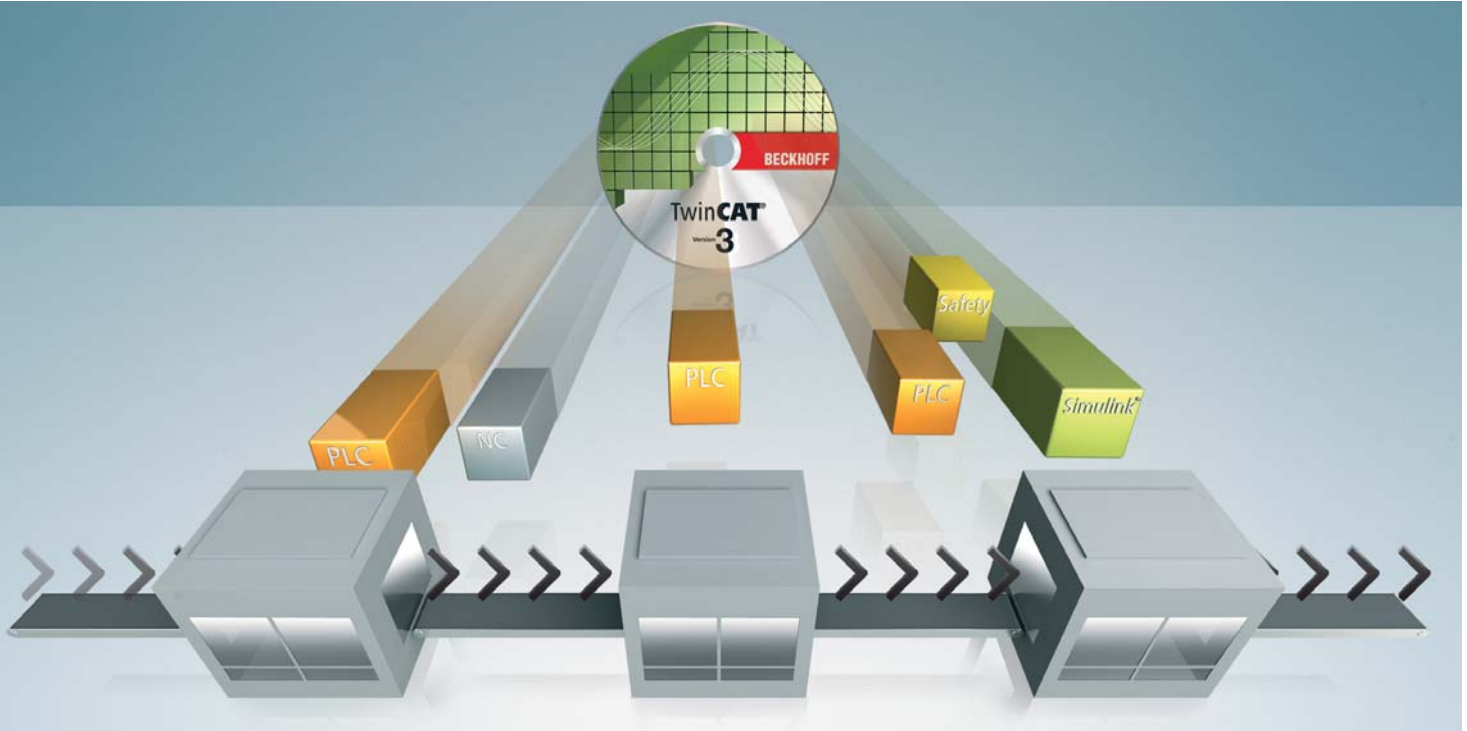


Modular automation with TwinCAT 3



Modern machines are composed of individual, hierarchically arranged modules or units. To a large extent, these units can be developed and tested independently of one another. They can then interact with other modules on the machine via standardized interfaces. A similar modularity is also necessary in the control software, so that each unit has its own 'controller,' which in turn communicates with the other controllers via the standardized interfaces.

Due to the modular structure of the extended runtime and the interfaces already predefined for many applications, TwinCAT 3 from Beckhoff offers the possibility to instance the different controllers among the machine units together on the machine's central control hardware. These can be created independently of one another and in different programming languages. A large selection of already existing or self-developed basic modules thereby forms an automation kit, from which new applications can be easily created.

History

The approach of implementing traditional automation devices such as programmable logic controllers (PLCs) and motion controllers in software on powerful standardized hardware has been state of the art for many years and is an approach now pursued by many suppliers. There are many different advantages, but the most important advantage is surely that the software is independent of the hardware to the largest extent, allowing the performance of the hardware to be adapted to suit the application. In addition, the application automatically benefits from the further development of the hardware. This applies in particular to PC hardware, where increases in performance still continue at a dramatic rate. The relative independence from a supplier, which is also the result of this separation of software and hardware, is also important to the user.

Since the PLC and the motion controller – and possibly other automation components – continue to exist as independent, logical units with this approach, there are very few changes in the application architecture in comparison with traditional automation technology. The PLC determines the logical operational sequence of the machine and assigns the motion controller to implement certain axis functions. Due to the increased performance of the controllers and the possibility to use higher-level programming languages (IEC 61131-3), complex machines can also be automated in this way.

Modularization

In order to master the complexity of modern machines and at the same time to reduce the necessary engineering expenditures, many machine manufacturers have begun to modularize their machines. Individual functions, assemblies or machine units are thereby regarded as modules, which are as independent as possible and are embedded into the overall system via uniform interfaces. Ideally a machine is then structured hierarchically, whereby the lowest modules represent the simplest, continually reusable basic elements. Joined together they form increasingly complex machine units, up to the highest level where the entire machine is created.

Different approaches are followed in the implementation of machine modularization from the point of view of control. They can be roughly subdivided into a local approach and a rather more central approach. In the local approach, each machine module is given its own controller, which determines the PLC functions and possibly also the motion functions of the module. The individual modules can be put into operation and maintained separately from one another and scaled relatively independently. The necessary interactions between the controllers are coordinated via communication networks (fieldbuses or Ethernet) and standardized via appropriate profiles.

The central approach concentrates the control functions of all modules in the common controller and uses very little pre-processing intelligence in the local I/O devices. The interactions can take place much more directly within the central controller, as a result of which the communication routes become considerably shorter, dead times are eliminated and the common use of the control hardware lowers the total cost.

However, the central method also has the disadvantage that the necessary modularization of the control software is not automatically specified. At the same time, the possibility of accessing any information from other parts of the program in the central controller obstructs the module formation and the reusability of this control software in other applications. Since no communication channel exists between the control units, an appropriate profile formation and standardization of the control units frequently fall by the wayside.

The best of both worlds

The ideal controller for modular machines borrows from both the local and the central control architecture. A central, powerful and as-general-as-possible computer platform is 'naturally' used as the control hardware. The advantages of central control technology: lower total costs and the possibility to access all of the information from the overall system without communication losses are crucial arguments.

The advantages of the local approach already outlined above can also be put into practice in the central controller by means of appropriate modularization of the control software. Instead of allowing a large, complex PLC program and an NC with many axes to run, many small 'controllers' can co-exist in a common runtime on the same hardware with relative independence from one another. The individual control modules are encapsulated and offer their functions to the outside via standardized interfaces or use appropriate functions of other modules or the runtime. A meaningful profile formation takes place by the definition of these interfaces and the standardization of the appropriate parameters and process data. Since the individual modules are implemented within one runtime, direct calls to other modules are also possible – in turn via appropriate standardized interfaces. The modularization can therefore take place at meaningful limits, without having to give consideration to communication losses.

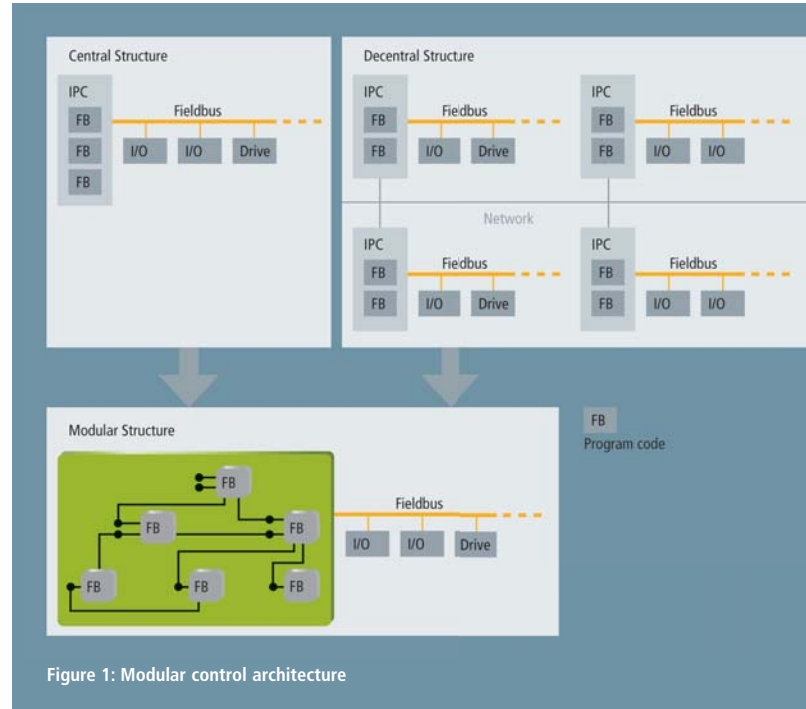


Figure 1: Modular control architecture

During the development or commissioning of individual machine modules, the associated control modules can be created and tested on any control hardware with the appropriate runtime. Missing connections to other modules can be emulated during this phase. On the complete machine they are then instanced together on the central runtime, which only needs to be dimensioned such that the requirements of all instanced modules (memory, tasks and computing power) are fulfilled.

TwinCAT 3 modules

A TwinCAT module consists of a series of formally defined properties, which are partly prescribed and partly optional. The properties are formalized to the extent that general use becomes possible, both among themselves and from the outside. Each module includes a module description file (XML format) for the configuration of the modules and their relationships to one another. If a module is instanced in the TwinCAT runtime, then it registers itself with a central system instance, the module manager. This makes it reachable and parameterizable for other modules and also for general tools.

Modules can be compiled independently of one another and therefore also developed, tested and maintained independently of one another. They can be structured very simply and contain, for example, only one small function such as a low-pass filter; however, they can also be internally very complex and contain, for example, the complete controller of a machine unit. There are many different applications for modules; all of the tasks of an automation system can be and are packed into modules. As a result, no distinction is made between whether the modules primarily represent the basic functions of an automation system, such as real-time tasks, fieldbus drivers or a PLC runtime system, or user and application-specific algorithms for the control or regulation of a machine unit.

Properties of the modules

Figure 2 shows a general TwinCAT module with its essential properties. The red blocks define the prescribed properties and the blue blocks define

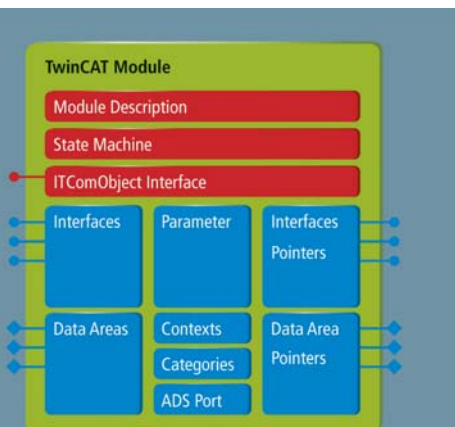


Figure 2: TwinCAT module

the optional properties. In addition to a module description, which is important for the configuration of the modules among themselves, each module possesses a general state machine, via which the initialization, parameterization and linking of the modules are controlled. The prescribed ITComObject interface controls access to the state machine and the module parameters.

The actual function of the module is used via additional interfaces. There is already a large selection of predefined interfaces, which describe general functions – e.g. the cyclic calling of the internal module logic. Additional interfaces, not only standardized but also application-specific, can be added. Access to the interfaces of other modules is possible via interface pointers.

The data areas describe areas of data that are made available by a module to other modules. This allows modules to access common data efficiently via a standardized method. The data of another module is thereby accessed via the data area pointers.

System modules

The TwinCAT runtime also makes a set of so-called system modules available, which provide the basic services of the runtime to other modules. These system modules possess a fixed, constant object ID and can thus be reached by other modules. One example of a system module is the real-time system, which makes the basic services of the real-time system available via its ITcRTIME interface – e.g. the generation of real-time tasks. The ADS router is likewise implemented as a system module, so that other modules can register their ADS port there.

Creation of modules

Modules can be created both in C++ and in IEC 61131-3. The object-oriented extensions of the PLC integrated in TwinCAT 3 are used for this. Modules from both worlds can interact via interfaces in exactly the same way as pure C++ modules with one another. With the aid of the object-oriented extensions in the PLC, the same interfaces are used as in C++. The PLC modules also register themselves with the module manager and are accordingly reachable via it. The complexity of the modules is also variable in the PLC modules; it makes no difference whether only a small filter module is generated or whether a complete PLC program is packed into a module.

In fact, by means of automatism, every PLC program is a module in the sense of TwinCAT 3 modules. Each classic PLC program is automatically packed into a module and registers itself with the module manager and with one or more task modules. Access to the process data of a PLC module (e.g. mapping to a fieldbus driver) takes place likewise via the defined data areas. This behavior remains invisible to the PLC programmer until they decide to explicitly define parts of the PLC program as TwinCAT modules in order to use them with appropriate flexibility.

TwinCAT 3 runtime

The TwinCAT runtime offers a software environment in which TwinCAT modules are loaded, implemented and managed. It offers additional

basic functions so that the system resources can be used (memory, tasks, fieldbus and hardware access etc.). The individual modules do not have to be created using the same compiler and can therefore be independent of one another and can originate from different manufacturers.

A series of system modules is automatically loaded at the start of the runtime, so that their properties are available to other modules. However, access to the properties of the system modules takes place in the same way as access to the properties of normal modules, so that it is unimportant to the modules whether the respective property is made available by a system module or a normal module.

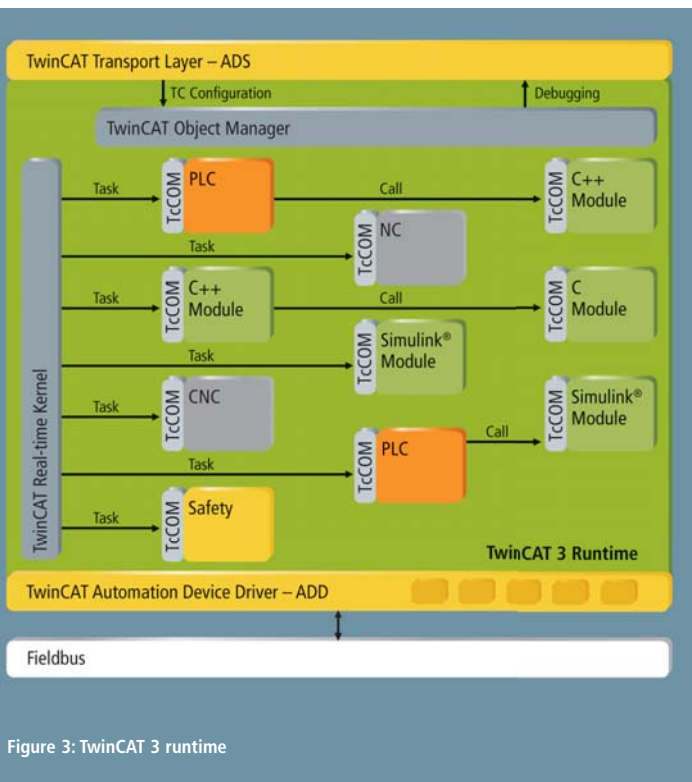


Figure 3: TwinCAT 3 runtime

Expandability

TwinCAT 3 has itself been developed internally in the form of modules. Properties such as fieldbus drivers or motion functions have been created in accordance with the same rules and conventions that also apply to more application-oriented modules. New functions, algorithms and also the support of additional communication protocols can be easily added as a result and can supplement or replace the modules that exist in the standard scope.

The support of proprietary hardware (fieldbus card, frame grabber etc.) can be facilitated with appropriate driver modules and is then equal to, and available in addition to the existing drivers in TwinCAT 3.

www.beckhoff.com/TwinCAT3



Dr. Dirk Janssen, Manager
Software Development System,
CNC and I/O at Beckhoff

Highlights

Comfortable development

New control modules can be developed and tested independently of the specific overall solution. The developer can thereby make use of a large number of existing modules and combine them in a modern development environment based on Microsoft Visual Studio® to form new modules or entire control programs. All of the IEC 61131-3 automation languages as well as C and C++ are available and can be used alongside one another equally. Additional development tools such as Matlab®/Simulink® can likewise be used to generate TwinCAT 3 modules.

Powerful controller

Due to the interaction of the TwinCAT 3 control modules together on one control platform and of defined interfaces with one another, the full performance of an Industrial or Embedded PC can be utilized without communication or synchronization losses. TwinCAT 3 can use all cores of modern CPUs and can also be used on 64-bit systems.

Economical overall solution

The use of pre-developed and tested control modules significantly reduces engineering expenditures for new machines and plants. The control applications can be assembled to a large extent from existing modules; they are adapted to the current task solely by parameterization and interconnection with one another.

The control modules are instanced on a central, general controller, enabling the use of inexpensive hardware as a result. Lower-level, local control hardware can be eliminated.