

.NET and real-time - no contradiction

The runtime environment CLR of .NET technology is not suited for usage in automation engineering due to the missing real-time capability - this is the general view. KW-Software has now found a solution to smooth the way of Microsoft technology into the world of real-time.

The automation engineering provides very many runtime systems. On the one hand there is one specific runtime system for each type of control: for the PLC control, drive control, CNC control, robot control, intelligent sensor/actor, screw control and fieldbus gateways. In addition, this great number is multiplied by the number of manufacturers of one device type as nearly each manufacturer provides his own runtime system.

The great number of runtime systems also involves a great number of required software tools: For each device one specific programming system or configuration tool is available. Today, the engineering in the field of automation still orients itself more to devices and their technologies than to the real application. As a result the expenses for the program development are more and more in disproportion to the total costs of a project.

The cornerstones of a runtime system

Communication and programming interface are the two main cornerstones of a runtime system. Regarding the communication between the individual devices, the manufacturers currently make great endeavors to agree on certain standards: Here, especially Ethernet networks are in the spotlight. In contrast to communication, the programming interface of devices has hardly been brought up yet. Here, the different code formats, which will be loaded onto the automation devices, noticeably restrict the flexibility of distributed systems. It is rarely possible to optimally make the programs consistent with each other according to the functional requirements of the process to be controlled. Therefore it is often not possible to completely use the functionality and performance of the devices. - Thus, a standardized programming interface is the key to effectively manageable distributed systems. Using a standardized programming interface for all controls, from drive control up to the PLC, a new milestone in the world of automation would be set. It is no utopia to realize this: Using the .NET technology of Microsoft together with automation engineering frameworks based on this technology such standardized interfaces can be generated today and therefore allow to make use of the advantages of distributed systems.

The .NET runtime system CLR

Part of .NET is a runtime system, the Common Language Runtime (CLR). That means, the .NET target platforms have a standardized runtime system for different programming languages. This is one of the fundamental features of .NET. Here, the standardized intermediate code, used in conjunction with the terms MSIL (Microsoft Intermediate Language), CIL (Common Intermediate Language, used in ECMA standard) or managed code is of big advantage. MSIL is the common intermediate code base for different programming languages. As a result, the .NET compilers for C++, MS Visual C# or Visual Basic do not

generate processor-dependent object code but MSIL intermediate code. Therefore, is it not obvious to generate MSIL code for IEC-61131 programs?

The language MSIL is completely open, object-oriented, device-independent and allows any form of intermixing, which is then downloaded onto the device. An application program for a control unit could be composed for example of programming parts written in C#, C++ and IEC 61131-3.

.NET also in the field of real-time?

Up to now applying the .NET technology was restricted to the engineering and to non-real-time-critical processes. This is based on the background that the .NET runtime system CLR is not considered as real-time capable.

KW-Software has dealt with the question whether the CLR runtime system can also be used for control tasks.

One of the aspects to be checked was the performance. Here, it turned out that Microsoft CLR provides a high performance for program execution. This is realized by a Just-in-Time (JIT) compiler which converts the MSIL code to highly optimized, directly executable code for the particular CPU. The execution velocity is nearly the same than for the present compilers which directly generate machine code. As, for instance, a Microsoft CLR used in conjunction with a Pentium III 1 GHz processes 1000 BOOL integrations of an IEC program in 1.2 μ s. This value exactly is within the range of the performance reached with the classical ProConOS runtime system provided by KW-Software. Another important aspect was the real-time-capability: Due to the Garbage Collection it is stated that the CLR is not real-time-capable. The Garbage Collection, a component of the CLR, guarantees by an active memory management that the memory is freed automatically for objects no longer required. Thus, the programmer using the .NET programming languages such as C# does no longer take care about memory freeing. The disadvantage is that, as long as the Garbage Collection is running, the execution of application programs is blocked. Regarding real-time programs this means that the jitter for the maximum allowed delay of the program execution may be too high.

Picture: PDF Seite 3 (Heft Seite 64) oben

The future: MSIL and CLR are available as standard languages - engineering tool and device software are no longer compulsory coupled with each other. This results, for instance, in an integrated programming with Multiprog, Visual Studio .NET and other tools as shown in the figure.

The role of the Garbage Collection

This theoretical restriction does not apply in practice because it is unusual for real-time programs to generate objects during runtime which will no longer be used afterwards. When realizing IEC-61131 programs with MSIL, for instance the operator New is only used when loading and initializing the program and not during runtime. Thus, the execution of IEC-61131 programs on a CLR does not require a Garbage Collection and therefore it does not affect the real-time-capability.

The jitter measured for a Microsoft CLR has shown that it is in the lower ms range. For typical PLC applications with cycle times higher than 5 ms this is completely sufficient.

Thus, the requirements for the use of CLR as runtime system for controls are fulfilled and also the use of a standardized programming interface based on MSIL for PLC, CNC and other control systems is possible. This opens up completely new possibilities to mix programming languages for control applications: Program parts developed using different tools such as Visual Studio and Multiprog can be mixed freely. Additional components developed by KW-Software also allow to remotely operate MS Visual Studio. In this case, the communication with the CLR system is done via a TCP/IP DCOM connection. Here, all debug functions provided by Visual Studio are available. Parallel, nearly simultaneous debugging with the help of other tools such as the PLC programming system Multiprog is also possible.

Due to the performance of the CLR programming interface also the firmware usually stored in the automation devices can be loaded as MSIL program. A proprietary runtime system is no longer required. For instance, a PLC runtime system can be realized with MSIL as an integrated part of the application program. Possibly only specific driver interfaces have to be implemented as unmanaged code. These interfaces can be called by defining library functions in MSIL which then provide PLC typical functionalities.

Picture: PDF Seite 3 (Heft Seite 64) unten

KW-Software has developed components for the CLR environment with which Microsoft Visual Studio for example can also be operated remotely. All debug functions of Visual Studio are available in this environment. (Figures: KW-Software)

CLR for real-time operating systems

Another aspect is that Microsoft CLR is only available today for Windows operating systems. Although Windows XP and CE have gained a quite high market share, the embedded world is nowadays inconceivable without the numerous real-time operating systems of other manufacturers. As for the used processors often no Windows operating system exists, the most controls available today do not depend on Windows operating systems. This leads to the question: How can a CLR be used for embedded systems?

The solution is ProConOS embedded CLR, the .NET runtime system by KW-Software, especially developed for the embedded world. With ProConOS embedded CLR for the real-time operating system VxWorks, for example C# programs created with Visual Studio can be executed on the VxWorks system.

The real-time behavior of the embedded CLR is decisively determined by the interrupt latency and task switch times of the underlying VxWorks and is thus within the μs range.

In principle, the embedded CLR can be adjusted to all operating systems. Thus, the ProConOS embedded CLR can be used parallel to an MS CLR also under Windows operating systems.

Whereby it needs to be clearly stated that the functionality of the ProConOS embedded CLR is particularly tailored to control tasks in automation engineering. Thus, the fact of low storage requirements is taken into account for the product, in order to render the .NET technology for embedded runtime systems usable.

Even online changes can be done. This means that programs and associated data objects can be changed while these are being executed. The ProConOS embedded CLR is able to recognize the measures required for an online change self-sufficiently, based on a changed MSIL program. This way, the embedded CLR can support online changes independent of the

programming tool and the programming language. Thus, online changes are also possible with Visual Studio. For example, in a C# program, the algorithm can be changed or variables can be added, without having to stop the program execution. *hap*

INTERVIEW with Andreas Orzelski

KW-Software firstly introduces an Embedded CLR for the automation engineering at the SPS/IPC/DRIVES. Andreas Orzelski, Managing Director, explains the meaning of the new development.

Mr. Orzelski, using the CLR of .NET as runtime system: Where do you see the limits for this process?

We believe that .NET will also change the software world in the field of automation. Compared to today's solutions .NET will make a much more close connection between the world of automation and the office world. Thus, I cannot see any limits. Technically, Microsoft CLR has its limits regarding online changes and real-time capability. For this purpose we offer a solution with Embedded CLR.

You are working on a CLR version for real-time operating systems. When can this product be bought and how will the possible marketing be done?

This year we will show the new technology Embedded CLR at the SPS/IPC/DRIVES and demonstrate that automation can be realized with .NET. We want to introduce a buyable Embedded CLR version next year at the SPS/IPC/Drives. This new product Embedded CLR will be part of our open .NET platform. The Embedded CLR can be used together with our IEC-61131 programming system MULTIPROG and the open engineering platform Automation Framework (*annotation of the author: refer to Special - "Functional Engineering" in the Computer & Automation magazine*) and also standalone as runtime system for own applications.

What could be the consequences of this development?

The .NET technology provides the basis for distributed automation because CLR and MSIL are available as standard languages for devices. As a result, engineering tool and device software are no longer compulsory coupled to each other, but the world of devices opens up. The automation manufacturers will become very flexible regarding the consistency and functionality of their devices.

Could you imagine a cooperation with other suppliers offering IEC-61131 software regarding this?

We focussed on .NET as it provides an open platform. Through the standard MSIL language, our Embedded CLR is also open for other software. Thus, it is imaginable that other IEC-61131 tools generate code and this code will be processed on the Embedded CLR.

Michael Petig, Development Manager at KW-Software, is responsible for the development of PLC system software and in particular for runtime systems.

FOTO PE