IntervalZero

ProductBrief

March 2016

RTX 2016

IntervalZero's RTX hard real-time software transforms Microsoft Windows into a real-time operating system (RTOS).

RTX is a key component of the RTX RTOS Platform, which streamlines and simplifies the development of complex systems by leveraging x86 multicore multiprocessor technology and symmetric multiprocessing (SMP) functionality to lower costs, boost quality and performance, and reduce reliance on proprietary hardware such DSPs and FPGAs.

Application Determinism

- Guaranteed precision – set timer periods down to 1 microsecond; IST latencies of less the 10 microseconds.
- Separation from Windows – Windows processes cannot interfere with your real-time applications.
- Scalability – one scheduler is used across all real-time processors. SMP-aware scheduler utilizes both priority-driven and pre-emptive algorithms to ensure critical thread context switches. Yields to threads of high priority occur in the sub-microsecond range.

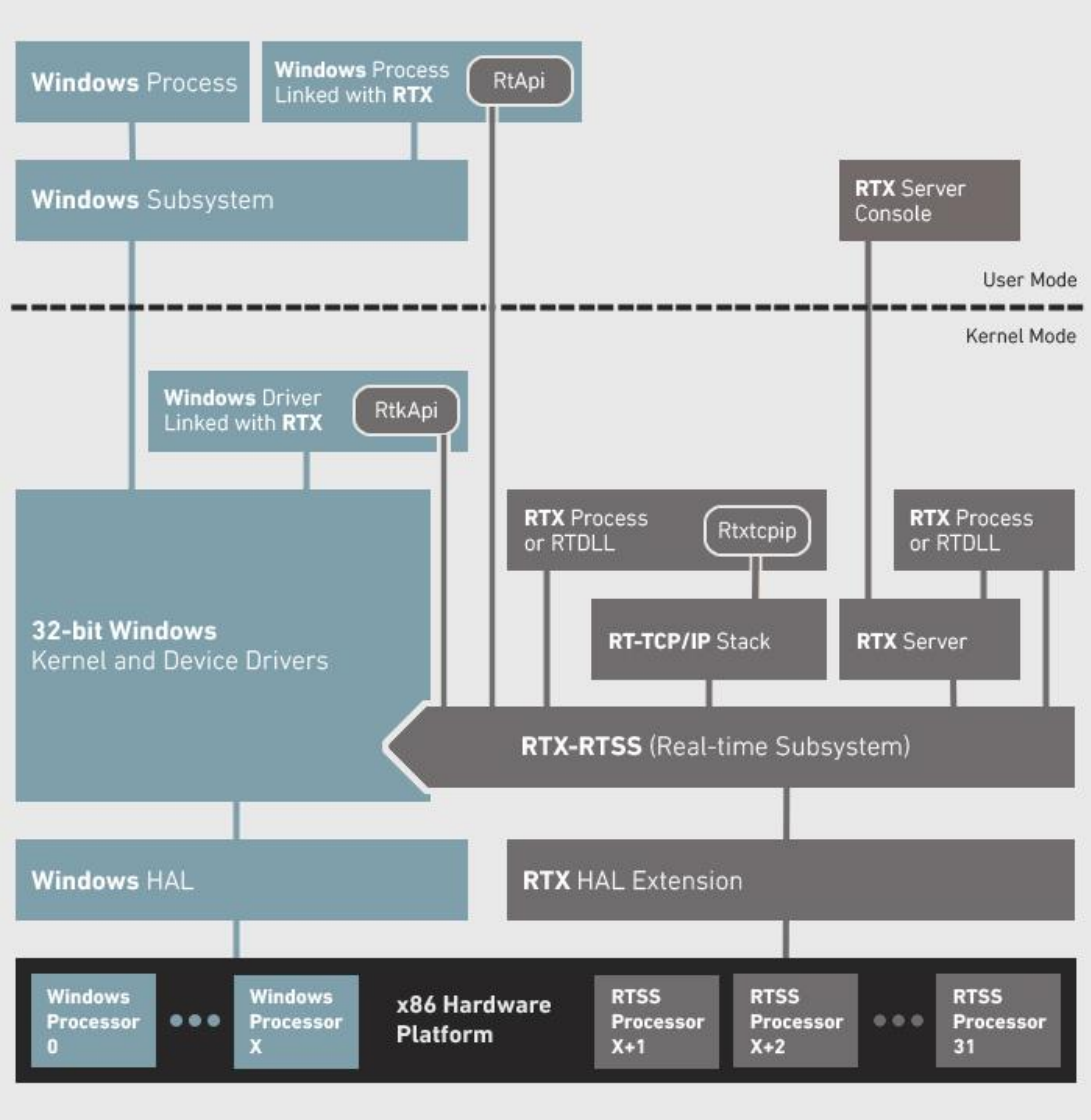
Control of Real-time Applications and Environment

- Flexibility – configure how much or how little of your processing capability is used for real-time processes (1 to 31 processors).
- Full control of our real-time processes – load balancing as needed. RTX provides the ability to set thread and interrupt affinities.
- Peace of mind if Windows issues a STOP message or shutdown – real-time applications have the ability to continue running to a safe shutdown.

Simplify Development and Increase Productivity

- Use a single Operating System – Windows – for applications.
- Use Commercial-off-the-shelf (COTS) target systems; no special hardware required.
- Use one development environment for developing code - Visual studio (2013 and 2015).
- Use common languages (C/C++) for Windows and real-time applications.
- Use common Win32 API – same code can be run as a Windows or real-time process.
- Ability to use managed code for Windows applications and still communicate with real-time applications through a managed code interface.
- No driver model to follow; RTX real-time processes can talk directly to hardware.
- Use standard IPC communication between Windows applications and real-time processes (events, mutexes, and semaphores).
- Use shared memory between Windows and real-time processes for sharing data.

RTX Architecture



Communication

- Network capability - real-time processes can use standard socket API calls to communicate.

Reduce Costs by 25-50%

- Eliminate redundant HMI systems.
- Eliminate proprietary controller and communication cards.
- Improve asset utilization by taking advantage of underused multicore capacity.

Improved Operational Efficiencies and Cost Reduction

- Reduced manufacturing costs through fewer physical parts.
- Eliminate some inventory costs and reduce maintenance costs.
- Field upgrades are done by software download rather than a board replacement.

RTX Editions

The chart below shows the available RTX editions.

The edition …	Includes support for real-time operations on…
Solo	One dedicated RTSS processor in a multicore/multiprocessor environment.
Entry	Up to two dedicated RTSS processors in a multicore/multiprocessor environment.
Basic	Up to three dedicated RTSS processors in a multicore/multiprocessor environment.
Professional	Up to seven dedicated RTSS processors in a multicore/multiprocessor environment.
Premium	Up to 15 dedicated RTSS processors in a multicore/multiprocessor environment.
Ultimate	Up to 31 dedicated RTSS processors in a multicore/multiprocessor environment.

Key Features of RTX

- Real-time subsystem
  - Scalable from 1 to 31 real-time processors
  - SMP-aware scheduler utilizes both priority-driven and pre-emptive algorithms to ensure critical thread context switches; yields to threads of high priority occur in the sub-microsecond range
  - Real-time Win32-like API
  - Configurable thread and interrupt affinity
  - Configurable timer period
  - Ability to attach to line-based and message-based (MSI/MSI-X) interrupts
  - Shutdown handling on Windows STOP or shutdown
  - Deterministic memory
  - Access to Windows file system and registry
  - Dynamic-link library support through RTDLLs
- Real-time inter process communication between Windows user processes and RTX processes
  - Native and managed interface
  - Objects available: events, mutexes, and semaphores
  - Data sharing through shared memory
- Real-time kernel API for communication between a Windows driver and RTX processes

- RT-TCP/IP stack that provides:
    - TCP/UDP/IP networking for RTSS processes
    - Support for IPv4 and IPv6
    - Basic Winsock 2.0 support
    - Utilities (RtssArp, RtssIpConfig, RtssPing, and RtssRoute)
- Windows user groups for limiting access to RTX features (RTXAdministrators and RTXUsers)
- Tools and utilities:
    - Control panel for configuring the RTX subsystem
    - RTSS Task Manager – displays a list of running RTSS processes and registered RTDLLs
    - RTX Server console – displays print messages from RTSS processes
    - RTX Time View – provides the ability to profile real-time applications
    - RTSS Object Viewer – provides the ability to view internal objects and states
    - RTSS Performance View – provides information on CPU usage of RTSS processors
    - SRTM – provides information on timer latency
- Software Development Kit (SDK):
    - Headers and libraries for application development
    - Visual Studio Support (2013 and 2015)
        - Application wizard for development
        - C-Runtime support
        - System-wide debugger that allows for:
            - Launching a process for debugging in Visual Studio
            - Remote debugging of a process through Launch
- Sample source to show basic concepts
- Product documentation
    - Getting Started Guide
    - Real-time features and concepts
    - MiniTutorials and videos
    - API reference
    - Tool usage