RTX 5.1
The Real-Time Environment for Windows®

Runtime Documentation
# Table of Contents

**RTX Runtime Documentation** ........................................................................................................... 1  
**About RTX Runtime** .......................................................................................................................... 1  

**Configuring RTX Runtime** ................................................................................................................ 1  
  - Starting RTX Automatically ................................................................................................................... 1  
  - Starting RTX Manually .......................................................................................................................... 2  
  - Stopping RTX Manually ........................................................................................................................ 2  
  - Configuring RTX on a Multiprocessor System ...................................................................................... 3  
    - Setting up RTX on a Multiprocessor System .................................................................................... 3  
    - Configuring RTX to Use a Dedicated Processor .............................................................................. 4  

**Controlling RTSS Applications, RTSS DLLs, and RTDLLs** ................................................................. 4  
  - About RTSS Applications, RTSS DLLs, and RTDLLs ........................................................................ 4  
    - RTSS Applications ............................................................................................................................ 4  
    - RTSS DLLs ....................................................................................................................................... 4  
    - RTDLLs ............................................................................................................................................. 4  
  - Running an RTSS Application ............................................................................................................. 5  
  - Supporting Boot Time Applications and RTSS DLLs ........................................................................ 6  
    - Register an RTDLL ............................................................................................................................ 7  
    - Unregister an RTDLL .......................................................................................................................... 8  
  - Specifying the Number of RTSS Applications .................................................................................... 8  
  - Terminate an RTSS Process .................................................................................................................. 8  

**Modifying RTX Behavior** ................................................................................................................... 9  
  - Setting the HAL Extension Timer ..................................................................................................... 9  
  - Shutdown Print Handling ..................................................................................................................... 9  
  - Setting the Starvation Timer .............................................................................................................. 10  
  - Freeing the Stack Upon Thread Termination .................................................................................... 10  
  - Handling Process Exceptions ............................................................................................................ 10  
  - Setting the Time Quantum Value ..................................................................................................... 11  
  - Selecting an Alternative Debugger ................................................................................................... 12  

**Managing Devices** ........................................................................................................................... 12  
  - Managing Devices ............................................................................................................................... 12  
  - Modifying RTX Device Properties ..................................................................................................... 13  
  - Viewing Device Properties ................................................................................................................. 15  
  - Converting an RTX Device to a Windows Device ............................................................................. 17  
  - Converting a Windows Device to an RTX Device ............................................................................. 18
About RTX Runtime

RTX (Real-Time Extension) Runtime provides functionality to execute and test real-time applications, RTSS DLLs, and RTDLLs in the RTX environment. RTX Runtime consists of a collection of utilities and tools that allow you to control RTSS processes, measure performance for applications, modify the behavior of RTSS applications, and manage devices controlled by RTX. The RTX environment provides deterministic scheduling of real-time threads and inter-process communication mechanisms between a real-time environment and the native Windows-environment. RTX Runtime can be used on both uniprocessor (UP) and multiprocessor (MP) systems.

Configuring RTX Runtime

Starting RTX Automatically

RTX provides an option that lets you specify how RTX is automatically started.

To specify how RTX is automatically started

1. From the Windows Start menu, click Programs | VenturCom | RTX | RTX Properties to access the RTX Properties control panel.
2. From the RTX Properties control panel, click the Settings tab.
3. From the Startup section, select either Boot or Demand.

When set to Boot, the RTX subsystem is started when the system is booted. This allows RTSS programs to be set to run at boot time (using the RTSSrun utility). If the Startup option is set to Boot and you attempt to start or stop RTX manually using the Start and Stop buttons located on the Control tab, a message indicating that the RTX Startup option must first be switched back to Demand.
When set to **Demand**, the RTX subsystem does not start until an RTSS process is run. If **Demand** is set, the RTX subsystem can be started and stopped at anytime.

4. Click **Apply** to save the setting without closing the control panel or **OK** to save the setting and close the control panel.

**Starting RTX Manually**

The RTX Properties control panel provides a mechanism for manually starting RTX.

**To start RTX:**

1. From the **Windows Start** menu, click **Programs | VenturCom | RTX | RTX Properties** to access the RTX Properties control panel.
2. From the **RTX Properties** control panel, click the **Control** tab.
3. From the **Control** tab, click **Start RTX**. All RTSS processes listed in the Status area change from **Stopped** to **Running**.

**Stopping RTX Manually**

The RTX Properties control panel provides a mechanism for manually stopping RTX.

**To stop RTX:**

1. From the Windows Start menu, click **Programs | VenturCom | RTX | RTX Properties** to access the RTX Properties control panel.
2. From the **RTX Properties** control panel, click the **Control** tab.
3. From the **Control** tab, click **Stop RTX**. All RTSS processes listed in the Status area change from **Running** to **Stopped**.
Configuring RTX on a Multiprocessor System

Setting up RTX on a Multiprocessor System

On a UP system RTX, shares processing time with Windows. On an MP system, RTX can be configured in one of the following ways:

Dedicate a processor to exclusively handle RTX processing. This processor is unknown to Windows.

Share a processor with Windows-related functions when RTX is not using the processor. There are certain considerations that should be taken into account when deciding on how to configure RTX in a multi-processor environment.

- If your real-time application is only using a small amount of processor time, you might want to configure RTX as **shared** so that Windows-related processing can take advantage of the unused portion of the processor.

- If your real-time application is utilizing a large portion of the processor, you probably would want to configure RTX as **dedicated**, so that the application has the room to use it if the need arises.

By default, RTX is configured to share a processor with Windows. To dedicate a processor for handling real-time processing, see *Configuring RTX to Use a Dedicated Processor* on page 4.
Configuring RTX to Use a Dedicated Processor
When RTX is configured as dedicated, the processor will only process RTX-related functions.

To configure RTX to use a dedicated processor, add the following line to Boot.ini:

```
NUMPROC=x,
```

where x is a value less than the total number of processors on your system, Windows will start up with x numbers of processors and RTX will control one of the remaining processors.

Controlling RTSS Applications, RTSS DLLs, and RTDLLs

About RTSS Applications, RTSS DLLs, and RTDLLs

RTSS Applications
RTSS applications are any type of applications that run in a real-time environment.

RTSS DLLs
An RTSS DLL is actually not a DLL but an RTSS process that exports functions for use by other RTSS processes. It shares a common address space with RTSS processes.

RTDLLs
RTDLLs are RTSS objects that can be dynamically loaded and unloaded using the standard Microsoft `LoadLibrary` and `FreeLibrary` calls. They are automatically unloaded from memory when the last RTSS process referencing them terminates. Because RTDLLs do not require linking to an explicit export library, they provide a convenient, flexible runtime accommodation of changes to RTDLLs or applications.
Running an RTSS Application

Use the RTSSrun utility to run an RTSS process in the RTSS environment or load an RTSS DLL.

When you use RTSSrun to run an RTSS process, it scans the RTSS process service slots for a free slot. It copies the RTSS executable image to the target directory for the process slot and establishes a pointer to the path. It then copies the command line to the registry and starts the service for this RTSS process. If successful, the slot number of the new process is returned; otherwise, RTSSrun returns -1.

Each time you use run an RTSS process, the current version of the rtss file is copied to a location in the %SystemRoot% directory. Thus, whenever you rebuild a boot-time RTSS application, you must rerun RTSSrun in order to access the new version.

Format

To run an RTSS application:

From the command line, type rtssrun [/q] filename.rtss

where

/q
This optional parameter disables messages indicating success or failure to start the RTSS process.

filename
Name of the file to be run.

Examples

RTSSrun sample.rtss
    Runs sample.rtss and displays a message indicating whether the process started successfully or failed.

RTSSrun /g sample.rtss
    Runs sample.rtss but does not indicate whether it started successfully or failed to start.
Supporting Boot Time Applications and RTSS DLLs

Use RTSSrun to register RTSS DLLs or RTSS applications so that they run at boot time.

To run an RTSS process:

From the command line, type RTSSrun [/q] /b filename.rtss

where

/q
This optional parameter disables messages indicating success or failure to start the RTSS process.

/b
Runs the RTSS process early in the system boot sequence.

Filename
Name of the file to be run.

Examples

RTSSrun /b sample.rtss
Runs sample.rtss and displays a message indicating whether the application started successfully or failed.

RTSSrun /g /b sample.rtss
Runs sample.rtss but does not indicate whether it started successfully or failed to start.
Register an RTDLL
Use the command line utility, RTSSrun, to register an RTDLL. Once an RTDLL is registered, it can be loaded using the standard Microsoft LoadLibrary function.
Whenever you register an RTDLL, the current version of the .rtdll file is copied to a location in the %SystemRoot% directory. Thus, whenever you rebuild an RTDLL, you must rerun RTSSrun in order to access the new version.

To register an RTDLL:
From the command line, type RTSSrun /d [/s /t] filename.rtdll
where
/d
Loads an RTDLL and adds entries to the registry. This is a required parameter.
/s
This optional parameter allows an RTDLL to be loaded by several processes at the same time. Otherwise, only threads within a single process can simultaneously load a particular RTDLL.
/t
Tests loading and unloading of RTDLLs. If this optional parameter is specified, it must be used in conjunction with the /d parameter.
filename
Name of the file to be run.

Examples
RTSSrun /d sample.rtdll
Registers sample.rtdll
RTSSrun /d /s sample.rtdll
Registers sample.rtdll so that it can be loaded by several processes simultaneously.

See Also
Unregister an RTDLL.
Unregister an RTDLL

Use the RTSSkill utility to unregister an RTDLL that is not loaded. You cannot unregister an RTDLL if any RTSS process currently has it loaded.

To unregister an RTDLL:

From the command line, type `RTSSkill filename.rtdll`

where

filename represents the name of the actual file.

See Also

Register an RTDLL

Specifying the Number of RTSS Applications

You can specify the number of RTSS applications that can be run on your system.

To specify the number of slots:

1. From the Windows Start menu, click Programs | VenturCom | RTX | RTX Properties to access the RTX Properties control panel.
2. From the RTX Properties control panel, click the Settings tab.
3. From RTSS process slots, select the desired number. The minimum number of slots is 10; the maximum is 999.
4. Click Apply to save the setting without closing the control panel or OK to save the setting and close the control panel.

Terminate an RTSS Process

Use the RTSSkill utility to terminate a particular RTSS process.

To terminate an RTSS process:

From the command line, type `RTSSkill /b process number`

where

/b

Undoes the RTSSrun /b action. The program will no longer start automatically at boot time.

Process Number = the particular process number.

Example

To terminate process 001, type

RTSSkill 001
Modifying RTX Behavior

Setting the HAL Extension Timer

Controls the real-time HAL extension timer period.

**To set the HAL extension timer:**

1. From the RTX Properties control panel, click the **Settings** tab.
2. From **HAL timer period**, specify the desired period, expressed in microseconds.
3. Click **Apply** to save the setting without closing the control panel or **OK** to save the setting and close the control panel.

If you set an RTX interval to less than the HAL timer period, RTX will set the interval equal to the HAL timer period. If you specify an interval that is not a multiple of the HAL Timer period, RTX will force the interval to the closest multiple of the HAL Timer period. Two timers with the same interval as the HAL Timer period will necessarily have the same phase.

Shutdown Print Handling

During a system shutdown or stop (blue screen), the kernel switches the display mode from graphics mode to text mode. During this display transition, RTX determinism may be adversely affected. The Shutdown Print Handling feature is used to control the behavior of the display mode switch.

This feature is only available on UP and Shared MP systems using Windows 2000 and Windows XP. On Windows 2000 and Windows XP multiprocessor (MP) systems where RTX uses a dedicated processor, this setting is not available because the blue (stop) screen only occurs on the processor used by Windows, not on the processor dedicated to RTX.

**To utilize the shutdown print handling functions:**

1. From the RTX Properties control panel, select the **Settings** tab.
2. From Shutdown print handling, select either **Always Supported**, **Support On Demand**, or **Not Supported**.
3. Click **Apply** to save the setting without closing the control panel or **OK** to save the setting and close the control panel.

**Always Supported** - this is the traditional approach. If a kernel STOP is occurring, the display is automatically switched to text mode. When the switch takes place RTX determinism is affected.

**Support On Demand** – this mode does not unconditionally switch the display mode. The switch to text mode only occurs if an RTSS thread attempts to print a message to the screen. The first demand for a print initiates the switch over and the adverse latency is endured during this time.

**Not Supported** – this mode disables the switch to text mode. As a result, RTX determinism is maintained but RTSS applications cannot render or print text messages to the screen.

**NOTE:** In the case of a Shared MP system, the shutdown print handling setting only works if the STOP event occurs on the CPU being shared. If the system STOP occurs as the result of a critical execution error on one of the other CPUs, then a blue screen and switch to text mode takes place regardless of this setting.
Setting the Starvation Timer

RTX provides a mechanism for controlling the starvation timer that monitors threads. If enabled, RTX stops the offending thread, issues a popup exception message, and then allows the Windows OS to resume normal operation.

To set the starvation timer:

1. From the RTX Properties control panel, click the Settings tab.
2. From Starvation Time Out, select the desired timeout period (0 - 10000). The timeout period is specified in milliseconds.
3. Click Apply to save the setting without closing the control panel or OK to save the setting and close the control panel.

To disable the starvation timer, click Disable. A checkmark indicates that the timer is disabled.

Freeing the Stack Upon Thread Termination

Controls whether RTSS frees the memory used for an RTSS thread's stack when the thread is terminated. When selected, RTSS frees the memory.

To free an RTSS Thread stack:

1. From the RTX Properties control tab, click the Settings tab.
2. Click Free Stack on Terminate Thread calls. A checkmark indicates that the stack is cleared when a thread is terminated.
3. Click Apply to save the setting without closing the control panel or OK to save the setting and close the control panel.

Handling Process Exceptions

In the event of a process exception, RTX provides a mechanism for controlling the disposition of faulting RTSS processes. Processes can be terminated or frozen.

To select the desired option for handling process exceptions:

1. From the RTX Properties control panel, select the Debug tab.
2. From Process Exception Disposition, select either Terminate the faulting process or Freeze the faulting process. Freezing the process allows you to debug the process data, since the image for a frozen process has not unloaded yet. You can then use RTSSkill to unload it.
Setting the Time Quantum Value

The value a thread will be given for its default time quantum.

To set the time quantum value:

1. From the RTX Properties control panel, click the Settings tab.

2. From Time Quantum, specify the desired value (0 - unlimited). A value of 0 (zero) means run to completion.

3. Click Apply to save the setting without closing the control panel or OK to save the setting and close the control panel.
Selecting an Alternative Debugger

By default, Microsoft's Visual Studio Debugger is the default debug program for debugging RTSS executables. This option allows you to choose an alternative debugging program. When one of the alternative debuggers is selected, only that debugger can be used to debug an RTSS executable.

To select the desired debugging program:

1. From the RTX Properties control panel, select the Debug tab.
2. From Kernel Debugger, click Enable Kernel debugging. A checkmark indicates that kernel debugging is enabled.
3. Select either Microsoft Kernel Debugger or Compuware NuMEAG SoftICE.

To disable alternative kernel debugging, click Enable Kernel debugging to remove the checkmark.

Managing Devices

Managing Devices

The RTX Runtime product provides options for viewing properties associated with devices, editing some of the properties associated with RTX devices, converting Windows devices to RTX devices, and converting RTX devices back to Windows devices.

See Also

Viewing Device Properties
Modifying RTX Device Properties
Converting a Windows Device to an RTX Device
Converting an RTX Device to a Windows Device
Modifying RTX Device Properties

The RTX Properties control panel allows you to select interrupt levels and specify whether several RTX devices with Plug and Play capability can share one interrupt level. This applies only to RTX devices on ACPI systems using Windows 2000 or Windows XP. This procedure is not applicable to Windows NT or non-ACPI Windows 2000/Windows XP systems where the system assigns the IRQ values.

To modify device properties:

1. From the Windows Start menu, click Programs | VenturCom | RTX | RTX Properties to access the RTX Properties control panel.

2. From the RTX Properties control panel, click the Plug and Play tab to display the list of devices associated with the system. If Show filtered list is checked, only the most common devices are displayed.

3. Double-click on the RTX device that you want to modify. The Device Properties dialog box appears displaying information about the device.
In this dialog, you can edit the device’s IRQ value and select its Disposition. The default for all RTX devices is to have separate interrupt levels and an unshareable (Device Exclusive) disposition.

4. Before you edit an IRQ value, check which IRQ values are still available on your machine. You can do that through Device Manager | View | Resources by type | Interrupt request.

5. To share the IRQ value, set the Disposition to Shared. To make the IRQ value unshareable, select Device Exclusive as the Disposition type. Driver Exclusive is usually not used. The Disposition definition is found in the Windows 2000 DDK documentation.

6. Not all PCI slots can be set to be unshareable. If you want your interrupt device to have an unshareable interrupt level (IRQ), you should try to locate an empty, non-shared PCI slot. Some interrupt devices, such as PCI NIC devices, may need a non-shared, bus-mastering PCI slot. It may take several tries to figure out the right slot you desire. A slot that is good for Windows NT may fail with Windows 2000 or Windows XP.

7. Click OK. The RTX Device Properties dialog box closes and you are returned to the RTX Properties control panel.

8. Click OK to close the RTX Device Properties control panel or Apply to keep it opened. The IRQ changes will not take effect until you reboot the machine.

9. To verify that there is no resource conflict with the requested IRQ, open Device Manager, double-click on the device, and check to see if the device is working properly.

See Also

Viewing Device Properties
Converting a Windows Device to an RTX Device
Converting an RTX Device to a Windows Device
Viewing Device Properties

The RTX Properties control panel contains a Plug-and-Play (PnP) tab that displays device properties for both Windows and RTX devices.

To view device properties:

1. From the Windows Start menu, click Programs | VenturCom | RTX | RTX Properties to access the RTX Properties control panel.
2. From the RTX Properties control panel, click the Plug and Play tab to display the list of devices associated with the system. If Show filtered list is checked, only the most common devices are displayed.
3. Double-click on the device that you want to view. The Device Properties dialog box appears displaying information about the device.
See Also

Modifying RTX Device Properties
Converting a Windows Device to an RTX Device
Converting an RTX Device to a Windows Device
Converting an RTX Device to a Windows Device

RTX provides a method for converting an RTX device back to a Windows device.

To give back an existing RTX device to Windows 2000 or Windows XP

1. From the Windows Start menu, click Programs | VenturCom | RTX | RTX Properties to access the RTX Properties control panel.

2. From the RTX Properties control panel, select the Plug and Play tab. The window displays the devices on your system.

3. Right-click on the RTX device you wish to convert, then click Give Back.

4. Click Apply to select another device or click OK to close the RTX Properties control panel.

To uninstall the device:

1. Select the device you are trying to convert to Windows.

2. From the Action menu, click Uninstall.

3. Restart your system.

4. To verify that the device has been installed correctly, open the RTX Properties control panel and check the Devices tab. Your device will be listed under the Windows Hardware list, along with the IRQ and Shareable flag set by default.

See Also

Viewing Device Properties
Modifying RTX Device Properties
Converting a Windows Device to an RTX Device
Uninstalling a Device
Converting a Windows Device to an RTX Device

RTX provides a mechanism for converting an existing device from a Windows 2000 device to an RTX device. Once the device has been associated as an RTX device, it can then be controlled with RTX.

To convert a Windows 2000 or Windows XP device to an RTX device:

1. From the Windows Start menu, click **Programs | VenturCom | RTX | RTX Properties** to access the RTX Properties control panel.
2. From the **RTX Properties** control panel, select the **Plug and Play** tab. The window displays the devices on your system.
3. Right-click on the Windows device you wish to convert, then click **Convert to RTX**.
4. Click **Apply** to select another device or click **OK** to close the RTX Properties control panel.

If the RTX Properties control panel could not remove the specified device, you will receive a warning to uninstall the device and the Device Manager will be started.

To uninstall the device:

1. Select the device you are trying to convert to RTX.
2. From the **Action** menu, click **Uninstall**.
3. Restart your system.
4. To verify that the device has been installed correctly, open the RTX Properties control panel and check the Devices tab. Your device will be listed under the RTX Hardware list, along with the IRQ and Shareable flag set by default.

See Also

Viewing Device Properties
Modifying RTX Device Properties
Converting an RTX Device to a Windows Device
Uninstalling a Device
Uninstalling a Device

Follow the instructions below for installing a Windows or RTX device.

To uninstall the device:

1. From the Windows Device Manager, select either:
   • the device you want to convert to RTX or,
   • the device you want to convert back to Windows.
2. From the Action menu, click Uninstall.
3. Restart your system.
4. To verify that the device has been installed correctly, open the RTX Properties control panel and check the Devices tab. Your device will be listed under the Windows Hardware list or RTX Hardware list, along with the IRQ and Shareable flag set by default.

Tools and Utilities

About Tools and Utilities

RTX Runtime consists of a collection of utilities and a control panel that allows you to control RTSS processes, measure performance for applications, and modify the behavior of RTSS applications. The following are the RTX Runtime utilities:

• SRTM
• KSRTM
• RTSSkill
• RTSSrun
• RTSSview
• RTX Properties Control Panel

SRTM (System Response Time Measurement)

SRTM is an RTAPI timer latency measurement tool that measures timer latency observed by an application. There are two supplied versions: one for a Win32 environment, the other for an RTSS environment. See Measuring Timer Latencies on page 24 for information on using SRTM.

See Also

Using KSRTM
KSRTM (Kernel System Response Time Measurement)
KSRTM is a driver and a Win32 utility that measures HAL-level timer latencies and provides reports and histograms of the results. See Measuring HAL-Level Timer Latencies on page 26 for information on using KSRTM.

See Also
SRTM

RTSSkill
The RTSSkill utility provides several distinct functions. Depending upon the parameter specified, it can be used to:
- Terminate an RTSS process.
- Unregister an RTDLL.
- View RTSS Processes and Registered RTDLLs.

See Also
RTSSkill in Examples

RTSSrun
The RTSSrun utility provides several distinct functions. Depending upon the parameter specified, it can be used to:
- Run an RTSS application.
- Register an RTDLL.
- Register an RTSS application or RTDLL to run at boot time.

RTSSview
RTSSview is a utility that allows you to view information about RTSS processes and their associated objects, such as events, semaphores, and loaded RTDLLs.
RTX Properties Control Panel

The RTX Properties control panel displays configuration information and also lets you set various configurable RTX registry parameters.

To access the RTX Properties control panel:

1. On the Microsoft Windows Start menu, point to Settings, and then click Control Panel.
2. From the Control Panel, double-click the RTX Settings icon to open the RTX Properties control panel.

![RTX Properties Control Panel](image)

The RTX Properties control panel has options for configuring various settings.
Obtaining RTX Information

Display Version and Build Numbers

The RTX Properties control panel contains displays product and version information.

To view the version and build numbers for RTX:

From the RTX Properties control panel, click on the About tab. The RTX Properties control panel appears.

View RTSS Processes and Registered RTDLLs

Use the RTSSkill utility to view a list of all RTSS processes, including registered RTDLLs that have not been loaded, by omitting the process number or filename.

To view all RTSS processes:

From the command line, type RTSSkill to display a list of all RTSS processes including registered RTDLLs.
**View RTSS Process Information**

Use the RTSSview utility to view information about RTSS processes and their associated objects, such as events, semaphores, and loaded RTDLLs.

**To view RTSS process information:**

From the command line, type rtssrun rtssview [/A]

**Where**

/A provides internal system processes and WIN32 proxy processes.

**Results**

RTSSview lists each type of RTSS object, followed by its memory address in the kernel address space and the process ID. It also lists information specific to each particular object:

**Process** — Gives the name or type of process, followed beneath the listing by each thread object owned by that process.

**Thread** — Provides the priority and state of the thread, plus the value of the flag settings which indicate certain thread behaviors. RTX assigns a priority value to each thread in the user's process, with values ranging from 0 to 127. Threads that have negative priority numbers belong to system processes. The thread flags have the following assignments:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Do not deallocate memory</td>
</tr>
<tr>
<td>0x1</td>
<td>Do not free the thread's stacks</td>
</tr>
<tr>
<td>0x4</td>
<td>Thread terminated</td>
</tr>
<tr>
<td>0x8</td>
<td>Thread cannot be resumed</td>
</tr>
<tr>
<td>0x10</td>
<td>Interrupted thread</td>
</tr>
<tr>
<td>0x20</td>
<td>Floating point</td>
</tr>
</tbody>
</table>

**Timer** — Provides the clock number, the remaining time, and the timing interval.

**Semaphore** — Provides the count, maxcount, and the object name (if there is one).

**Mutex** — Lists the count, the address of the owner, and its object name (if there is one).

**Shared memory** — Gives the base memory address, the size of the shared memory, and the object name.

**RTDLL** — Lists the count, address of the owner, and the object name.

**Event** — Provides the state of the event, whether the event is reset manually or not, and the name of the event.

**File** — Provides the Windows handle for the file and the filename.
Displaying RTX Configuration on a Multiprocessor System

The RTX Properties control panel displays multiprocessor configuration information.

To determine RTX's current configuration on a multiprocessor system:

1. From the Windows Start menu, click Programs | VenturCom | RTX | RTX Properties to access the RTX Properties control panel.
2. From the RTX Properties control panel, click the Multiprocessor tab. The status area will display the current RTX configuration. Possible configurations are:
   - RTX running on a dedicated processor
   - RTX sharing a processor with the Windows operating system

The RTX configuration is based upon the boot.ini line selected during boot time. See Setting up RTX on a Multiprocessor System on page 3.

NOTE: The Multiprocessor tab only appears on multiprocessor systems.

Measuring Timer Latencies

KSRTM versus SRTM

Both KSRTM (Kernel System Response Time Measurement) and SRTM (System Response Time Measurement) measure timer latencies. While KSRTM provides detailed information about the source of the worst-case latency, it measures only the time to the beginning of the Interrupt Service Routine (ISR). SRTM measures the total time to an RTSS program's timer handler (i.e., a full thread context); it is the more realistic measurement of latencies encountered by RTSS applications.

The worst-case SRTM reported latencies range from 35 to 100 microseconds on most Pentium-based systems.
Measuring Timer Latencies

RTX Runtime provides, SRTM, a real-time API timer latency measurement tool that measures timer latency observed by an application. Each of the two supplied versions of SRTM, Win32 and RTSS, measures its own environment. An SRTM histogram often shows a characteristic double peak profile: the first peak is a near-best case, the second is the dirty-cache case. The diagram occasionally shows additional "humps" from processor-level interrupt masking.

By default, this tool generates a 15-second tone and prints a histogram of timer response latencies. You can run SRTM in the RTSS environment to measure RTSS latencies or in the Win32 environment to measure Windows NT latencies.

To measure timer latency observed by an application:

Type `RTSSrun "c:\program files\venturcom\rtx\samples\srtm" [-h] [-f] [-s] 15`

Format

`srtm [-h] [-s] [-l] [-f] seconds_to_sample`

where

- `-h` = Display histogram (in addition to summary).
- `-s` = Turn on sound (square wave driven by timer).
- `-l` = Use a 10 ms timer period (default is 1 ms).
- `-f` = Use fastest available timer (1 ms or better).

`seconds_to_sample` = Duration in seconds to sample timer latencies.

You can also run SRTM using Windows NT Explorer by double-clicking the srtm.rtss icon (for RTSS environment) or the srtm.exe icon (for Win32 environment).

See Also

Using KSRTM
Measuring HAL-Level Timer Latencies

KSRTM is a driver and a Win32 utility that measures HAL-level timer latency. Short code paths make it less sensitive to cache jitter than SRTM. It can determine which Windows OS component or device driver is causing the greatest latency event for a real-time application.

KSRTM measures timer response latencies and obtains reports and histograms of the results.

This tool exercises the real-time HAL extension, and measures and reports timer response latencies against a synchronized clock. It can present a histogram of such latencies and report which kernel component was running when the longest latency event occurred. It can also provide a list of drivers.

KSRTM optionally measures the latencies associated with kernel (DPC-level) and multimedia (user-level) timers. It also allows the timer to drive the sound speaker (a 500-microsecond timer will generate a 1000 Hz square wave). Any timing jitter will produce an audible frequency wobble.

Note: KSRTM only works in uniprocessor systems.

Usage

ksrtm -s 2

Format

ksrtm [-r] [-k] [-m] [any of the following flags] seconds_to_sample

where

-r = Real-time HAL extension timer (default).
-k = Kernel (DPC-level) timer.
-m = Multimedia (user-level) timer.
-h = Histogram of latencies (in addition to summary).
-n = No use of real-time priorities.
-s = Sound output (square wave driven by timer).
-i = IRQL HIGH stalls (test interrupt isolation).
-c = Cache flush/dirty and TLB shoot down stalls.
-d = Display loaded driver information.
-l = Longest latency event information.
-u minutes = Minutes between display updates.

seconds_to_sample = Duration in seconds to sample timer response latencies.

See Also

Using SRTM
Examples

RTSSview Example

The following example illustrates information provided by RTSSview for a system running an application called TimerTest, linked with the C run-time library. The unnamed mutexes are created by the C run-time initialization code.

============= RTSSview v4.3 ==============
Object(ID) Address Information
15

Command w32_dll.rtss - Supplemental Win32 for C runtime library
Process(010) 806a3c28
  Thread(00004) 806a3cc8 Pri=0 Flags=1 State=Suspended
Command timertest.rtss Timer_test2
Process(001) 805d6008
  Thread(0000c) 805d60a8 Pri=0 Flags=1 State=Sleeping
  Thread(0000d) 805ef490 Pri=15 Flags=0 State=WaitInternal
  Timer 805f9f10 Clock=1 Remain=7000us Interval=12000us
Command rtssview.rtss
Process(002) 805fad28
  Thread(0000e) 805fadc8 Pri=0 Flags=1 State=Ready

Mutexes:
  805f18f0 Count=0 Owner=0 Name="** UNNAMED **"
  805ef6d0 Count=0 Owner=0 Name="** UNNAMED **"
  805f06d0 Count=0 Owner=0 Name="** UNNAMED **"
  805f0b70 Count=0 Owner=0 Name="** UNNAMED **"
  805f3690 Count=0 Owner=0 Name="** UNNAMED **"
  805f2b70 Count=0 Owner=0 Name="** UNNAMED **"
  805f5b70 Count=0 Owner=0 Name="** UNNAMED **"
  805f56d0 Count=0 Owner=0 Name="** UNNAMED **"
  805d5010 Count=0 Owner=0 Name="** UNNAMED **"
  805d5690 Count=0 Owner=0 Name="** UNNAMED **"
  805beb70 Count=0 Owner=0 Name="** UNNAMED **"
  805bb010 Count=0 Owner=0 Name="** UNNAMED **"

Dynamically Loaded Libraries:
  805fa6b0 Count=-2141245552 Owner=805f2f98 Name="Timer.dll"

============= End RTSSview ==============

See Also

RTSSview Utility
## RTSSkill Examples

### RTSSkill Example 1
The following example illustrates information provided by RTSSkill when run with no arguments for a system running an application called TimerTest that uses a single RTDLL.

<table>
<thead>
<tr>
<th>PID</th>
<th>Start</th>
<th>State</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>R</td>
<td>TimerTest.rtss Timer_test2</td>
<td></td>
</tr>
<tr>
<td>002</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>005</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>/boot R w32_dll.rtss - Supplemental Win32 for C runtime library</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RTSS Registered Dlls:
- Loaded timer.rtdll
- Not Loaded foo.rtdll

### RTSSkill Example 2
The following example illustrates information provided by RTSSkill 001 for the system in the preceding example.

<table>
<thead>
<tr>
<th>PID</th>
<th>Start</th>
<th>State</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>R</td>
<td>TimerTest.rtss Timer_test2 (terminated)</td>
<td></td>
</tr>
<tr>
<td>002</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>005</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>006</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>007</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>/boot R w32_dll.rtss - Supplemental Win32 for C runtime library</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RTSS Registered Dlls:
- Not Loaded timer.rtdll
- Not Loaded foo.rtdll
Viewing RTSS Process Information

Use the RTSSview utility to view information about RTSS processes and their associated objects, such as events, semaphores, and loaded RTDLLs.

To view RTSS process information:

1. Locate the directory where RTSSView is located (typically C:\Program Files\VenturCom\RTX\bin).
2. Click on the RTSSView.RTSS icon. The RtxServer window appears displaying information similar to the following:

RTSSview lists each type of RTSS object, followed by its memory address in the kernel address space and the process ID. It also lists information specific to each particular object:

Process — Gives the name or type of process, followed beneath the listing by each thread object owned by that process.

Thread — Provides the priority and state of the thread, plus the value of the flag settings which indicate certain thread behaviors. RTX assigns a priority value to each thread in the user's process, with values ranging from 0 to 127. Threads that have negative priority numbers belong to system processes. The thread flags have the following assignments:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Do not deallocate memory</td>
</tr>
<tr>
<td>0x1</td>
<td>Do not free the thread's stacks</td>
</tr>
<tr>
<td>0x4</td>
<td>Thread terminated</td>
</tr>
<tr>
<td>0x8</td>
<td>Thread cannot be resumed</td>
</tr>
<tr>
<td>0x10</td>
<td>Interrupted thread</td>
</tr>
<tr>
<td>0x20</td>
<td>Floating point</td>
</tr>
</tbody>
</table>
Timer — Provides the clock number, the remaining time, and the timing interval.
Semaphore — Provides the count, maxcount, and the object name (if there is one).
Mutex — Lists the count, the address of the owner, and its object name (if there is one).
Shared memory — Gives the base memory address, the size of the shared memory, and the object name.
RTDLL — Lists the count, address of the owner, and the object name.
Event — Provides the state of the event, whether the event is reset manually or not, and the name of the event.
File — Provides the Windows NT or Windows 2000 handle for the file and the filename.

See Also
RTSSView example in Examples.
Index

A
About .......................................................................................................................... 1
RTX Runtime .............................................................................................................. 1

B
Blue screen and Stop Messages.................................................................................. 9
Build Numbers .......................................................................................................... 22

C
Control Panel for RTX Properties ............................................................................... 2, 21
Converting Devices to RTX Devices .......................................................................... 18
Converting Devices to Windows Devices .................................................................... 17

D
Debugging .................................................................................................................. 12
Dedicated Processor .................................................................................................. 3
Device Properties ...................................................................................................... 13, 15
Managing Devices .................................................................................................... 12

E
Exception Handling .................................................................................................. 10

H
HAL .......................................................................................................................... 9, 26
Measuring HAL-Level Timer Latencies....................................................................... 26
Setting the HAL Extension Timer ............................................................................. 9

K
KSRTM ...................................................................................................................... 20
KSRTM versus SRTM ............................................................................................... 24
Performance Measurement Tools ............................................................................. 19

L
Latency ..................................................................................................................... 19, 20, 25

M
Modifying Device Properties...................................................................................... 13
Multiprocessor Configuration..................................................................................... 3
Displaying Multiprocessor Configuration Information ................................................. 24
Setting up RTX on a Multiprocessor System ............................................................. 3

P
Performance Measurement Tools............................................................................... 19
KSRTM ....................................................................................................................... 20
SRTM ......................................................................................................................... 19
Plug and Play Devices .............................................................................................. 15
<table>
<thead>
<tr>
<th>Section</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Exceptions</td>
<td>10</td>
</tr>
<tr>
<td>Process Slots</td>
<td>8</td>
</tr>
<tr>
<td>RTDLL Registration</td>
<td>6, 7, 8</td>
</tr>
<tr>
<td>RTSS Process Information</td>
<td>29</td>
</tr>
<tr>
<td>RTSS Process Slots</td>
<td>8</td>
</tr>
<tr>
<td>RTSSkill Utility</td>
<td>8, 20</td>
</tr>
<tr>
<td>Examples</td>
<td>28</td>
</tr>
<tr>
<td>RTSSrun Utility</td>
<td>5, 20</td>
</tr>
<tr>
<td>RTSSView Utility</td>
<td>20, 22, 23</td>
</tr>
<tr>
<td>Example</td>
<td>27</td>
</tr>
<tr>
<td>RTX Properties Control Panel</td>
<td>2, 21</td>
</tr>
<tr>
<td>Runtime</td>
<td>1</td>
</tr>
<tr>
<td>Shared Processor</td>
<td>3</td>
</tr>
<tr>
<td>Shutdown Print Handling</td>
<td>9</td>
</tr>
<tr>
<td>SRTM</td>
<td>19</td>
</tr>
<tr>
<td>KSRTM versus SRTM</td>
<td>24</td>
</tr>
<tr>
<td>Stack</td>
<td>10</td>
</tr>
<tr>
<td>Freeing Upon Thread Termination</td>
<td>10</td>
</tr>
<tr>
<td>Starting RTX Automatically</td>
<td>1</td>
</tr>
<tr>
<td>Starvation Timer</td>
<td>10</td>
</tr>
<tr>
<td>Stop Messages</td>
<td>9</td>
</tr>
<tr>
<td>Thread Termination</td>
<td>10</td>
</tr>
<tr>
<td>Freeing the Stack</td>
<td>10</td>
</tr>
<tr>
<td>Time Quantum</td>
<td>11</td>
</tr>
<tr>
<td>Timer Latencies</td>
<td>25</td>
</tr>
<tr>
<td>Timer Latency</td>
<td>20</td>
</tr>
<tr>
<td>Version Numbers</td>
<td>22</td>
</tr>
<tr>
<td>Viewing</td>
<td>29</td>
</tr>
<tr>
<td>RTSS Process Information</td>
<td>29</td>
</tr>
</tbody>
</table>