RTX64- The Ideal RTOS Platform for the IoT Era

An overview on how IntervalZero's RTX64 transforms Windows into a Real-Time Operating System (RTOS) and delivers on the promise of Internet of Things (IoT) & Industry 4.0

IntervalZero

Executive Summary

This white paper explores how RTX64 is ideally suited for the future of machine control because it capitalizes on commercially available off-the-shelf industrial PCs to address both the deterministic requirements of machine control and the emerging, strategic requirements of the Internet of Things(IoT). RTX64 is remarkable because it transforms Microsoft Windows into an RTOS and this platform is able to execute the deterministic machine control yet can run IoT and non-deterministic Smart Edge and other applications concurrently. This approach is in contrast to other real-time solutions that require two hardware platforms to accomplish the same task – one proprietary platform to serve as a machine controller and one PC-based platform to address the needs for IoT and the User Experience HMI.

Challenges for Machine Builders In The IoT Era

Advancing Core Technology While Adding IoT

The pressure on machine builders to dramatically expand the functionality of their products is mounting due to emerging Internet of Things (IoT) requirements. The customer's growing expectation is that machine operations data will be used to continually optimize the machine's efficiency via IoT feedback loops. Further, in some industries, that same secure IoT connectivity can even create the opportunity for remote management and maintenance, which customers see as a way of reducing operational costs. In an industrial context, customers of machined builders even expect IoT to even improve the performance of adjacent machines and processes of a work cell or plant.

As a result, machine builders are feeling the unprecedented pressure to deliver systems that not only innovate the deterministic dimensions of their machine to improve the throughput and performance, but now must deliver on the promise of IoT.

In preparing for the future, machine builders are discovering that certain deterministic technologies address hard real-time machine requirements better than others especially when IoT requirements must also be considered. While DSPs, FPGAs, or MCUs once answered the call for determinism in a machine, custom hardware-based systems do not easily address the emerging IoT requirements without adding an industrial PC (IPC). Since IoT requirements are non-deterministic, machine builders that rely on FPGAs and that are embracing IoT usually deploy the new IoT features on an IPC. This means machine builders are faced with integrating and supporting two pieces of hardware – the proprietary controller for the determinism and the IPC for the IoT requirements.

With the right architecture like that of IntervalZero's RTX64 RTOS platform, both the deterministic requirements for hard-real time and the non-deterministic requirements for IoT and cloud connectivity that are required in today's machines can be delivered on a single Industrial PC (IPC). This saves integration efforts and cuts hardware costs by more than 50%.

The Shift From FPGA to Industrial PC

Next-generation industrial, vision, medical and other systems that depend on determinism seek to combine high-end graphics, rich user interfaces, IoT and cloud connectivity with hard real-time performance, prioritization and precision. Today's industrial PCs running 64-bit Windows, complemented by a separate deterministic scheduler on multicore multiprocessors, can deliver that precise real-time performance on software-defined peripherals.

The pace of advancement in silicon integration and performance continues unabated, driven by the demands of ever-richer applications like smart edge IoT or cloud connectivity. That advancement, however, has been taking different directions, especially in the case of the integration of functionality onto single silicon dies. The task for the machine builders is to decide how to turn these revolutionary developments into real products that can provide what customers demand that

now includes high-definition audio and video, machine vision, real-time industrial products such as six-axis motion control, real-time connectivity and a rich user interface. That user interface often must also include the ability to present complex real-time graphical data that is linked to the application in real-time.

Today's PC hardware has that capability, and the optimal way to leverage the hardware is to deliver a software-only solution rather than plug proprietary FPGA boards into the PC. In other words, with processor cores that are powerful enough, there is no need to rely on custom hardware to implement specialized functions; that can now be done with software. Software can be more easily updated and improved than hardware, and it is here where machine builders can implement their real value.

The best way to smooth the task of implementing complex real-time software applications is to start with the right hardware environment. As noted above, hardware integration has been taking different directions. On the one hand, there is a trend to integrate the kinds of devices used by real-time systems onto a single die. These might include a multicore processor, a DSP, an FPGA or an advanced graphics unit. We have recently seen, for example, devices that integrate processor cores and FPGAs, or processor cores with advanced graphics units that are also capable of intense number computation such as DSP.

On the other hand, there is the opportunity to tap into today's multi-core CPUs, whose tremendous power and performance is the result of multiple cores and, to a smaller extent, clock speeds approaching 3 GHz. These standard commercial off-the-shelf industrial PCs (IPCs) provide platforms that, with some additional instructions and a scheduler, can deliver DSP-level processing, performance, prioritization and precision. With today's CPUs, this processing can be done in floating point for more diverse calculations than with the fixed point typically found in a DSP. Such performance changes the focus from trying to optimize the use of every instruction to actually fully exploiting the real power of the multicore IPC.

This trend has already resulted in devices that can outstrip traditional DSP processors. Another major development in this arena is the move to 64-bit architectures that are backward compatible with their 32-bit predecessors, but which offer enormously enhanced performance. This has several advantages, because even a highly integrated chip with different integrated functions with their different instruction sets and protocols throws up obstacles to a unified software environment, which adds both hardware hurdles as well as burdens on the development team to circumvent them.

Solution: RTX64 Executes On An Industrial PC

The implementation of IntervalZero's RTX64 takes the latter path and transforms Windows into a fully functional real-time operating system (RTOS) that runs entirely on x64 multicore hardware. Additionally, in so doing it provides access to 128 Gbytes of nonpaged memory, depending on actual mapped physical RAM size. Overall, Windows' 512 Gbytes of physical memory dwarfs the 4 Gbytes physical memory limitation of 32-bit Windows. This vast amount of available memory opens the door to previously unavailable applications like MRI medical imaging and high-end video editing to name a few.

Above all, RTX64 provides a single commodity hardware environment in the form of multicore x64 devices. This enables a single software environment that can accommodate Windows with its rich user interface, available applications and development environment. And Windows is seamlessly connected to the full-function real-time symmetric multiprocessing (SMP) RTX64 environment that can scale from 1 to 63 cores. Applications compile to a single code base with no need for FPGAs or DSPs to execute logic based on different code that must be separately compiled and linked with the main application. One set of hardware, one operating system environment, one set of tools and one base of code. That translates to one team that can communicate and work together and produce high-performance, scalable applications while dramatically shortening time to market.

RTX64 Deep Dive

RTX64—A Fresh Start into 64-Bit

The new RTX64 was built from the ground up to open the world of 64-bit real-time computing, and it is not a port of the 32-bit product. Professional audio and video, high-end medical devices along with advanced industrial control systems that incorporate machine vision and rich user interfaces, all place demands that can only be met by advanced 64-bit systems that can include the rich user interface possible.

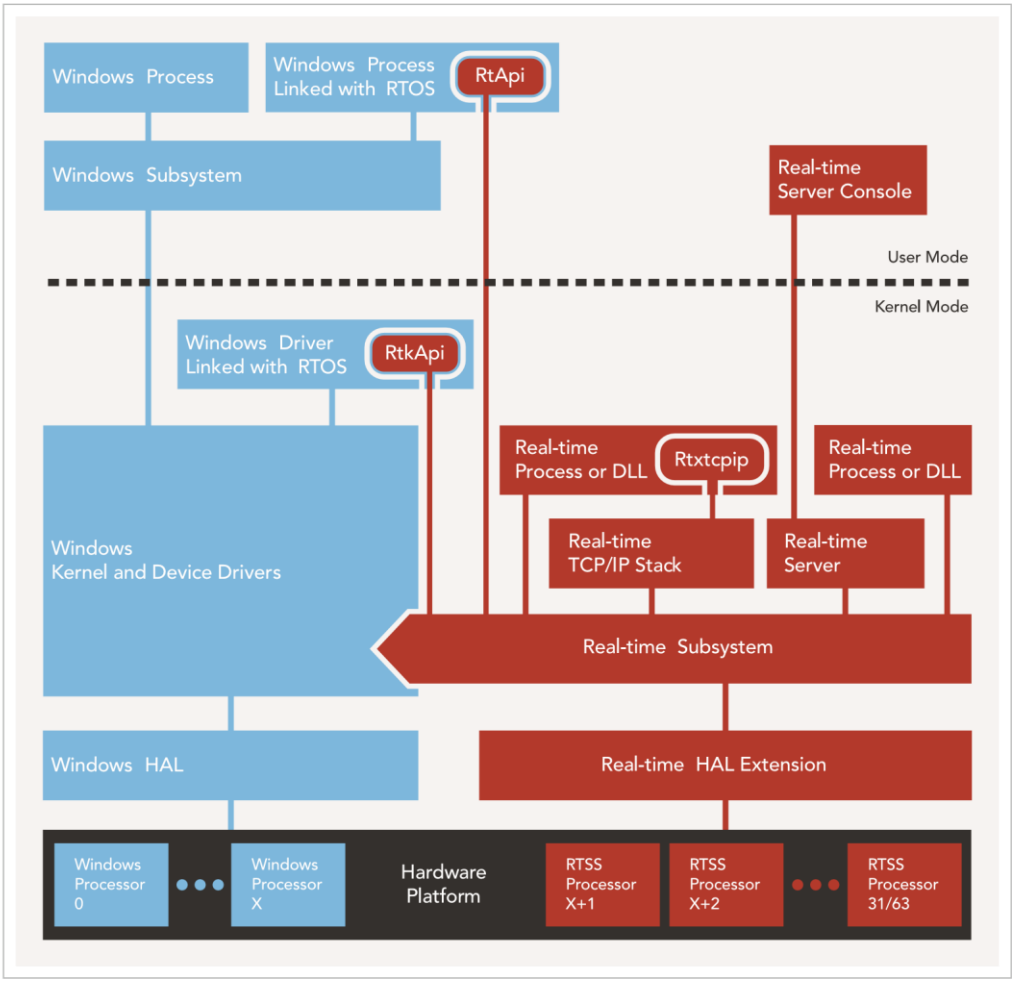


Figure 1

RTX64 provides an architecture that takes advantage of the advancing technologies–specifically, high-speed, multicore x64–that can outperform and outscale the traditional embedded environment that relies on DSPs, FPGAs and microcontrollers.

RTX64 provides an architecture that takes advantage of the advancing technologies—specifically, highspeed, multicore x64—that can outperform and out-scale the traditional embedded environment that relies on DSPs, FPGAs and microcontrollers (Figure 1). It does this by implementing their functions at even higher performance in a single hardware environment, and it can do this in conjunction with Windows, which offers the rich user environment and access to a huge number of applications that can take advantage of and support the real-time operations.

To start with, RTX64 has a hardware abstraction layer (HAL) that is distinct from the Windows HAL and simply operates alongside it. Thus, from the start, no modification of Windows is needed. The two operating systems (GPOS and RTOS) execute side-by-side and communicate via various mechanisms. The RTX64 HAL can scale from 1 to 63 cores to deliver deterministic real-time performance with timing down to 1 µs (dependent on hardware support). The scheduler, which resides in the RTX64 real-time subsystem (RTSS), can assign threads to cores to achieve symmetrical multiprocessing (SMP) without relying on virtualization or complex inter-process communications which adds latency.

This ability to operate independently is a result of the vast memory space that is available to all cores without memory partitioning. Up to 128 Gbytes of non-paged memory and up to 512 Gbytes of physical memory can be accessed by the entire system. This offers a remarkable advantage for medical applications that increasingly depend on visualization such as the Optical Coherence Tomography (OCT) technology now under development, or for real-time surgical robots that depend on accurate rendering and processing of organ images like a beating heart. It is essential for advanced industrial control systems that must not only present visual data to the user, but also process it in real-time to drive motion control of tools and also for the inspection of parts produced by the process.

Having a memory space like this available to such a high-performance general-purpose hardware platform allows OEMs to develop specialized software that can perform extremely specialized functions that would have otherwise required specialized hardware components. Experience has shown that mixing different hardware involves quite different sets of software that depend on different disciplines (e.g., C++ vs. Verilog), which not only greatly slows development time, but also places limits on performance and scalability. Scaling such systems only brings increased complexity with each disparate piece of additional hardware with its own interfaces and unique software needs.

The RTX64 real-time subsystem (RTSS), which includes a real-time scheduler, is fully independent from the Windows kernel and the Windows scheduler. There is no inherent interaction or interference of Windows and real-time threads. Only intended communications between threads by the developer can occur using the real-time API. A real-time API is provided for use with user-mode Windows applications, or a real-time kernel API for use with Windows kernel drivers.

In scenarios such as those enabled by RTX64, applications can present themselves to the user as common Windows applications, while behind the user interface, many of their features are taking advantage of RTX64 real-time processes. For example, a machine tool control program might present a view of the part being machined along with controls and settings that the user can access via a touch screen. However, the actual application consists of two parts. The Windows program can communicate with the real-time control program on two levels—the kernel and the user level—by means of real-time APIs.

At the kernel level, a Windows driver can send data to the RTX64 side, which is perhaps controlling the travel of a tool, and receive current position data, which it then passes to the user interface, or subjects to some sort of processing via a real-time kernel API (RtkApi). At the user level, the operator can set values or the position of switches, etc., on the touch screen and these will communicate with a Windows process. That process in turn uses the real-time API (RtApi) to communicate with the RTSS. These two classes of API communicate directly with the RTSS, which is where the real-time control program resides.

The Important User Experience

As the demand for rich user interfaces for real-time and embedded systems continues to grow, developers are being faced with the dilemma of how to link such interfaces with RTOS environments that are traditionally not designed to support complex user interfaces. With the RTX64 extension to Windows, it is straightforward to use one's favorite graphical tools to design a user interface that can link directly to the underlying real-time application using the RTX APIs. Even more attractive to some could be the ability to simply purchase an off-the-shelf software control and data acquisition (SCADA) tool, which comes with many pre-designed but customizable gauges, sliders, switches and representations of pumps, tanks, actuators, etc., and develop from there using the same RTX64 APIs to hook up to the system.

The same goes for video data. There is a wide selection of tools and applications that can represent physical phenomena, such as heat distribution, fluid dynamics, stress and more, and they all run under Windows. Image processing applications exist that can do edge detection and other operations needed for parts inspection. The list goes on. The OEM offers, at this level of the Windows user interface, a rich selection of "build or buy" options, all of which he can confidently use and/or experiment with knowing that the interface to the underlying real-time application is well defined and will work out of the box.

SMP Offers Performance and Scalability

There are, of course, different schools of thought on how to take advantage of multicore processors. These basically break down into asymmetrical multiprocessing (AMP), or virtualization and symmetrical multiprocessing (SMP). One approach to AMP requires that a copy of the operating system run on each of the cores. This then requires assignment of memory to the individual cores and brings with it the need for inter-process communications that add to overhead and latency – the enemy of determinism. If one tries to implement a user interface with Windows the same inefficiencies apply, requiring inter-process communications between Windows and multiple instantiations of RTOSs and memory partitions. Then try processing (under the RTOS) and displaying (under Windows) real-time video data in such a system—involving more IPC—and things clog up very quickly. Scaling the system to more cores requires more copies of the RTOS, more memory partitioning and reconfiguration of the application.

Another approach to AMP is to implement virtualization with a hypervisor, which is a separate layer of software running directly on the hardware that divides the hardware among the operating systems (Figure 2). Some multicore processors even have built-in hardware assistance for virtualization, which basically presents a virtual "motherboard" to each operating system. Virtualization is often used to support "separation kernels," which are isolated from the rest of the system, communicating only via tightly controlled mechanisms and protocols. This can be useful in certain cases, but its goal is isolation, whereas the goal of SMP is integration. The hypervisor inherently adds some latency which limits its usefulness in a RTOS context. New solution combining real-time hypervisor, SMP and 64 bits is however very interesting. Not only it allows consolidation on a single HW platform, but it still facilitates integration with SMP. This solution will be explored soon in a different white paper.

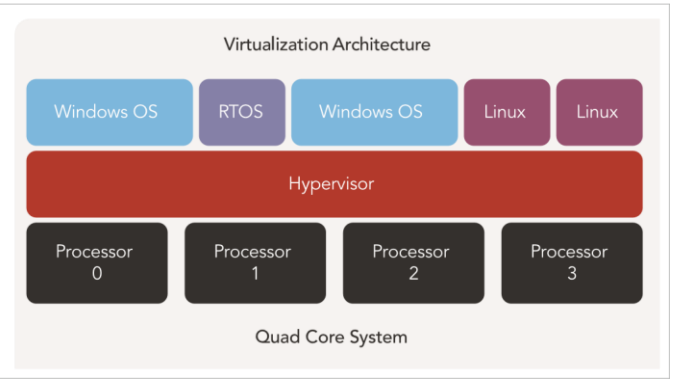


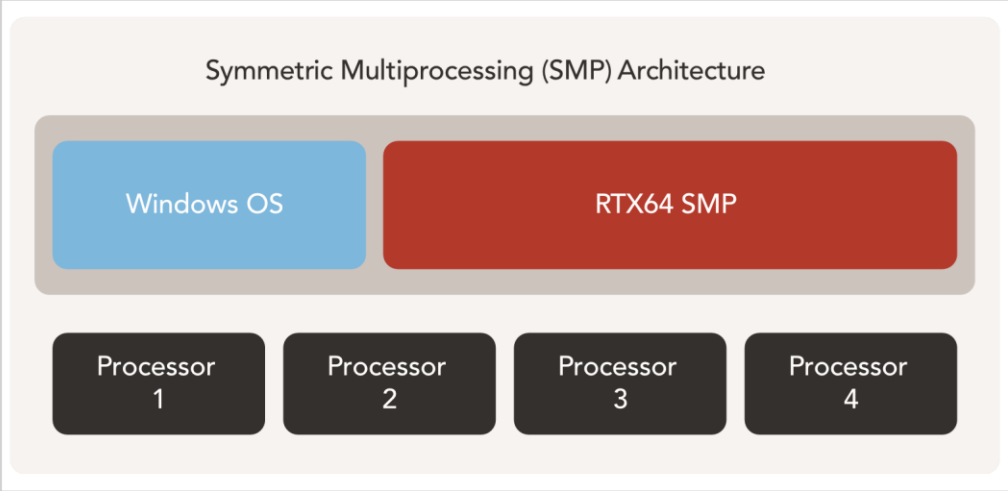
Figure 2

Another approach to AMP is to implement virtualization with a hypervisor, which is a separate layer of software running directly **on the hardware that divides the hardware** among the operating systems.

By contrast to AMP solutions where there is a barrier between the RTOS and core, RTX64 represents a real-time operating system extension to Windows and works with Windows as a single operating system environment that uses the SMP approach to treat the multiprocessor hardware as a single shared resource. It requires only a single copy of the entire operating system environment including the real-time subsystem with its real-time scheduler that has access to all cores assigned to the subsystem (Figure 3). Unlike with AMP, the code can be written once and can be later scaled as functions are added by statically reassigning threads or adding cores and repartitioning. Since all the cores, and hence all the threads, have direct access to shared data and all resources are visible to all real-time processes, there is no need for additional copies or the use of complex inter-process communications schemes or remote procedure calls.

Figure 3

RTX64 represents a real-time operating system extension to Windows and works with Windows as a single operating system environment that uses the SMP approach to treat the multiprocessor hardware as a single shared resource.

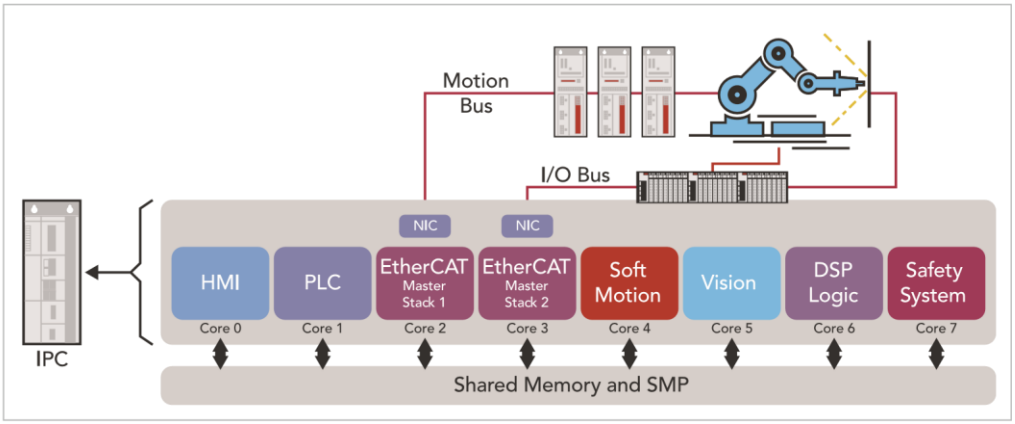


The ability to use a single extended operating system environment across a homogenous hardware platform reduces the machine builders' major hardware decision to, "Do I have enough cores to do what I need to do?" or, "How many more cores do I need to add in order to scale this application to the additional functionality I need?" It no longer involves bridging interfaces between disparate hardware elements like FPGAs and DSPs, or adapting code to parts with increased performance but different programming needs. It no longer involves bringing in different hardware specialists to create or upgrade a product. The team defines the performance in terms of a single programming language like C++.

This leads to the additional advantage of having a single set of development tools, such as Windows Visual Studio, for the entire project. Windows serves as the development environment for the entire system—Windows functions as well as real-time coding. Other Windows-based tools can be brought into the mix as well, such as requirements analysis, version control or static analysis tools to name a few. The user mode of the real-time subsystem also includes an RTX64 server console that connects to the RTSS. The real-time crew can also use their favorite real-time debuggers, profilers and analyzers to tweak the real-time subsystem. They can all communicate and consult with each other in the same terms. Nobody has to learn Verilog or a DSP coding language.

Figure 4

EtherCAT provides for gateways to integrate existing fieldbus components such as CANopen or Profibus. EtherCAT runs under RTX64 in software without the need for any specialized EtherCAT card plugged into the system bus.



Connectivity—Internet and Real Time

In the IoT era, embedded systems no longer stand alone but rather are required to interconnect and share information. In some cases, the network must remain deterministic within a single machine controller as portrayed in Figure 4 above which represents a theoretical robot machine controller. Sometimes the network must be deterministic between controllers as might be the case with collision detection and collision avoidance when multiple robots are working on the same part or loading and unloading a machine. And to support the IoT efforts, sometimes the communication is non-deterministic to the cloud, so that the system of controllers can be optimized. IntervalZero has four different real-time networking solutions that are application dependent. RTX64 supports all three of these communications paths and also delegates the non-deterministic tasks to Windows.

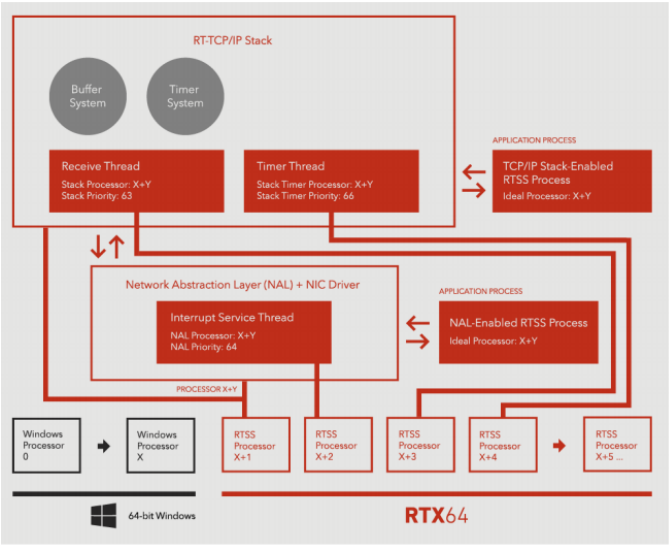


Figure 5 RTX64 Real Time Networking Architecture

Windows IoT-Supporting Role In The RTX64 RTOS Architecture

The Windows Ecosystem Reflected In Machine

Every machine control application is built on an ecosystem of hardware, software and services that turns data into intelligence. No single technology or service provider can supply all the necessary building blocks, so it is imperative that the machine builder chose a development environment that has a strong ecosystem of supporting applications and partners.

Microsoft Windows delivers the most robust ecosystem because it offers machine builders the best-in-class tool chains, security, IoT connectivity, partners, integrators and applications that can support all the machine builders' current and future needs. As described above, RTX64 partitions and reserves one or more cores for its own real-time scheduler. The remaining cores of a multicore PC are allocated to Windows (see the logical allocation in Figure 3 above). The machine builder can capitalize on the strength of the Windows development environment and ecosystem of applications, integrators, and partners to deliver best in class machines with a WOW! HMI, IoT Features and Smart Edge applications.

Breakthrough User Experience with Windows

Machine builders have long relied on Windows for the operator console HMI or other user experience. With RTX64, machine builders can use Visual Studio - a single integrated develop environment – to develop both the HMI and the machine controller. Unlike typical deployments that require hardware for the controller and separate hardware for the HMI, RTX64 consolidates both to a single Industrial PC.

Breakthrough IoT with Windows

With the rise of IoT, secure connectivity from controller to cloud has become a mandatory requirement and the machine builder can easily address this requirement because Windows supports protocols like OPC UA.

Breakthrough Smart Edge with Windows

According to Gartner, digital business initiatives often create data that is more efficiently processed when the computing power is close to the thing or person generating it. Edge computing solutions address this need for localized computing power. IT infrastructure and operations (I&O) leaders tasked with managing these solutions should understand the associated business value and risks.

Around 10% of enterprise-generated data is created and processed outside a traditional centralized data center or cloud. By 2025, Gartner predicts this figure will reach 75%. If machine builders are not considering this trend, then they could be missing out on a market advantage or even a new revenue stream opportunity.

Gartner defines edge computing as solutions that facilitate data processing at or near the source of data generation. For example, in the context of the Internet of Things (IoT), the sources of data generation are usually things with sensors or embedded devices. Edge computing serves as the decentralized extension of the campus networks, cellular networks, data center networks or the cloud.

Organizations that have embarked on a digital business journey have realized that a more decentralized approach is required to address digital business infrastructure requirements. As the volume and velocity of data increases, so too does the inefficiency of streaming all this information to a cloud or data center for processing. In these situations, there are benefits to decentralizing computing power, to placing it closer to the point where data is generated — in other words, to pursuing edge computing. Microsoft Windows and RTX64 is the ideal platform for supporting the vision.

Summary

The RTX64 RTOS Platform is strategic for all machine control in the IoT era because the machine builder can now address all the deterministic requirements and can deliver breakthrough IoT functionality available today in Windows ecosystem - all from a single Industrial PC.