partial simulation of the machine or plant is also possible. On top of that, linking to the Matlab®/Simulink® tool landscape also offers the possibility to support development methods such as Rapid Control Prototyping. The individual software modules are developed in several steps here:

- Generation of the hardware model to be controlled
- Generation of the software module
- Test of the module in simulation mode (HIL – hardware in the loop)
- Commissioning of the module
- Commissioning of the complete system

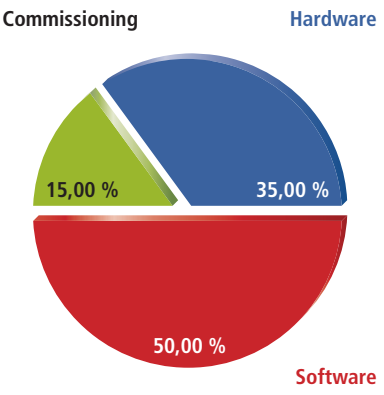
In parallel to the modeling and generation of the simulation module for the hardware, a simulation module of the control software can also be generated that already contains the process, but not yet the finished algorithms. The aim of this procedure is to test the plant hardware without the controller already being fully available (SIL – software in the loop).

For debugging (point 6) during the development of modules, efficient debugging functions must be available in all supported description languages. Whereas the monitoring of online values is already a standard feature of editors for the IEC 61131-3 languages, this has not been possible with editors for the C++ language until now. Here, too, TwinCAT 3 is the first tool to enable this directly in the development environment. A further standard debugging function is the setting of breakpoints. Whereas the use of breakpoints is already the state of the art in the IEC 61131-3 languages or in C++, this does not apply to the "block diagram" of a module generated by Matlab®/Simulink®. It is a similar case with the localization of errors. Whereas these are typically shown in the form of line numbers in the code for modules generated by Matlab®/Simulink®, TwinCAT 3 permits the graphic illustration of the error in the block diagram. In other words, TwinCAT 3 enables the universal use of debugging functions that provide identical options in all available language resources, such as online monitoring, breakpoint, call stack, disassembly window, etc.

A further possibility to significantly increase the efficiency of engineering is the use, as mentioned in point 7, of functions that automatically generate the



The complete integration of TwinCAT 3 into Visual Studio® provides the user with the ability to use the IEC 61131 and C/C++ programming languages in parallel. The generated objects can exchange data with each other and mutually call each other independently of the language they were written in.



Proportional values of automation 2005;
Source: Heinze, R., openautomation

configuration of the control hardware or parts of the control program. Here once again, TwinCAT 3 offers the ideal platform for this: this is achieved, on the one hand, by the support of standard formats, e.g. the use of the PLCopen XML format as the data exchange format for PLC projects, and by the complete control configuration by means of XML-based data formats. On the other hand, a Beckhoff Automation Interface (API) is available with which the complete functionality of TwinCAT 3 can be accessed.

As stated in point 8, the provision of a universal development infrastructure is an important factor in making the engineering of machines and plants more efficient. This includes among other things the tying-in of the development environment to source code management, bug tracking and project management tools. This development infrastructure is optimally provided through the embedding of TwinCAT 3 in the Visual Studio 2010® framework as well as by the support of its native interfaces.

Benefits for the machine manufacturer

In order to master the complexity of modern machines and at the same time cater to demands for ever increasing flexibility and ever shorter planning times, the trend in machines and plants is moving toward modular, re-usable software structures. TwinCAT 3 enables the modular structures of the machine to be mapped directly in the software, independent of the selected description language. This includes the mapping both of the machine module, in the form of encapsulated software objects (classes), and of the object interfaces, with the aid of interfaces.

A substantial benefit of this universal object-oriented approach is the increase in reusability. Individual software modules can simply be reused both within the same software project – e.g. in the multiple instancing of a controller generated by Matlab®/Simulink® – and for new projects. In addition, object-oriented programming leads to improved legibility of the control code, which is associated with simple expandability and an increase in the quality of the control code.

Provided that corresponding interfaces are defined, parallelization of the software engineering is possible on the basis of this encapsulation. This is additionally supported by tying-in to modern development infrastructures such as source code administration, bug tracking, etc. Support for automatic functions for the generation of controller configurations and program parts as mentioned in point 8 also leads to a significant increase in efficiency.

Further Information:
www.beckhoff.com/TwinCAT3