RTS Hypervisor

System configuration

Configure with ease

RTS Hypervisor lets you exactly meet your system requirements in no time at all. With the Configuration File, you precisely assign computer resources and operating systems (OS) to different CPU cores and specify your preferences for the runtime environment.

Tailor-made system design

In the Configuration File, you define, amongst others, operating modes of used OSs, memory allocation, shared memory sections, events, virtual networks, timers, and IRQs. It allows you to assign PCI devices, I/O and USB ports, disks and partitions, and a lot more. The RTS Hypervisor lets you allocate permissions to make existing OSs visible or invisible, to use shared memory sections, and to change time synchronization.

Time-saving setup

The Configuration File comes with plain ASCII format, which allows you to easily edit the configuration with any plain-text editor. Contained information is divided into sections. Thanks to clearly assigned keywords you can find, check, and edit a specific parameter with ease.

Your benefits

- **Tailored design:** Clearly allocate computer resources and OSs to single CPU cores and configure the runtime environment as per your needs
- **Easy configuration:** Edit the Configuration File with any plain-text editor or via XML
- **Time-saving editing:** Remain overview thanks to the clearly defined file structure with assigned keywords

»The Hypervisor provides visibility into hardware devices, clear assignment and access handling, and a thorough documentation with helpful examples. This makes the system really easy to use.«

David A. Rauth
Kulicke & Soffa (KNS), Pennsylvania, USA

Quickly configured:
Example setup for one RTOS and one GPOS using half of the CPU cores each.

```
# RTH CONFIG

################################################################################
# System
#

[/SYSTEM]

    "IOMMU"          = uint32: 1    # set to 1 to restrict device memory access
    "security_level" = uint32: 1    # set to 1 to restrict API permissions

    "CPU_resource_partitioning" = uint32: 1      # set to 1 to optimize
                                                 # CPU resource usage

################################################################################
# OS
#

[/OS/0]

    "name"          = "Windows"
    "boot_priority" = uint32: 2             # start after QNX
    "virtualized"   = uint32: 1
    "cores_percent" = uint32: 50            # assign half of the CPUs

    "trace_partition_number" = uint32: 0    # write log to /SHM/0

    [/OS/0/RUNTIME/0]

        "image_0" = "bootmgfw.efi"

[/OS/1]

    "name"          = "QNX Neutrino"
    "boot_priority" = uint32: 1             # start first
    "memory_size"   = uint64: 0x20000000    # 512 MB
    "virtual_MMU"   = uint32: 1             # set to 0 to disable restricted memory access
    "restricted_IO" = uint32: 0             # set to 1 to restrict I/O access
    "cores_percent" = uint32: 50            # assign half of the CPUs


    "trace_partition_number" = uint32: 1    # write log to /SHM/1

    [/OS/1/RUNTIME/0]

        "image_0" = "pcrth_096_x86_64.ifs"


################################################################################
# Interrupts
#

[/IRQ]

    "default" = uint32: 0    # default is Windows
    "IRQ_04"  = uint32: 1    # assign COM1 to QNX


################################################################################
# PCI
#

[/PCI]

    "default" = uint32: 0    # default is Windows

    "auto_interrupt_mode" = uint32: 1    # use MSI automatically
                                         # to avoid IRQ conflicts

# Simple device assignment
[/PCI/0]

    "OS"     = uint32: 1
    "bus"    = uint32: 5
    "device" = uint32: 4

# Assignment of one function in a multi-function device
[/PCI/1]

    "OS"       = uint32: 1
    "bus"      = uint32: 5
    "device"   = uint32: 4
    "function" = bytelist: 1

# Device assignment with vendor ID / device ID
[/PCI/2]

    "OS"        = uint32: 1
    "vendor_ID" = uint32: 0x8086
    "device_ID" = uint32: 0x10D3


################################################################################
# Shared Memory
#

[/SHM/0]

    "name" = "trace_0"
    "size" = uint64: 0x4000

[/SHM/1]

    "name" = "trace_1"
    "size" = uint64: 0x4000
```

Real-Time Systems GmbH is a leading provider of real-time virtualization, hypervisor and operating system technology for embedded and real-time systems. With an extensive knowledge of quality software design, Real-Time Systems GmbH provides a competitive advantage to their customers. The Intel® co-development partner was founded in 2006 and is headquartered in Ravensburg, Germany. Since 2018, Real-Time Systems GmbH is a company of congatec GmbH with partners in Europe, USA, and Asia.

www.real-time-systems.com

Our partners

Real-Time Systems GmbH
Gartenstraße 33
88212 Ravensburg
Germany

Phone: +49 751 359 558 0
E-Mail: info@real-time-systems.com