acontis technologies GmbH

SOFTWARE

Hypervisor

User's Manual

Version 8.x

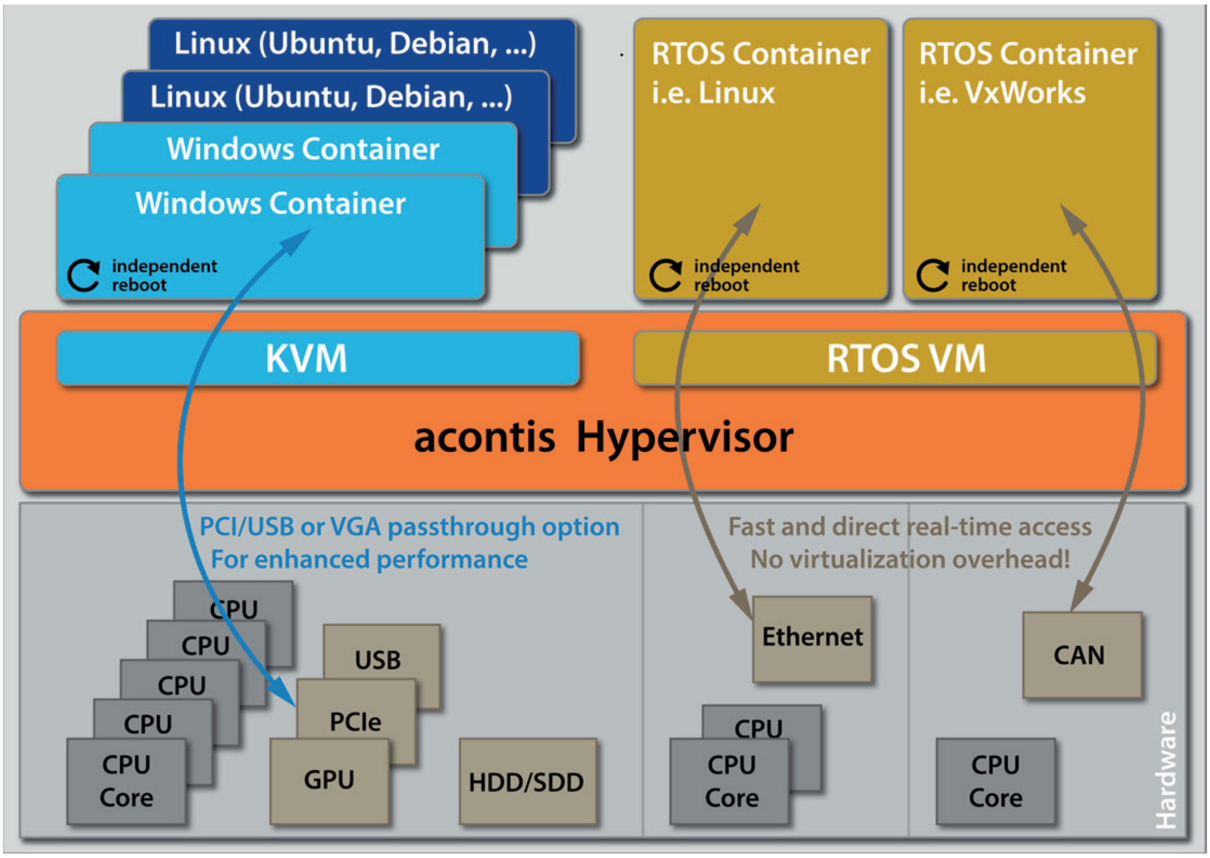
Edition: November 21, 2023

Contents

1 Introduction

1.1 Overview

Using the existing, industry proven acontis real-time RTOS-VM virtualization technology, multiple hard real-time operating systems (Real-time Linux, VxWorks, etc.) can run in native speed. In addition, based on the well-known and proven KVM virtualization solution, multiple standard operating systems like Windows and Linux (Ubuntu, Debian, Fedora, etc.) operate in parallel. Besides virtual hardware, KVM provides para-virtualization, PCI/USB and VGA pass-through for highest possible performance. Each guest OS is fully independent and separated and can be rebooted or shutdown while the other guests continue without being affected. The Hypervisor Host (or service OS) is based on a lightweight Ubuntu Linux operating system.



RTOS Virtual Machine Hypervisor

The RTOS Virtual Machine hypervisor technology provides an independent layer to run any RTOS in native speed. No virtualization overhead is introduced and all RTOS drivers as well the operating systems and applications have direct and fast hardware access. The same technology is successfully used by customers all over the world since more than 10 years in the existing acontis real-time solution. The new acontis Hypervisor can utilize this technology without modification so existing customer applications can be re-used without modification.

KVM Hypervisor

KVM is one of the most popular Type 1 hypervisors used in many high-availability and security critical environments like the cloud solutions from Google, Amazon or Oracle.

To increase performance, for example for fast USB, Ethernet or Graphics operation, the respective devices can be completely passed through to Windows or Linux guests. Alternatively, para-virtualized devices for the hard disk, the Ethernet or graphics controller reduce the amount of necessary hardware without compromising throughput.

Key technical features

The acontis Hypervisor is a perfect symbiosis between the wide-spread KVM virtualization solution and the industry proven RTOS Virtual Machine hypervisor for real-time operating systems.

General

- Supports Multiple OSes: real-time Linux, On Time RTOS-32, VxWorks® RTOS, Standard Linux, Microsoft® Windows®, proprietary Roll-your-own, Bare metal, any unmodified OS
- RTOS containers including applications run on bare metal core with no virtualization overhead and direct hardware access
- Fully separated and independent guest operation
- User defined guest startup sequence
- Utilize any number of CPU cores per single guest
- Independent reboot of all guests while others continue operation
- Virtual Network between all guests
- Inter-OS Communication: Shared Memory, Events, Interlocked data access, Pipes, Message Queues and Real-time sockets for high speed application level communication
- Hypervisor provided fileserver for all guests

KVM

Windows and Standard Linux operating systems like Ubuntu or Debian run under control of the KVM hypervisor. This hypervisor provides plenty of sophisticated features:

- Multiple Windows and/or standard Linux instances
- Windows/Linux containers with snapshot support to easily switch between different application situations without the need to install multiple OS instances. Snapshots create a view of a running virtual machine at a given point in time. Multiple snapshots can be saved and used to return to a previous state, in the event of a problem.
- Pass-through support: To increase performance, for example for fast USB, Ethernet or Graphics operation, the respective devices can be completely passed through to Windows or Linux guests.
- Paravirtualization: Para-virtualized devices for the hard disk, the ethernet or graphics controller reduce the amount of necessary hardware without compromising throughput.
- Graphics virtualization to provide 3D accelerated graphics to multiple guests.

RTOS-VM

- Multiple real-time Operating Systems (Linux, VxWorks, On Time RTOS-32, etc.)
- Fast real-time interrupt handling and short thread latencies
- Direct hardware access with no virtualization overhead

- Compatible with the existing acontis Windows real-time extension (applications can be shared or cross-migrated between both solutions)

1.2 Linux tips for Windows users

The following points may help a Windows user with the first Linux steps.

- Windows Command Prompt is called `Terminal` in Linux. The shortkey to open one is `Ctrl` + `Alt` + `T`.
- Instead of `notepad.exe` you can use `gedit`. If you have no graphical desktop available `nano` will work.
- It is recommended to use the `midnight commander` tool. You can install it with the `sudo apt-get install mc` command and then run `mc`.
- `cd \MyDirMySubDir` is `cd /MyDir/MySubDir` and `cd..` must be `cd ..`.
- `dir` is `ls` and `ls -l` will show additional information.
- `run as administrator` has its equivalent in `sudo`, which has to be put before a command. For example `sudo gedit` starts the editor as administrator (called `root` in Linux). You can use `sudo -s` to switch console to root user and `exit` to return.
- Most programs give helpful information being called with parameter `--help` or by calling the manual `man`. In case of `gedit` this would be `gedit --help` and `man gedit`.
- `chmod` can change the rights (*read/write/executable*) of a file. In Windows you just create a file `test.bat` and you can already execute it. In Linux you name it `test.sh`, but it won't run without being made an `executable`.
- `chown` can change the owner of a file.

2 System Setup

2.1 Hardware Requirements

Ressource	Recommended
CPU	Minimum: • 1 Core for the Hypervisor Host, which can be shared with non-real-time OS • +1 Core for *each* real-time OS. • +1 Core for *each* non-real-time OS if **not** shared with Hypervisor Host. **Important:** Do **not** enable Hyperthreading. **Important:** CPU **must** support VT technology!
Memory	4GB for Hypervisor Host and RTOS + minimum 4GB for Windows (as a guest).
GPU	
2^{nd} GPU	
Disk Space	50GB for Hypervisor Host + minimum 50GB for Windows (as a guest).

Hint: For installation purposes is it recommended to use an USB stick with at least 4GB. Further informations could be found at the Hypervisor Quickstart Guide!

2.2 Installation and Configuration

Hint: The hypervisor could be installed either exclusively on a PC or side by side with an existing OS. The **default** case is exclusively. The side by side installation is described at next chapter *Side By Side Installation*!

2.2.1 Side By Side Installation

If the hypervisor has been installed in parallel to Windows or Linux, the file `setrootuuid.sh` has to be adjusted. The original `defaultBoot` values need to be incremented by 1, e.g. set to `3` and `3`, respectively.

Enter `gedit setrootuuid.sh` to adjust these values, they are near the end of the file:

```
# update default boot configuration
#####################################
defaultbootRE='\(GRUB_DEFAULT=\)[0-9]*'
[ -d /sys/firmware/efi ] && defaultBoot=3 \|\| defaultBoot=3 # 2 for UEFI,␣
↪2 for Legacy BIOS
```

Caution: The above numbers (2, 3) may change depending on the hypervisor version!

Restart the system by entering:

```
$ sudo reboot
```

In the next step you need to initialize the Hypervisor. See *Hypervisor Initialization* for details.

2.3 Hypervisor Files and Directories

2.3.1 Directories

Directory	SubDir	Content
`/hv`	`.`	Hypervisor root directory
	`bin/.`	Binaries and scripts
	`config/.`	General config files and configuration scripts
	`doc/.`	Manuals and guides
	`services/.`	Services (automatically started when booting)
	`templates/.`	Template files (for guests, configuration, …)
	`sdk/.`	Software Development Kit
	`sysmgr/.`	System Manager (graphical configuration and diagnosis tool)
`/hv/guests`	`.`	Guests root folder
	`files/.`	Filesystem folder to be shared between guests
	`examples/.`	Example guests root folder
	`etc/.`	Additional guest specific examples, tools etc.

2.3.2 Configuration Files

The hypervisor configuration files are located in /hv/config and in the guest folders. The initial configuration files are located in /hv/templates/config. They are copied to /hv/config when running the /hv/bin/inithv.sh initialization script for the very first time.

The root configuration file hv.config is used by the hypervisor for the overall guest configuration. The following table shows the locations of the root configuration file hv.config

Directory	SubDir	Description
/hv	config/	Used by the System Manager generated guests
	guests/examples/rt-linux	Used by the example rt-linx guest
	guests/examples/vxworks	Used by the example vxworks guest

2.3.3 Brand Labeling

In case if you need to do brand labeling the shipped product, here is the list of image files for wallpapers, boot logos etc

Directory	SubDir	graphic file	Description
/usr/share	plymouth/themes/xubuntu-logo	logo.png logo_16bit.png	OS Boot logo
	xfce4/backdrops	xubutnu-jammy.png	Background image
/etc		lsb-release	Contains name RTOSVisor

3 Hypervisor Initialization

3.1 First Time Initialization

After finishing the setup and rebooting a boot menu is shown and the default entry `Ubuntu Configure RTOSVisor` is selected, keep this and do **not** select any other entry. After logging in, you need to change to the hypervisor bin directory:

```
$ cd /hv/bin
```

In the next step, the hypervisor needs to be initialized, also memory settings for the real-time guest OS have to be defined (these settings are **rtos-dependent!**). Use the `inithv` script:

```
sudo ./inithv.sh [-baseaddr rtos_baseaddr] rtos_memsize shm_memsize
```

The optional -baseaddr parameter of the `inithv.sh` script will set the physical base address where the RTOS shall be located. This parameter is required for RTOS guests which are not relocatable (e.g. VxWorks or RTOS-32). The next parameter defines the amount of memory to be assigned to the real-time guest OS, the last parameter will define the shared memory pool size. All units are in `MB`.

- **Real-time Linux** guest: `sudo ./inithv.sh 256 16`
- **VxWorks** guest: `sudo ./inithv.sh -baseaddr 64 64 16`
- **On Time RTOS-32** guest: `sudo ./inithv.sh -baseaddr 64 64 16`

Caution: If you want to install a Windows or Ubuntu guest, the shared memory pool size must be at least `16` MByte and a power of `2` (`16`, `32`, `64`, `128`, etc.)!

Caution: If you get the message `WARNING: CPU frequency not stable...` when running the `inithv.sh` script, you may not have properly disabled power settings in the BIOS. Please follow the respective instructions above (Prerequisites).

After! rebooting the Hypervisor, resetting the configuration using the `hv_resethv` command may fix this issue (`hv_resethv` implicitly re-runs `inithv`).

Hint:

If you want to change the memory configuration at a later time, please run the `hv_adjmemconf` command.
The following example shows how to change the configuration to use 768 MByte RAM located at the base address 64 MByte for the real-time guest and 32 MByte RAM for the shared memory pool.

```
    $ hv_adjmemconf -baseaddr 64 768 32
```

Now, to get the changes effective, a reboot is required:

```
$ sudo reboot
```

3.2 Reset Hypervisor Initialization

At any time later, you can reset the hypervisor initialization using the hv_resethv command.

Running this command will also reset the configuration (located in /hv/config) as well as the example guests (located in /hv/guests/examples).

You should preserve the previous configuration and guest settings in case you have changed them.

The hv_resethv command will automatically try to preserve these folders (e.g. into /hv/config.bak). If these preserved folder already exist, you may run hv_resethv -force to automatically remove the current settings without preserving them.

```
$ hv_resethv [-force]
```

You may also use different a memory configuration when running hv_resethv. The same parameters are used as in inithv.sh.

The following command will reset the configuration and set the RTOS memory size to 768 MByte and the Shared Memory size to 32 MByte. The base address settings will taken over from the previous configuration settings.

```
$ hv_resethv 768 32
```

The following command will reset the configuration and set the RTOS memory base address to 96 MByte, the RTOS and Shared Memory size will be taken over from the previous configuration settings.

```
$ hv_resethv -baseaddr 96
```

The following command will reset the configuration, set the RTOS memory base address to 64 MByte, set the RTOS memory size to 256 MByte and the Shared Memory size to 16 MByte.

```
$ hv_resethv -baseaddr 64 256 16
```

4 Hypervisor Guests - General

The RTOSVisor supports two different kind of guest operating systems.

- Unmodified, *non* Real-time operating systems, for example Windows 10 or Ubuntu 22.04. These guests run in a virtual machine under control of the KVM hypervisor technology.
- Para-virtualized Real-time operating systems (RTOS), for example Real-time Linux. These guests uses the Virtual Machine Framework (VMF) paravirtualization and run bare metal on the physical hardware.

A single guest is located in its respective guest folder.

4.1 Example guests

Some pre-configured guests are provided in the `/hv/guests/examples` folder. New guests can be created based on template files stored in `/hv/templates`. It is **not** recommended to create new guest folders manually, for this purpose the graphical System Manager tool shall be used.

Caution: The example guests share some configuration files with System Manager configured guests. This may lead to inconsistencies if the System Manager was used to configure guests before. In such case, before starting one of the example guests again, the configuration has to be reset. Run the following steps to do so:

```
$ hv_resethv
$ sudo reboot
```

The `hv_resethv` command will backup the existing `/hv/config` and `/hv/guests/examples` folder and reset then to the initial content. See also *Reset Hypervisor Initialization* for more details.

4.2 Guest filesystem access

The Hypervisor Host provides access to its filesystem (located on a physical hard disk or SSD) to guests. By default the `/hv/guests` folder is exposed to all guests.

4.3 General guest folder content

All guest folders contain the following files.

- `guest_config.sh`: the main guest configuration script, you must **not** change this file
- `usr_guest_config.sh`: user specific guest configuration script
- `guest.config`: guest specific configuration, you must **not** change this file
- `usr.config`: user specific guest configuration

Additional configuration files may exist depending on the guest type and operation mode.

4.4 Example guest folders

All pre-configured guests are stored in the /hv/guests/examples folder. A configuration to run one single RTOS in conjunction with one Windows and/or one Ubuntu guest folder is supported. The hypervisor configuration file hv.config is located in the respective guest folder and used for the global hypervisor configuration.

Hint: If the System Manger tool is used to configure guests, the hypervisor configuration stored in /hv/config/hv.config is used instead.

Hint: In case the example Windows or Ubuntu guest is started **before** starting an RTOS guest, the hv.config file in the Windows or Ubuntu guest is used. By default, the hv.config file located in the Windows or Ubuntu example guest is a link to the hv.config file of the RT-Linux example guest. If another example RTOS guest shall be used, this link needs to be adjusted. For example, if the example VxWorks guest shall be used together with the example Windows or Ubuntu guest, proceed as follows:

```
$ cd /hv/guests/examples/windows
$ rm hv.config
$ ln -s /hv/guests/examples/vxworks/hv.config hv.config
$ cd /hv/guests/examples/ubuntu
$ rm hv.config
$ ln -s /hv/guests/examples/vxworks/hv.config hv.config
```

4.5 Guest operation

To operate with a specific guest, you need to switch into the guest folder first. The following shows how to select the shipped Real-time Linux example guest.

```
$ cd /hv/guests/examples/rt-linux
```

The following operations are supported to operate with guests:

- hv_vmf_start [config_file]: Load the VMF and the hypervisor configuration. If not configuration file is given, the one in the current guest folder is used.
- hv_vmf_is_loaded: Returns information if the VMF is loaded
- hv_vmf_stop: Stop all RTOS guests and unload VMF. It is **not** recommended to use this command while guests are running.
- hv_guest_start [-view]: Boot a currently powered off guest. The -view option will automatically open the console to interact with the guest.
- hv_guest_restart [-view]: Reboot a guest without reloading the Hypervisor configuration (only supported for RTOS guests).
- hv_guest_console: Open the console to interact with the guest. The console for RTOS guests is a shell like interface, the console for KVM guests typically is the guest desktop.
- hv_guest_stop: Stop the guest.

Caution: Only **one** debug console windows for each RTOS guest is supported. If multiple console windows are opened, the behavior is undefined.

5 RTOS Guests (RT-Linux, VxWorks, On Time RTOS-32, ...)

Important: You need to be familiar with chapter *Hypervisor Guests - General* before reading this chapter!

5.1 RTOS guests, general

Some ready to use RTOS images (containers) are provided, they are located in the respective template directory beyond /hv/templates (e.g. /hv/templages/rt-linux for RT Linux containers).

The first time, when a RTOS container is started, the RTOS Virtual Machine Framework (VMF) is loaded. Loading the VMF will also load the hypervisor configuration stored in the hv.config configuration file. This hypervisor configuration describes all guests and their related RTOS guest ID which is used to *attach* the guest to the VMF. In case a configuration entry has changed, all RTOS containers need to be stopped and the RTOS Virtual Machine to be reloaded. Configuration files are stored in *.config files.

5.1.1 RTOS Fileserver

The Hypervisor Host includes a file server which by default exposes the /hv/guests folder to RTOS guests.

The following configuration entry defines the root folder of the file server:

```
[Host\FileServer]
    "HomeDir"="/hv/guests"
```

The root folder is defined in /hv/config/hvbase.config for a System Manager based guest configuration or in one of the *.config files of the example guests.

This file server will use proprietary methods (using the Hypervisor's RTOS-Library) to expose files in this folder. The RT-Linux and RTOS-32 guests include a filesystem driver for these methods.

5.1.2 RTOS guest folder

Besides the standard files described in *General guest folder content*, all RTOS guest folders contain the following additional file(s).

- device.config: guest specific device configuration, you must **not** change this file

Additional configuration files may exist depending on the guest type and operation mode.

5.2 Example Real-time Linux guest

The initial example guest files are located in /hv/templates/example_guests. They are copied to /hv/guests/examples when running the /hv/bin/inithv.sh initialization script for the very first time.

Important: you need at least **256 Mbyte** RAM for the shipped RT-Linux image. Assure the /hv/bin/inithv.sh script was executed with the appropriate RAM size.

The example Real-time Linux guest is located in /hv/guests/examples/rt-linux

Real-time Linux guest specific files are stored in /hv/guests/examples/rt-linux/files:

- autostart.sh: RT-Linux autostart script, you may enhance this script to automatically run your RT-Linux application.
- timesync.sh: Time-synchronization script, this will assure the RT-Linux time and date is consistent with the Hypervisor Host.
- remote_cmd_exec.sh: Example script which is used to show how to run a RT-Linux application from within a Windows or Ubuntu guest.

5.2.1 File system access for RT-Linux guests

The Hypervisor Host file system is mounted into the /mnt/rtfiles folder of RT-Linux. The **Hypervisor Host** folder /hv/guests acts as the root folder for all guests and is mounted into this folder.

5.3 Example VxWorks guest

The example VxWorks guest is located in /hv/guests/examples/vxworks.

Important: The shipped VxWorks image can only be executed if it is loaded into memory at a fixed address of 64 MByte. In the default configuration this is the case. When using the System Manager tool it must be configured as the very first Real-time guest.

5.3.1 File system access for VxWorks guests

To access the Hypervisor filesystem from within VxWorks, you may use FTP.

Follow these instructions to install an FTP server in the Hypervisor in case it is not yet installed.

- sudo apt-get install vsftpd
- activate the entry write_enable=YES in /etc/vsftpd.conf (you need to edit it with root rights)
- create a new entry local_root=/hv/guests
- Restart the FTP server: sudo systemctl restart vsftpd

- **Create an FTP user target with password vxworks:**
  - adduser target
  - passwd vxworks
  - You may use visudo to add the user target to the sudoers group (duplicate the root entry)
- After executing the above steps, from within VxWorks, you should be able to access the guest folder via the VxWorks path `pc:`

5.4 Example On Time RTOS-32 guest

Caution: All On Time RTOS-32 example guest images are located at a fixed address of 64 MByte. Please assure, you have used the parameter `-baseaddr 64` when calling `inithv.sh`, `hv_adjmemconf` or `hv_resethv`.

The example RTOS-32 guest is located in `/hv/guests/examples/rtos-32`

The guest image is selected through the `osImage` variable in the configuration script. By default it is the RTOS-32 loader image `/hv/templates/rtos-32/Loader.bin` which is set in `/hv/guests/examples/rtos-32/guest_config.sh`. If you want to change the RTOS-32 image, you should set the `osImage` variable in the user specific guest configuration script `usr_guest_config.sh`.

If the `Loader.bin` is used, the loader will use the application DLL which is defined in the following configuration section in `/hv/guests/examples/rtos-32/guest.config`:

```
[Rtos\Loader]                                ; Used by Loader.bin
     "DllName"="rtos32app.dll"               ; File or link must exist in the guest directory
```

By default this is a link to the effective DLL: `/hv/guests/examples/rtos-32/rtos32app.dlm` which by default points to the `RTOS-32Demo.dlm` demo application.

You may select a different application, for example the `RealtimeDemo` as follows:

```
$ cd /hv/guests/examples/rtos-32
$ rm rtos32app.dlm
$ ln -s files/RealtimeDemo.dlm rtos32app.dlm
```

5.4.1 File system access for RTOS-32 applications

From within the RTOS-32 application you may access the Hypervisor filesystem via the `C:` drive. The root directory is set in `guest.config` in section `[\\Host\\FileServer]`. By default, it is set to the example guest folder at `/hv/guests/examples/rtos-32`.

5.5 RTOS Shared Mode operation

Caution: This feature is **not** supported in RTOSVisor V8.1!

Using the RTOSVisor, Intel(c) VT technology allows to activate so called “Shared Mode”. In Shared Mode it is possible to run a RTOS on the same CPU as the Hypervisor Host (thus, the CPU is shared between two OSes). Please note, that hard real time for RTOS is also achieved in this mode.

Intel(c) VT-x and VT-d are hardware virtualisation extensions for Intel Processors. Please take a look at official Intel page for full description of the technology: https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

5.5.1 Activating VT

VT should be activated *manually*. In this example we configure `2` CPU system, where first CPU is used to run Hypervisor Host and RTOS1 (so, in shared CPU mode) and second CPU will be used to run RTOS2 (as exclusive core).

Note: Before you start to change anything in your system, please make sure, you have executed `/hv/bin/inithv.sh` script at least once.

This script installs necessary `.deb` packages, configures memory settings, reads `ACPI` tables and etc.

5.5.2 Linux Kernel grub parameters

IOMMU must be *disabled* in order to activate VT-D.

Add the following parameters to the linux kernel in `grub.cfg`: `intremap=off intel_iommu=off maxcpus=1`

Make sure these parameters have been removed: `iommu=pt vfio_iommu_type1.allow_unsafe_interrupts=1`

Currently **only** one shared cpu is supported. This is the reason, why `maxcpus=1` is used.

5.5.3 Adapt `.config` files

1. **`[Upload]`**

   ```
   "VersionDrv"=dword:9070000
   ```

2. **`[Vmf]`**

   ```
   "VtAllowed"=dword:1
   ```

3. **`[Rtos]`**

   ```
   "MemoryType"=dword:3
   ```

4. **`[Rtos1]`**

   ```
   "MemoryType"=dword:3
   ```

5. **[Rtos\Vmf]**
   `"MapSystemTables"=dword:1`

5.6 Achieving Hard Real-Time Capabilities

5.6.1 Background information

It is important to understand, which factors have influence to the real-time capabilities of your hardware. Most important of them are BIOS Settings (no CPU power saving modes should be activated, no CPU throttling, no variable CPU Frequency, USB Legacy support should also be disabled). System Management Interrupts (SMIs) must be also avoided.

Video Card has usually a huge impact on the real-time capabilities of the real-time system, because it can not only generate SMI but also perform huge background DMA transfers from/to DRAM memory.

For example, if you have Intel i915 video card on your Hypervisor Host, its Linux driver can produce significant interrupt/context latencies.

First step, in order to achieve better latencies, is to disable LightDM graphics manager in your Xubuntu Installation of RTOSVisor. Simply run the following command `sudo apt-get remove lightdm` and the reboot your system.

This step converts your system into console only mode, no GUI. But it is not enough, because a video card can do DMA Transfers even without GUI.

The easiest and fastest solution here is to kill its driver as follows:

- establish connection with Hypervisor Host via SSH (port 22)
- kill video driver `sudo rmmod -f i915`. Now the computer monitor **cannot** be used anymore.
- start realtime demo and make sure, that context switch/interrupt delays much better now (200%+):

5.6.2 Real-time measurement

RT-Linux realtime demo

The shipped RT-Linux guests include a real-time measurement application.

Follow these steps to run the measurement:

```
$ cd /hv/guests/examples/rt-linux
$ hv_vmf_stop
$ hv_guest_start -view
```

Then log in (user = root, password = root)

RealtimeDemo

RTOS-32 realtime demo

You can use the RTOS-32 realtime demo application to determine the real-time capabilities of your system.

If you did not change the shipped RTOS-32 configuration, follow these steps to run the measurement:

```
$ cd /hv/guests/examples/rtos-32
$ rm rtos32app.dlm
$ ln -s files/RealtimeDemo.dlm rtos32app.dlm
$ hv_vmf_stop
$ hv_guest_start -view
```

6 KVM Guests (Windows, Ubuntu, Debian, ...)

Important: You need to be familiar with chapter *Hypervisor Guests - General* before reading this chapter!

6.1 KVM guests, general

There are no example KVM guests shipped with the RTOSVisor. Instead, these operating systems have to be installed under control of the hypervisor from an installation media (an ISO file).

6.2 Communication Subsystem

A KVM guest may use the communication subsystem which provides specific function to communication with other guests:

- Direct access to the Virtual Network (compared to bridged access if the communication subsystem is not used)
- RTOS-Library functions (Shared Memory, Pipes, Events etc.)

Hint: The communication subsystem is part of the RTOS Virtual Machine Framework (VMF)

To enable access to the communication subsystem, the following settings are required in the `guest_config.sh` or `usr_guest_config.sh` configuration file:

```
export rtosOsId=#####
export hvConfig=%%%%%
```

The `rtosOsId` value needs to be a unique RTOS guest id, this id is used by the guest to attach to the communication subsystem. Valid ids are in the range from `1` to `4`. The `hvConfig` value wil have to point to the appropriate `hv.config` file used for the hypervisor configuration.

The very first time, when a RTOS container or such a KVM guest is started, the RTOS Virtual Machine Framework (VMF) is loaded. Loading the VMF will also load the hypervisor configuration stored in the `hv.config` configuration file. This hypervisor configuration describes all guests and their related guest ID which is used to *attach* the guest to the VMF. In case a configuration entry has changed, all RTOS containers need to be stopped and the RTOS Virtual Machine to be reloaded. Configuration files are stored in `*.config` files.

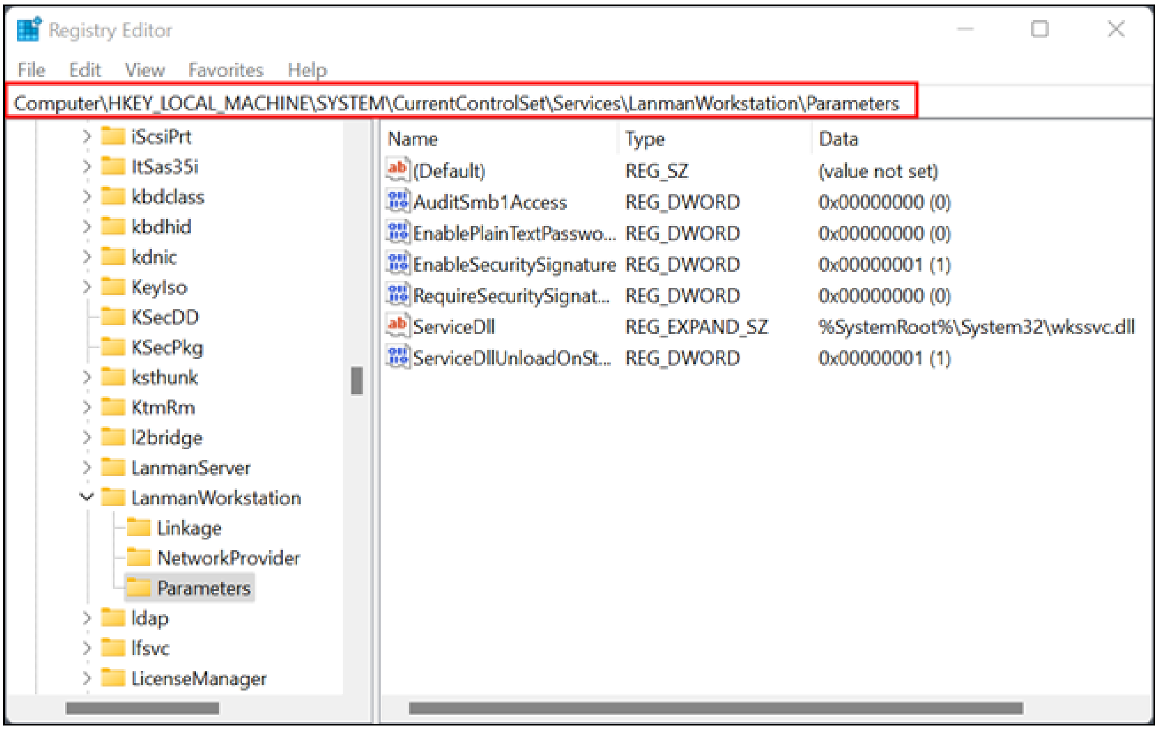
6.2.1 File sharing

The Hypervisor Host by default exposes the /hv/guests folder to KVM guests via a SMB share. The share can be accessed from within the guest using the IP address 10.0.2.4 and the share name qemu. For example on Windows 10 it can be accessed through \\10.0.2.4\qemu. The exposed folder is set in the /hv/bin/kvmguest_start.sh script in parameter smb of the USERNET configuration.

```
if [ $private_nw -eq 1 ]; then
    # user network
    USERNET="-device virtio-net,netdev=networkusr,mac=$ethmacVM2"
    USERNET=$USERNET" -netdev user,id=networkusr,smb=$HV_ROOT/guests"
    echo "private network MAC = "$ethmacVM2
else
    echo "no virtual network"
fi
```

Caution: You may get an error (0x80004005) when accessing the share due to some restrictive Windows settings. In that case, try to allow insecure guest logins. Windows blocks guest logins to network devices using SMB2 by default. You might need to disable that setting.

Open the Registry Editor and then navigate to HKLM\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parame using the menu on the left, or just paste the path into the address bar.

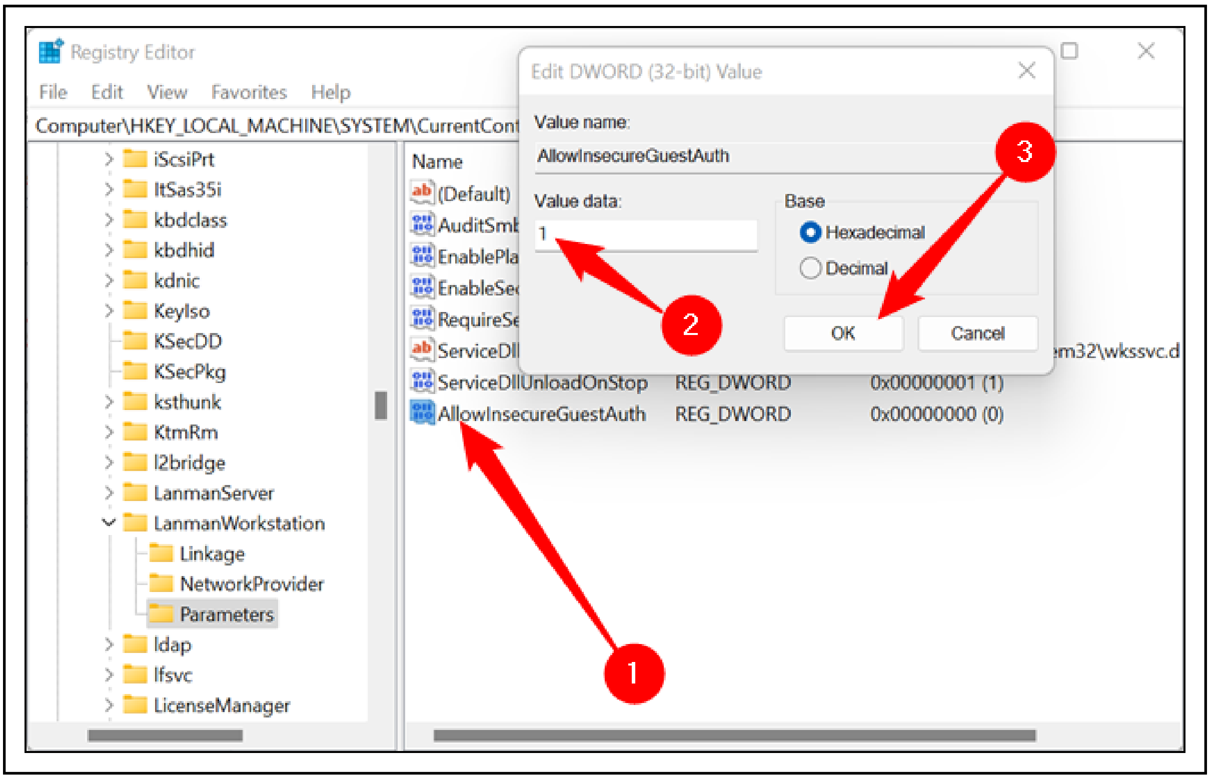


Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters

Name	Type	Data
(Default)	REG_SZ	(value not set)
AuditSmb1Access	REG_DWORD	0x00000000 (0)
EnablePlainTextPasswo...	REG_DWORD	0x00000000 (0)
EnableSecuritySignature	REG_DWORD	0x00000001 (1)
RequireSecuritySignat...	REG_DWORD	0x00000000 (0)
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\System32\wkssvc.dll
ServiceDllUnloadOnSt...	REG_DWORD	0x00000001 (1)

The DWORD you're looking for is named AllowInsecureGuestAuth — if it isn't there, you'll need to create it.

Right-click empty space, mouse to "New," then click "DWORD (32-bit) Value." Name it "AllowInsecureGuestAuth" and set the value to 1.



Name	Type	Data
ServiceDllUnloadOnStop	REG_DWORD	0x00000001 (1)
AllowInsecureGuestAuth	REG_DWORD	0x00000000 (0)

Edit DWORD (32-bit) Value

Value name: AllowInsecureGuestAuth

Value data: 1

Base: ● Hexadecimal ○ Decimal

6.2.2 KVM guest folder

Besides the standard files described in *General guest folder content*, all KVM guest folders contain the following additional file(s).

- `vm_shutdown_hook.sh`: guest shutdown handler
- `OVMF_CODE.fd`: UEFI firmware image
- `OVMF_VARS.fd`: UEFI firmware variable store
- `guest_gateway.config`: Configuration for guest port forwarding from an external ethernet network to the virtual network

Additional configuration files may exist depending on the guest type and operation mode.

6.3 KVM Guest Operation

Besides the commands described in *Guest operation* the following KVM guest specific commands are available.

- `hv_guest_start [-view | -kiosk]`: Start the guest. The `-view option` will show the guest desktop in standard mode (resizable window), the `-kiosk` option in full screen mode.
- `hv_guest_console [-kiosk]`: Show guest desktop in full screen mode (`-kiosk` option).
- `hv_guest_stop [-reset | -kill]`: Reset (`-reset` option) or Power off (`-kill` option) the guest (instead of gracefully shutdown without any option).
- `hv_guest_monitor`: Start the KVM guest monitor. Monitor commands are described in here: https://en.wikibooks.org/wiki/QEMU/Monitor

6.3.1 Kiosk mode

If you want to generally enable the kiosk mode after the next start of the guest VM, adjust the guest configuration:

```
$ cd GUEST_FOLDER
$ gedit usr_guest_config.sh
```

```
 export kiosk_mode=1
```

6.3.2 Displaying the guest desktop

To display the guest desktop, you need to run the `hv_guest_console` command from within the guest folder. Typically you need to first log in and then start the guest and the console. It will then be displayed on the RTOSVisor desktop.

Alternatively it is also possible to use X11 forwarding in a SSH session. The display device to be used is set in the `DISPLAY` environment variable. A local display is identified by `:0.0` a X11 forwarded display can look like `localhost:10.0`. You can determine it as follows:

```
$ echo $DISPLAY
```

To check if the display is accessible, run:

```
$ [[ $(xset -q 2 2>/dev/null) ]] && echo "display works"
```

If the result is `display works`, then everything is set correctly.

Hint: The display is accessed via the X protocol which needs authentication. The authentication file is defined in the `XAUTHORITY` environment variable. Typically it is located in the user's home folder: `/home/MyUser/.Xauthority`. If for some reason this file does not exist, you may **turn off** authentication:

```
$ xhost +
```

NOTE: The `root` user typically does not have access to the display.

Hint: This section applies to the System Manager. By default, the System Manager is automatically started as a service without any display access. Thus, the desktop (guest console) for KVM guests cannot be launched by default. To enable launching the console from within the System Manager, you may have to log in and restart the System Manager service:

```
$ hv_sysmgr restart
```

You typically restart the service from with the RTOSVisor locally (by logging in in its desktop), in that case thr RTOSVisor desktop will be used to show the guest desktop. Alternatively, if you restart the service inside a SSH session, there will be two options:

- If X11 forwarding is enabled, the guest desktop will be forwarded to a remote display.
- If X11 forwarding is disabled, the local desktop will be used. In this case you need to login first before launching the console.

6.3.3 Log files

The following log files (located in the guest folder) will be created when starting a guest:

- `qemuif.log` log messages of setting up network bridging on guest boot and terminate network bridging on guest shutdown.
- `kvmguest.log` guest VM logging (KVM hypervisor).
- `kvmview.log` guest VM viewer log file.
- `shutdown_svc.log` guest shutdown service log messages.
- `shutdown_hook.log` guest shutdown hook log messages.

6.4 Example KVM Windows guest

The example KVM Windows guest configuration is located in `/hv/guests/examples/windows`. You need to install the Windows guest using an ISO installation media. Please read the `Windows Guest Guide` for more information about Windows guests.

> **Caution:** The `hv.config` configuration file is a link to the RT-Linux example guest configuration file. See *Example guest folders* for more information.

6.5 Example KVM Ubuntu guest

The example KVM Ubuntu guest configuration is located in `/hv/guests/examples/ubuntu`. You need to install the Ubuntu guest using an `ISO` installation media. Please read the `Ubuntu Guest Guide` for more information about Ubuntu guests.

> **Caution:** The `hv.config` configuration file is a *link* to the RT-Linux example guest configuration file. See *Example guest folders* for more information.

6.6 KVM guest basic settings

Hint:

The term `GUEST_FOLDER` is related to the folder where the guest configuration files are located (e.g. `/hv/guests/examples/windows`).
The term `GUEST_NAME` is related to the guest name (e.g. `windows`).

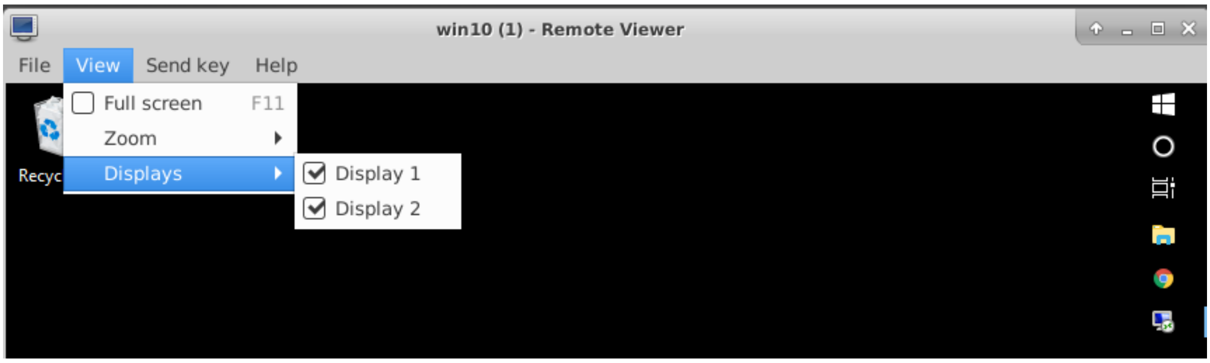
6.6.1 Guest Multiple monitors

If more than one monitor is connected to the system, these can also be used for VM guests (up to a maximum of 4 monitors).

For Windows guests, the following settings are required in `usr_guest_config.sh`:

```
export num_monitors=# (where # is the number of monitors)
```

To enable additional monitors being displayed, select the displays via the View – Displays menu in the viewer application.



6.6.2 Guest Multi Touch

In KVM Windows guests, multitouch functionality is available. Therefore, it is necessary to display the guest in full-screen mode either by using remote-viewer or kiosk mode.

To enable multitouch, it's necessary to identify the corresponding touch event on the Hypervisor Host:

```
ls -la /dev/input/by-id | grep -event-
```

The following settings are required in `usr_guest_config.sh`:

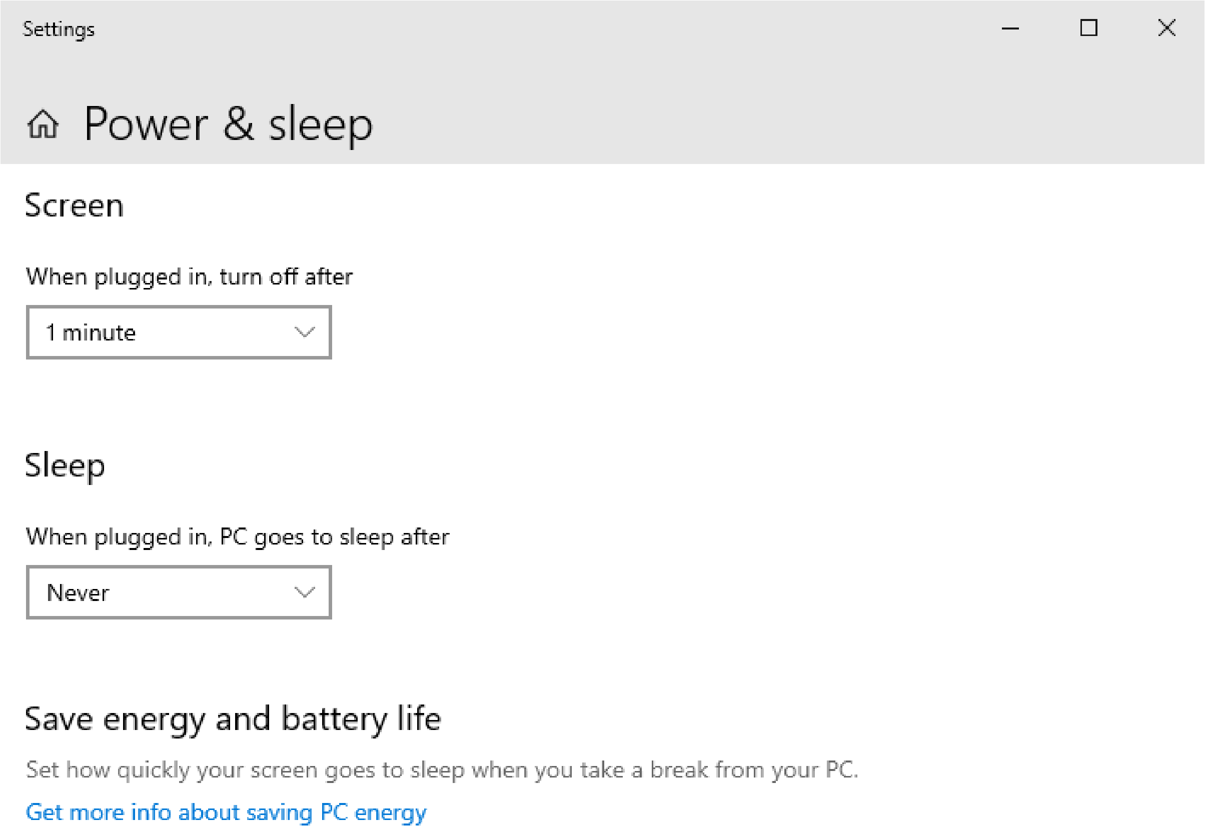
```
export enable_multitouch=1
export multitouch_event=    # fill in the corresponing touch event
```

6.7 Windows installation

Attention: See **Hypervisor - Windows Guest Guide** for tutorial.

6.7.1 Additional settings for Windows guests:

Disable Sleep (*Windows Settings -> System -> Power and sleep*):



Settings

Power & sleep

Screen

When plugged in, turn off after

1 minute

Sleep

When plugged in, PC goes to sleep after

Never

Save energy and battery life

Set how quickly your screen goes to sleep when you take a break from your PC.

Get more info about saving PC energy

6.8 Ubuntu installation

Attention: See **Hypervisor - Ubuntu Guest Guide** for tutorial.

6.9 Windows/Linux COM port guest access

It is possible to passthrough a COM port to a guest VM. It works and for both, legacy serial ports and for USB-to-Serial converters.

modify the usr_guest_config.sh file and change

```
export OTHER_HW=$OTHER_HW
```

to

```
export OTHER_HW=$OTHER_HW"  -serial /dev/ttyUSB0 -serial /dev/ttyS0"
```

this example creates COM1 Port in the VM and it uses USB-to-Serial converter on a Hypervisor Host and creates COM2 port in the VM and it is the real COM1 port on Hypervisor Host.

6.10 PCI Device passthrough (Windows/Linux)

6.10.1 Why pass-through

Typically, Windows or Linux guest operating systems will run in a sandbox like virtual machine with no direct hardware access. There are scenarious when this is not sufficient, for example some PCI devices (e.g. CAN cards) will not be virtualized and thus would not be visible in such guest OS. There may also be significant performance impacts in some cases if virtual hardware is used (especially for for graphics hardware). To overcome these limitations, the guest will have to use the real physical hardware instead of virtual hardware. PCI Device passthrough will directly assign a specific PCI device to a Windows or Linux guest.

It is **mandatory** to have hardware support for IOMMU (VT-d). VT-d should be supported by your processor, your motherboard and should be enabled in the BIOS.

Virtual Function I/O (VFIO) allows a virtual machine to access a PCI device, such as a GPU or network card, directly and achieve close to bare metal performance.

The setup used for this guide is:

- Intel Core I5-8400 or I3-7100 (with integrated Intel UHD 630 Graphics) - this integrated graphics adapter will be assigned to the Windows VM.
- AMD/ATI RV610 (Radeon HD 2400 PRO) – optional, as a second GPU, only needed to have display output for Hypervisor Host. Later, the Hypervisor Host is reached only via SSH.
- Intel I210 Gigabit Network Card - optional, to demonstrate how to pass through a simple PCI device to a Windows VM.

6.10.2 Ethernet PCI Card/Custom PCI device assignment

Some manual work is required to pass through the PCI Device to a Windows VM.

Understanding IOMMU Groups

In order to activate the hardware passthrough we have to prevent the ownership of a PCI device by its native driver and assign it to the vfio-pci driver instead.

In a first step, an overview of the hardware and related drivers is required. In the below example we want to passthrough the I210 Ethernet Controller to the guest VM.

```
rte@rte-System-Product-Name:~$ lspci
00:00.0 Host bridge: Intel Corporation 8th Gen Core Processor Host Bridge/
↪DRAM Registers
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v5/E3-1500 v5/6th Gen␣
↪Core Processor
00:02.0 VGA compatible controller: Intel Corporation Device 3e92
00:14.0 USB controller: Intel Corporation Cannon Lake PCH USB 3.1 xHCI␣
↪Host Controller
00:14.2 RAM memory: Intel Corporation Cannon Lake PCH Shared SRAM (rev 10)
00:16.0 Communication controller: Intel Corporation Cannon Lake PCH HECI␣
↪Controller
00:17.0 SATA controller: Intel Corporation Cannon Lake PCH SATA AHCI␣
↪Controller (rev 10)
```

(continues on next page)

(continued from previous page)

```
00:1c.0 PCI bridge: Intel Corporation Device a33c (rev f0)
00:1c.5 PCI bridge: Intel Corporation Device a33d (rev f0)
00:1c.7 PCI bridge: Intel Corporation Device a33f (rev f0)
00:1f.0 ISA bridge: Intel Corporation Device a303 (rev 10)
00:1f.4 SMBus: Intel Corporation Cannon Lake PCH SMBus Controller (rev 10)
00:1f.5 Serial bus controller [0c80]: Intel Corporation Cannon Lake PCH␣
↪SPI Controller
01:00.0 VGA compatible controller: [AMD/ATI] RV610 [Radeon HD 2400 PRO]
01:00.1 Audio device: [AMD/ATI] RV610 HDMI Audio [Radeon HD 2400 PRO]
03:00.0 Ethernet controller: Intel Corporation I210 Gigabit Network␣
↪Connection (rev 03)
04:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168/
↪8411 PCI Express Gigabit Ethernet Controller (rev 15)
```

The I210 device is determined as 03:00.0

Caution: The IOMMU has to be activated, you can verify this as follows:

```
sudo dmesg | grep -e "Directed I/O"
DMAR: Intel(R) Virtualization Technology for Directed I/O
```

Hint: In case the IOMMU is activated, all devices are divided into groups. The IOMMU Group is an indivisible unit. All devices in the same group must be passed through together, it is not possible to only pass through a subset of the devices.

In the next step, we need to get an overview of the IOMMU architecture and determine to which group the device we want to pass through belongs to.

```
for a in /sys/kernel/iommu_groups*; do find $a -type l; done | sort --
↪version-sort
```

```
/sys/kernel/iommu_groups/0/devices/0000:00:00.0
/sys/kernel/iommu_groups/1/devices/0000:00:01.0
/sys/kernel/iommu_groups/1/devices/0000:01:00.0
/sys/kernel/iommu_groups/1/devices/0000:01:00.1
/sys/kernel/iommu_groups/2/devices/0000:00:02.0
/sys/kernel/iommu_groups/3/devices/0000:00:14.0
/sys/kernel/iommu_groups/3/devices/0000:00:14.2
/sys/kernel/iommu_groups/4/devices/0000:00:16.0
/sys/kernel/iommu_groups/5/devices/0000:00:17.0
/sys/kernel/iommu_groups/6/devices/0000:00:1c.0
/sys/kernel/iommu_groups/7/devices/0000:00:1c.5
/sys/kernel/iommu_groups/8/devices/0000:00:1c.7
/sys/kernel/iommu_groups/9/devices/0000:00:1f.0
/sys/kernel/iommu_groups/9/devices/0000:00:1f.4
/sys/kernel/iommu_groups/9/devices/0000:00:1f.5
/sys/kernel/iommu_groups/10/devices/0000:03:00.0
/sys/kernel/iommu_groups/11/devices/0000:04:00.0
```

The I210 device (03:00.0) belongs to the IOMMU group 10. It is important to know that all devices in a single group are shared. No other device belongs to this IOMMU group so we can pass through the I210 device to the guest.

We need to determine the PCI vendor and device ID of the I210 device now:

```
rte@rte-System-Product-Name:~$ lspci -s 03:00.0 -vvn
03:00.0 0200: 8086:1533 (rev 03)
    ...
    Kernel driver in use: igb
    Kernel modules: igb
```

Add to the linux kernel command line parameters the following: `vfio-pci.ids=8086:1533`. To edit kernel parameters edit `/etc/grub.d/40_custom` file and then execute `update-grub` and reboot.

```
sudo gedit /etc/grub.d/40_custom

menuentry 'Hypervisor' --class ubuntu --class gnu-linux --class gnu --
↪class os $menuentry_id_option 'gnulinux-simple-8b4e852a-54b3-4568-a5a2-
↪dec7b16b8e07' {
        recordfail
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_gpt
        insmod ext2
        set root='hd0,gpt2'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt2 --
↪hint-efi=hd0,gpt2 --hint-baremetal=ahci0,gpt2  8b4e852a-54b3-4568-a5a2-
↪dec7b16b8e07
        else
          search --no-floppy --fs-uuid --set=root 8b4e852a-54b3-4568-a5a2-
↪dec7b16b8e07
        fi
        linux   /boot/vmlinuz-5.15.0-88-acontis root=UUID=8b4e852a-54b3-
↪4568-a5a2-dec7b16b8e07 ro  quiet splash $vt_handoff find_preseed=/
↪preseed.cfg auto noprompt priority=critical locale=en_US memmap=8k\
↪$128k memmap=8M\$56M memmap=256M\$64M memmap=16M\$324M maxcpus=3␣
↪intel_pstate=disable acpi=force idle=poll nohalt pcie_port_pm=off pcie_
↪pme=nomsi cpuidle.off=1 intel_idle.max_cstate=0 noexec=off noexec32=off␣
↪nox2apic intel_iommu=on iommu=pt intremap=off vfio_iommu_type1.allow_
↪unsafe_interrupts=1 vfio-pci.ids=10ec:8168
        initrd  /boot/initrd.img-5.15.0-88-acontis
}
```

```
sudo update-grub
sudo reboot
```

Hint: Normally no other steps are needed to to replace the native driver by the vfio-pci driver. If you have conflicts between these two drivers, it may be required to disable the loading of the native driver. Add the parameter `module_blacklist=igb` to the kernel command line in that case. See below example:

```
linux   /boot/vmlinuz-5.4.17-rt9-acontis+ root=UUID=ebbe5511-f724-4a1d-
↪b5a5-e8dafecaf451 ro  quiet splash $vt_handoff find_preseed=/preseed.cfg␣
↪auto noprompt priority=critical locale=en_US
```

(continues on next page)

(continued from previous page)

```
        memmap=8k\$128k memmap=8M\$56M memmap=256M\$64M memmap=16M\$384M maxcpus=7 intel_pstate=disable acpi=force idle=poll nohalt pcie_port_pm=off pcie_pme=nomsi cpuidle.off=1
        intel_idle.max_cstate=0 noexec=off nox2apic intremap=off vfio_iommu_type1.allow_unsafe_interrupts=1 intel_iommu=on iommu=pt module_blacklist=igb
```

Hint: As mentioned before, device passthrough requires IOMMU support by the hardware and the OS. It is needed to add `intel_iommu=on iommu=pt` to your kernel command line. These parameters are automatically added by Hypervisor when executing the `/hv/bin/inithv.sh` script.

Typically, one or multiple PCI devices will also be assigned to a RTOS, for example Real-time Linux. In such case (one or more PCI devices are passed through to a Windows or Ubuntu guest as well as one or multiple PCI devices are assigned to a RTOS) it is required to deactivate IOMMU Interrupt Remapping in the Linux Kernel.

The following kernel command line parameters usually are added as well by the `/hv/bin/inithv.sh` script into the GRUB Entry "Hypervisor" in the `/etc/grub.d/40_custom` file. You may verify and add these parameters if they are missing. In that case it is also required to execute update-grub and then reboot.

```
intremap=off vfio_iommu_type1.allow_unsafe_interrupts=1
```

VM configuration

Last step is to edit the `usr_guest_config.sh` file located in the `GUEST_FOLDER` and add the PCI Etherner Card information here.

Uncomment `#export OTHER_HW` variable and set it to:

```
export OTHER_HW=" -device vfio-pci,host=03:00.0"
```

6.10.3 Intel Integrated Graphics (iGVT-d) assignment

To use graphics passthrough, less steps compared to standard PCI hardware passthrough are required, because Hypervisor automates most of the steps.

Understanding GPU modes UPT and Legacy

There are two modes "legacy" and "Universal Passthrough" (UPT).

Hypervisor uses only Legacy mode, but it could be important to understand the difference.

UPT is available for Broadwell and newer processors. Legacy mode is available since SandyBridge. If you are unsure, which processor you have, please check this link https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures

In Legacy it is meant that IGD is a primary and exclusive graphics in VM. Additionally the IGD address in the VM must be PCI 00:02.0, only 440FX chipset model (in VM) is supported and not Q35. The IGD must be the primary GPU for Hypervisor Host as well (please check your BIOS settings).

In UPT mode the IGD can have another PCI adress in VM and the VM can have a second graphics adapter (for example qxl, or vga).

Please read here more about legacy and UPT mode: https://git.qemu.org/?p=qemu.git;a=blob;f=docs/igd-assign.txt

There a lot of other little things, why IGD Passthrough could not work. For ex. In legacy mode it expects a ISA/LPC Bridge at PCI Adress `00:1f.0` in VM and this is a reason, why Q35 chip does not work, because it has another device at this adress.

In UPT mode, there is no output support of any kind. So the UHD graphics can be used for accelerating (for ex. Decoding) but the Monitor remains black and there is a non-standard experimental qemu vfi-pci command line parameter `x-igd-opregion=on`, which can work.

Blocklisting the i915 driver on the Hypervisor Host

The standard `Intel Driver i915` is complex and it is not always possible to safely unbind the device from this driver, that is why this driver is blocklisted by Hypervisor when executing `/hv/bin/inithv.sh` script.

Deactivating Vesa/EFI Framebuffer on Hypervisor Host

Please also know, when i915 driver is disabled, there are other drivers which are ready to jump on the device to keep the console working. Depend on your BIOS settings (legacy or UEFI) two other drivers can occupy a region of a video memory: `efifb` or `vesafb`.

The Hypervisor blocklists both by adding the following command line parameter:

```
video=vesafb:off,efifb:off
```

Please also check if it works: `cat/proc/iomem`. If you still see that one of this driver still occupies a part of a video memory, please try manually another combination:

```
video=efifb:off,vesafb:off.
```

Legacy BIOS, pure UEFI and CSM+UEFI in Hypervisor Host

It plays also significant role, in which mode your machine is booted: Legacy BIOS, pure UEFI or UEFI with CSM support. In pure UEFI (on Hypervisor Host) QEMU cannot read video ROM. In this case you could extract it manually (for ex. Using *Cpu-Z* utility or just boot in CSM mode, when iGPU is a primary GPU in BIOS), patch it with correct device id and provide it to qemu as *romfile=* parameter for `vfi-pci`. Please google for *rom-parser* and *rom-fixer* for details.

SeaBIOS and OVMF (UEFI) in VM

It also plays role which BIOS you use in the VM itself. For QEMU there are two possibilities: SeaBIOS (legacy BIOS, which is default for qemu) and OVMF (UEFI).

For your convenience the RTOSVisor is shipped with precompiled OVMF binaries located in the /hv/templates/kvm directory: OVMF_CODE.fd OVMF_VARS.fd

By default, OVMF UEFI does not support OpRegion Intel feature, which is required to have a graphics output to a real display. There are three possibilities how to solve this problem and the easiest one seems to be the using special vbios rom vbios_gvt_uefi.rom, please read more here https://wiki.archlinux.org/index.php/Intel_GVT-g.

For your convenience, this file is already included in the Hypervisor package in the /hv/bin directory.

Hypervisor uses OVMF (UEFI) for graphics pass-through.

How to do it in Hypervisor

The final working configuration which we consider here:

- CPU Graphics is a primary GPU in BIOS (Hypervisor Host)
- Hypervisor Host boots in pure UEFI mode
- OVMF is used as BIOS in Windows VM
- vbios_gvt_uefi.rom is used as VBIOS in romfile parameter for vfio-pci
- Legacy mode for IGD, so the Windows VM has only one graphics card Intel UHD 630

Which commands should be executed in Hypervisor to do a pass-through of a Intel Integrated Graphics to a Windows VM?

None! Almost everything is done automatically. When executing /hv/bin/inithv.sh script (which is required to install real-time linux kernel and to reserve kernel memory for hypervisor needs), a separate GRUB entry “Hypervisor + iGVT-d” is created.

This entry contains already all necessary linux kernel parameters required to do a graphics pass-through: blocklisting intel driver, disabling interrupt remapping, assigning a VGA device to a vfio-pci driver and other stepts.

But one step should be done once, configuring your VM.

Change usr_guest_config.sh script located in the GUEST_FOLDER. Two variables should be uncommented and activated:

```
export uefi_bios=1
export enable_vga_gpt=1
```

Just reboot your machine, choose “Hypervisor+iGVT-d” menu item. If everything is correct, the display of your Hypervisor Host should remain black. Connect to the machine using SSH connection or use a second graphics card for Hypervisor Host (read next chapter) and then start the guest using hv_guest_start

Wait 30-60 seconds and.. display remains black? Of course. Windows does not have Intel Graphics drivers.

Remember we configured Windows for a Remote Desktop Access in previous steps? Connect to Windows VM via RDP and install latest Intel Drivers https://downloadcenter.intel.com/product/80939/Graphics

If everyting is done correctly, your display should now work and display a Windows 10 Desktop.

Using Second GPU Card for Hypervisor Host

X-Windows on the Hypervisor Host does not work properly, when the primary GPU in the System is occupied by the `vfi-pci` driver. It detects the first GPU, tries to acquire it, fails and then aborts. We should let it know, that it should use our second GPU card instead.

Log in to the Hypervisor Host (Press `Ctrl` + `Alt` + `F3`, for example), shutdown LightDM manager sudo service lightdm stop.

Execute

```
$ sudo X -configure
```

it creates the `xorg.conf.new` file in the current directory. When this command is executed, the X server enumerates all hardware and creates this file . By default, in modern systems, Xserver does not need the `xorg.conf` file, because all hardware is detected quite good and automatically. But the config file is still supported.

Look at this file, find a section “Device” with your second GPU (look at the PCI Adress). Copy content of this section to a separate file `/etc/X11/xorg.conf.d/secondary-gpu.conf`. Save and reboot.

```
Section "Device"
    ### Available Driver options are:-
    ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
    ### <string>: "String", <freq>: "<f> Hz/kHz/MHz",
    ### <percent>: "<f>%"
    ### [arg]: arg optional
    #Option "Accel" # [<bool>]
    #Option "SWcursor" # [<bool>]
    #Option "EnablePageFlip" # [<bool>]
    #Option "SubPixelOrder" # [<str>]
    #Option "ZaphodHeads" # <str>
    #Option "AccelMethod" # <str>
    #Option "DRI3" # [<bool>]
    #Option "DRI" # <i>
    #Option "ShadowPrimary" # [<bool>]
    #Option "TearFree" # [<bool>]
    #Option "DeleteUnusedDP12Displays" # [<bool>]
    #Option "VariableRefresh" # [<bool>]
    Identifier "Card0"
    Driver "amdgpu"
    BusID "PCI:1:0:0"
EndSection
```

Assigning an External PCI Video Card to Windows VM

External PCI Video Cards are not automatically recognized by `/hv/bin/inithv.sh` script (unlike the CPU integrated video), so this entry is not added to the grub boot menu.

So if you want to pass the external GPU to a VM through, you first need to create a separate GRUB entry or modify an existing one.

Let’a assume we’ve already executed `/hv/bin/inithv.sh` script (as described in previous chapters) and it created a “Hypervisor” boot entry.

Boot computer using this “Hypervisor” entry.

Open `/boot/grub.cfg`, find its corresponding

```
menuentry 'Hypervisor'
```

section

Edit it and rename the menu entry name to:

```
menuentry 'Hypervisor + Nvidia Quadro Passthrough'
```

The kernel command line in this section should like like:

```
linux   /boot/vmlinuz-5.4.17-rt9-acontis+ root=UUID=ebbe5511-f724-4a1d-
↪b5a5-e8dafecaf451 ro  quiet splash $vt_handoff find_preseed=/preseed.cfg␣
↪auto noprompt priority=critical locale=en_US
        memmap=8k\$128k memmap=8M\$56M memmap=256M\$64M memmap=16M\$384M␣
↪maxcpus=7 intel_pstate=disable acpi=force idle=poll nohalt pcie_port_
↪pm=off pcie_pme=nomsi cpuidle.off=1
        intel_idle.max_cstate=0 noexec=off nox2apic intremap=off vfio_
↪iommu_type1.allow_unsafe_interrupts=1 intel_iommu=on iommu=pt
```

1) First thing, we should discover the topology of PCI devices

find it out, which PCI slot is used for our external PCI card. type `lspci`

```
00:00.0 Host bridge: Intel Corporation Device 9b53 (rev 03)
00:02.0 VGA compatible controller: Intel Corporation Device 9bc8 (rev 03)
00:08.0 System peripheral: Intel Corporation Xeon E3-1200 v5/v6 / E3-1500␣
↪v5 / 6th/7th Gen Core Proc
00:12.0 Signal processing controller: Intel Corporation Device 06f9
00:14.0 USB controller: Intel Corporation Device 06ed
00:14.1 USB controller: Intel Corporation Device 06ee
00:14.2 RAM memory: Intel Corporation Device 06ef
00:16.0 Communication controller: Intel Corporation Device 06e0
00:17.0 SATA controller: Intel Corporation Device 06d2
00:1b.0 PCI bridge: Intel Corporation Device 06c0 (rev f0)
00:1b.4 PCI bridge: Intel Corporation Device 06ac (rev f0)
00:1c.0 PCI bridge: Intel Corporation Device 06b8 (rev f0)
00:1c.4 PCI bridge: Intel Corporation Device 06bc (rev f0)
00:1c.5 PCI bridge: Intel Corporation Device 06bd (rev f0)
00:1f.0 ISA bridge: Intel Corporation Device 0685
00:1f.4 SMBus: Intel Corporation Device 06a3
00:1f.5 Serial bus controller [0c80]: Intel Corporation Device 06a4
02:00.0 VGA compatible controller: NVIDIA Corporation GM206GL [Quadro␣
↪M2000] (rev a1)
02:00.1 Audio device: NVIDIA Corporation Device 0fba (rev a1)
04:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. Device 8125␣
↪(rev 05)
05:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network␣
↪Connection
```

Ok. `02:00.0` is our device. Let’s discover, which driver occupies this device.

```
type "lspci -s 02:00.0 -vv"

02:00.0 VGA compatible controller: NVIDIA Corporation GM206GL [Quadro␣
↪M2000] (rev a1) (prog-if 00 [VGA controller])
      Subsystem: NVIDIA Corporation GM206GL [Quadro M2000]
      ...
      Kernel modules: nvidiafb, nouveau
```

So we know now, which drivers should be blocklisted in kernel: `nvidiafb` and `noueveau`.

But, it looks like we have an integrated audio device in `02:00.1`. Most probably we should deactivate its driver as well.

Let’s investigate our IOMMU groups, type `find /sys/kernel/iommu_groups/ -type l`:

```
/sys/kernel/iommu_groups/7/devices/0000:00:1b.0
/sys/kernel/iommu_groups/15/devices/0000:05:00.0
/sys/kernel/iommu_groups/5/devices/0000:00:16.0
/sys/kernel/iommu_groups/13/devices/0000:02:00.0
/sys/kernel/iommu_groups/13/devices/0000:02:00.1
/sys/kernel/iommu_groups/3/devices/0000:00:12.0
/sys/kernel/iommu_groups/11/devices/0000:00:1c.5
/sys/kernel/iommu_groups/1/devices/0000:00:02.0
/sys/kernel/iommu_groups/8/devices/0000:00:1b.4
/sys/kernel/iommu_groups/6/devices/0000:00:17.0
/sys/kernel/iommu_groups/14/devices/0000:04:00.0
/sys/kernel/iommu_groups/4/devices/0000:00:14.1
/sys/kernel/iommu_groups/4/devices/0000:00:14.2
/sys/kernel/iommu_groups/4/devices/0000:00:14.0
/sys/kernel/iommu_groups/12/devices/0000:00:1f.0
/sys/kernel/iommu_groups/12/devices/0000:00:1f.5
/sys/kernel/iommu_groups/12/devices/0000:00:1f.4
/sys/kernel/iommu_groups/2/devices/0000:00:08.0
/sys/kernel/iommu_groups/10/devices/0000:00:1c.4
/sys/kernel/iommu_groups/0/devices/0000:00:00.0
/sys/kernel/iommu_groups/9/devices/0000:00:1c.0
```

so, we see, that our NVidia card belongs to a IOMMU group 13, together with its audio device.

So, repeat steps for audio device, type `lspci -s 02:00.1 -vv`:

```
02:00.1 Audio device: NVIDIA Corporation Device 0fba (rev a1)
...
Kernel modules: snd_hda_intel
```

Now all these 3 drivers should be blocklisted in your system.

add the following to your “Hypervisor” entry kernel command line: `module_blacklist=nouveau,nvidiafb,snd_hda_intel`

2) Assign devices to KVM.

In order to make it possible to pass NVidia VGA and Audio device to a Windows VM through we should assign a special driver `vfi-pci` to each our drvice.

Let’s determine Vendor and Device IDs:

```
type "lspci -s 02:00.0 -n"
02:00.0 0300: 10de:1430 (rev a1)

type "lspci -s 02:00.1 -n"
02:00.1 0403: 10de:0fba (rev a1)
```

add the following to our kernel command line: `vfio-pci.ids=10de:1430,10de:0fba`

so our final command line should now look like:

```
linux   /boot/vmlinuz-5.4.17-rt9-acontis+ root=UUID=ebbe5511-f724-4a1d-
↪b5a5-e8dafecaf451 ro  quiet splash $vt_handoff find_preseed=/preseed.cfg␣
↪auto noprompt priority=critical locale=en_US
        memmap=8k\$128k memmap=8M\$56M memmap=256M\$64M memmap=16M\$384M␣
↪maxcpus=7 intel_pstate=disable acpi=force idle=poll nohalt pcie_port_
↪pm=off pcie_pme=nomsi cpuidle.off=1
        intel_idle.max_cstate=0 noexec=off nox2apic intremap=off vfio_
↪iommu_type1.allow_unsafe_interrupts=1 intel_iommu=on iommu=pt module_
↪blacklist=nouveau,nvidiafb,snd_hda_intel
        vfio-pci.ids=10de:1430,10de:0fba
```

if you boot the computer into the this new grub entry, you could check if vfio-pci driver successfully acquired our devices:

```
type "dmesg | grep vfio"

[    5.587794] vfio-pci 0000:02:00.0: vgaarb: changed VGA decodes:␣
↪olddecodes=io+mem,decodes=io+mem:owns=none
[    5.606445] vfio_pci: add [10de:1430[ffffffff:ffffffff]] class 0x000000/
↪00000000
[    5.626451] vfio_pci: add [10de:0fba[ffffffff:ffffffff]] class 0x000000/
↪00000000
```

everything is ok.

3) Check if your PCI Card is not a primary graphics device in your system. Boot into BIOS and find the corresponding settings. For every BIOS there are own namings for this option.

Often there is no such option in BIOS and it then detects which HDMI ports are connected and if both Integrated GPU and the external PCI card are connected to monitors, then the integrated card is choosen by BIOS as a primary device. If not, then your PCI card is selected as the primary device.

If your card is not a primary device and integrated CPU graphics is used for Hypervisor Host, skip reading this section. But if not, you should disable frame buffer, because its drivers occupy video card PCI regions.

Add then `` video=efifb:off,vesafb:off disable_vga=1`` to your kernel command line.

This trick is also useful, when your PC has no integrated CPU graphics and the external PCI Video card is the only device in your system.

4) assign your VGA and Audio devices to your Windows VM (Qemu)

usually the VM configuation is located in this file: `GUEST_FOLDER/usr_guest_config.sh`

find and change `OTHER_HW` line from

```
export OTHER_HW=$OTHER_HW
```

to

```
export OTHER_HW=$OTHER_HW" -device vfio-pci,host=02:00.0 -device vfio-pci,
↪host=02:00.1 -nographic"
```

Note: Please note, you need to install your video card manufacturer drivers to a Windows VM, to make the graphics working.

Create a temporary VGA device or a Windows RDP/Qemu VNC connection to your VM to install graphics drivers.

6.10.4 Keyboard and Mouse assignment

Normally your Hypervisor Host with Windows VM and Integrated Graphics passed through works in headless mode. Windows VM outputs to a monitor durch DVI-D/HDMI connection and the Hypervisor Host is controlled via SSH connection. Windows has a look and feel as it works without an intermediate hypervisor layer.

So, Windows needs a keyboard and mouse.

Go to /dev/input/by-id/ and find something that looks like a keyboard and mouse and it should containt **"-event-"** in its name.

```
ls -la

usb-18f8_USB_OPTICAL_MOUSE-event-if01 -> ../event5
usb-18f8_USB_OPTICAL_MOUSE-event-mouse -> ../event3
usb-18f8_USB_OPTICAL_MOUSE-if01-event-kbd -> ../event4
usb-18f8_USB_OPTICAL_MOUSE-mouse -> ../mouse0
usb-SEM_USB_Keyboard-event-if01 -> ../event7
usb-SEM_USB_Keyboard-event-kbd -> ../event6
```

configure your VM with these parameters by editing usr_guest_config.sh:

```
export vga_gpt_kbd_event=6
export vga_gpt_mouse_event=3
```

Please also note, disconnecting evdev devices, such as keyboard or mouse, can be problematic when using qemu and libvirt, because it does not reopen device when the device reconnects.

If you need to disconnect/reconnect your keyboard or mouse, there is a workaround, create a udev proxy device and use its event device instead. Please read more hier https://github.com/aiberia/persistent-evdev.

If everything works, you'll find new devices like uinput-persist-keyboard0 pointing to /dev/input/eventXXX. Use these ids as usual in:

```
export vga_gpt_kbd_event=XXX
export vga_gpt_mouse_event=ZZZ
```

6.11 Hypervisor Host reboot/shutdown

6.11.1 Reboot/shutdown initiated by Windows guest

In this section we will show how the whole system can be rebooted or shutdown from within the Windows guest

- Start the Windows guest
- Open the Explorer
- Switch to folder \\10.0.2.4\qemu\files\WinTools\DesktopShortcuts

- Copy all the shortcuts onto your desktop (you may have to adjust the shortcuts based on your Windows version and language)
- Run the *"System Shutdown"* or *"System Reboot"* shortcut

Hint: To avoid being asked to allow execution, you may adjust the related Windows settings.

Open the Control Panel, select Internet Options, select security, select local intranet, select sites, select advanced and add `\\10.0.2.4\qemu`

Hint: Whenever the guest is started, the script `/hv/bin/kvm_shutdown_svc.sh` is executed which will then wait for the guest to terminate. After the guest has terminated (either via a graceful shutdown or when it crashed or powered off) the `vm_shutdown_hook.sh` script located in the `GUEST_FOLDER` will be executed. This script will then reboot or gracefully shutdown the whole system.

You may adjust the `vm_shutdown_hook.sh` script according to your needs (e.g. to add additional cleanup tasks).

6.11.2 Reboot/shutdown initiated by Linux guest

Currently there is no built-in option to reboot or shutdown the system. In general, the same solution as for Windows guests can be applied.

6.12 Start a Real-time Application

6.12.1 Start RT-App in Windows guest

In this section we will show how to execute an application on the RT-Linux guest from within the Windows guest.

Caution: It is assumed that

- all the provided desktop shortcuts have been placed on the desktop, see last chapter.
- RT-Linux was started before the Windows guest.

- Start the Windows guest
- Download and install the appropriate putty package from https://www.putty.org/
- Execute the *"Hypervisor Attach"* shortcut
- Open putty and connect to `192.168.157.2`, accept the connection (do **not** login)
- Close putty (we only need to execute this step to accept the security keys, they are needed to run the RT-Linux application)
- Execute the *"Run RT-Linux Application"* shortcut
- A message should be shown.

Hint: This shortcuts executes the "remote_cmd_exec" sample script located in /hv/guests/examples/rt-linux/files/remote_cmd_exec.sh.

You may adjust this file according to your needs.

6.12.2 Start RT-App in Linux guest

In this section we will show how to execute an application on the RT-Linux guest from within the Windows guest.

Caution: It is assumed that

- all the steps described in Hypervisor Ubuntu Guest have been executed.
- RT-Linux was started before the Ubuntu guest.

- Start the Ubuntu guest
- Execute the *"Hypervisor Attach"* shortcut or call hv_attach in console
- Execute ssh root@192.168.157.2 in console
- Log into Real-Time Linux:

```
$ vmf64 login: root
$ password:  root
```

7 KVM Guest Network

There are various options how to configure a KVM guest for networking.

- Private network between the guest and the Hypervisor Host (e.g. for file sharing)
- Virtual network provided by the communication subsystem to communicate between KVM guests and RTOS guests via TCP/IP.
- One single dynamically bridged automatic network connection to the external network (the network adapter typically is shared between all guests and the hypervisor).
- Network adapter statically bridged with a physical adapter in the Hypervisor Host (typically only used by a single guest).
- Network adapter statically bridged with the virtual network (if the communication subsystem shall not be used by the guest).
- Network adapter dynamically bridged with the virtual network (if the communication subsystem shall not be used by the guest).
- Guest internal statically bridged network (KVM guests can communicate over this internal network without using a physical ethernet controller and without using the virtual network).

The network configuration is based on various settings provided in `usr_guest_config.sh`. See details below.

If one or multiple network configurations are enabled using a static brdige, such bridge is created once when the guest is started. This bridge then will be preserved until they system is rebooted.

To verify which bridges are available, run:

```
$ brctl show
```

You can remove these bridges (after stopping all guests) using the command `hv_brdelall`:

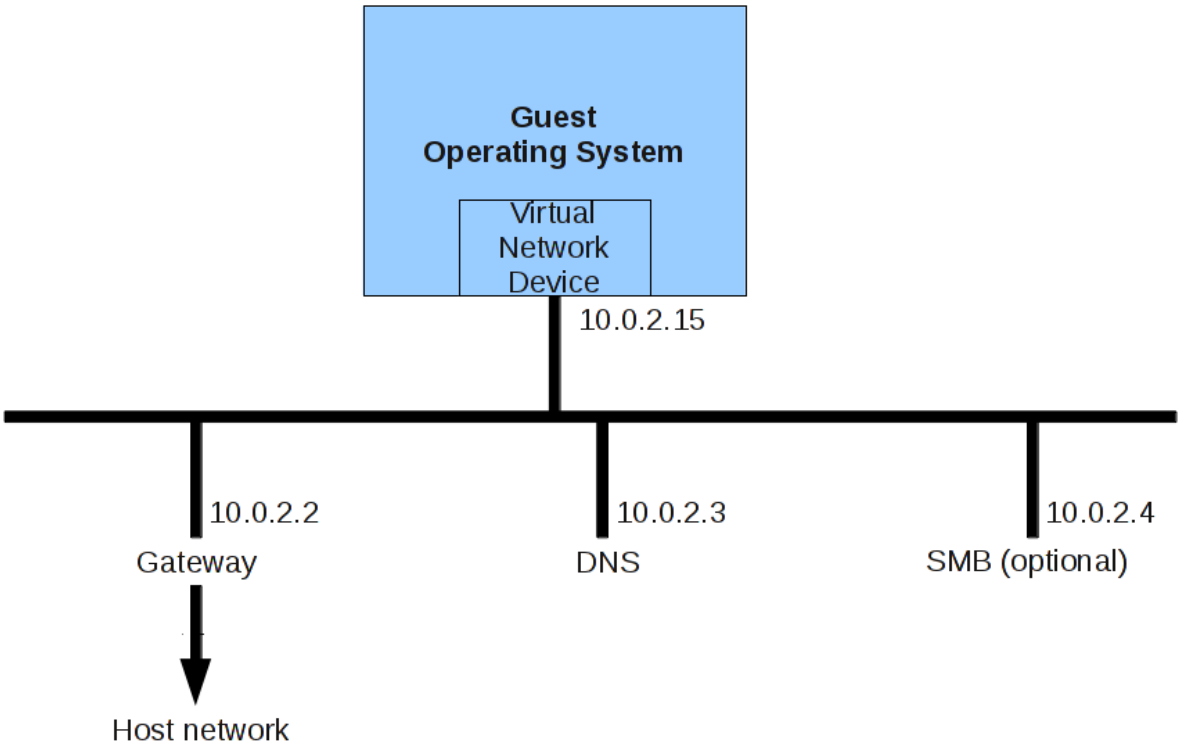
```
$ hv_brdelall
```

Caution: All static bridges are created once when a guest is started. If you stop the guest and change related configuration you need to remove the related bridges, otherwise the changes will not become effective and the behaviour may become undefined. To accomplish this, just stop all guests and remove all bridges:

```
$ hv_brdelall
```

7.1 Private guest/Hypervisor network

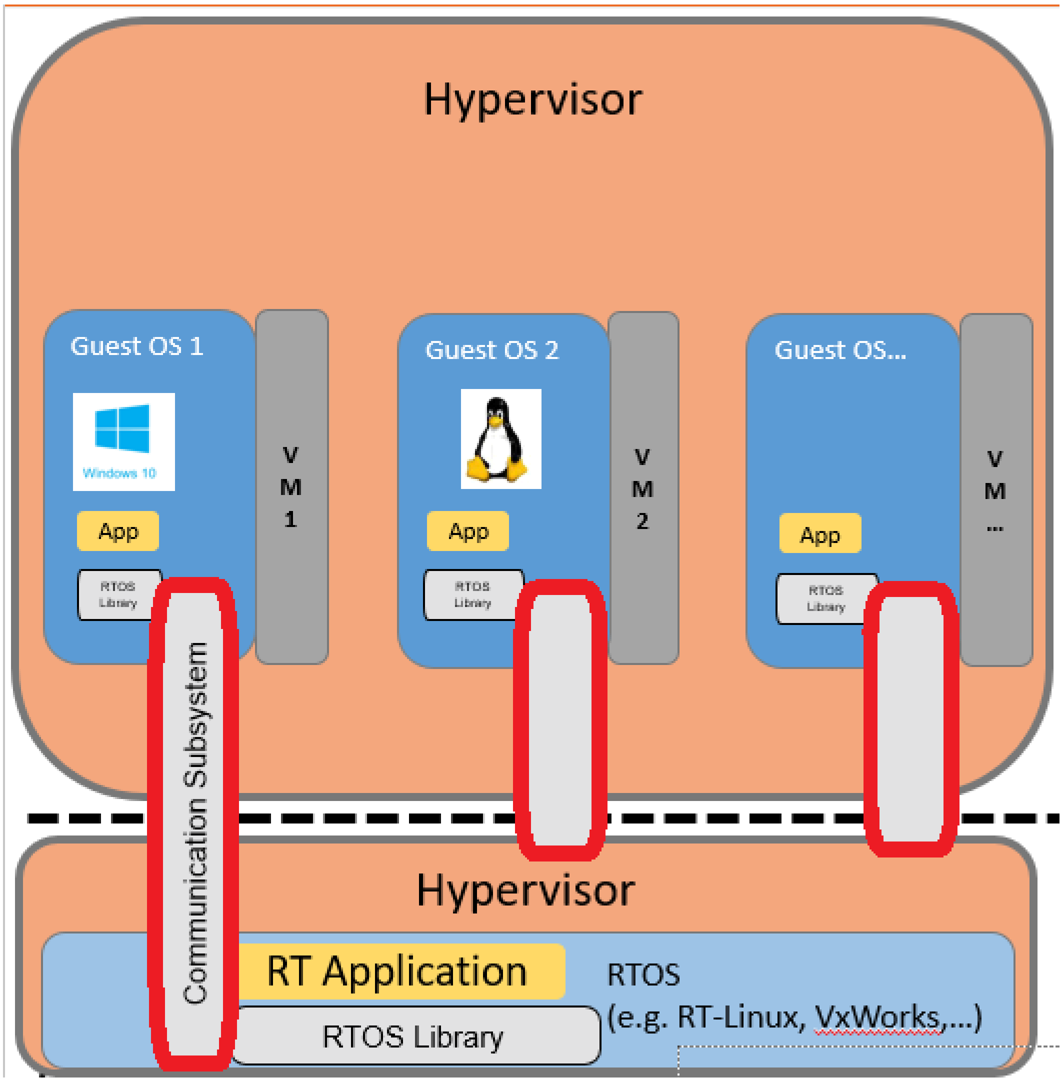
The private network is a Hypervisor internally provided network which is connected to the external physical network via NAT (if the Hypervisor Host is connected to the external network). This connection is safe because it cannot be accessed from outside, but it is also much slower. It is also used for file sharing with the Hypervisor Host. The architecture of the private network looks as follows:



By default, the SMB server is enabled. This server provides access to the Hypervisor Host file system. For details, see *File sharing*

7.2 Communication subsystem Virtual Network

The communication subsystem is provided by the hypervisor to support guest communication. One of the communication means is the Virtual Network.



To use the communication subsystem in KVM guests, the hypervisor support package needs to be installed. Prior to use the communication subsystem, the KVM guest has to `attach` to the Virtual Machine Framework (VMF) using the Uploader tool. After installation of the support package, a respective shortcut should be available on the desktop.

The IP addresses of the virtual network are set to fixed values.
The default IP addresses are:
Hypervisor Host: 192.168.157.1
First RTOS guest: 192.168.157.2
Windows example guest: 192.168.157.3

You have to set the IP addresses as well as the MAC addresses in the guests to unique values.

7.3 Automatic external network connection (dynamically bridged)

One single automatic network connection to the external network is provided. To enable this network, set `external_nw` to a value of `1`. This network typically is shared between all guests and the hypervisor.

The `netif_mode` setting determines, how this network is set up.

- 0: manual parameter setting. The parameters `netif_m`, `defaultgw_m`, `dns_m`, `brip_m` and `brnm_m` have to be set properly. The Hypervisor Host network settings also need to be set manually then.
- 1: automatic parameter setting. This is the default setting and should be used if the network where the hypervisor is connected to supports DHCP.
- 2: no IP for Hypervisor Host. In this case, the Hypervisor Host has no IP connection via the adapter defined in `netif_m`.

You may have to set the Hypervisor Host network adapter accordingly. See *Hypervisor Host network configuration* for details.

When the guest is started, a bridge is dynamically created in the Hypervisor Host. This will bridge the physical device with a so called `tap` device representing the guest. After stopping the guest, this bridge is removed.

The name of the statically created bridge will be `vmbridge`.

7.4 Bridged external network connection (static bridge)

One or multiple physical network adapters can be used in the guest using a (static) bridge. Using this method, multiple network adapters can be assigned to guests. Typically in such a use case, a single network adapter is only used by one single guest (not used by the Hypervisor Host nor shared by multiple guests).

To enable this mode, you need to set `phys_nw` to a value of `1` and enable one or multiple entries `physnw_dev`, `physnw_ip` and `physnw_repmac`. For each single network adapter all three values need to be provided. You should replicate the MAC address if only a single guest is using this adapter. You **must not** replicate the MAC address if the Hypervisor Host or other guests are using the adapter as well.

The name of the statically created bridge will be `br-DEVICE`, for example `br-enp1s0` for the `enp1s0` device.

7.5 Bridged virtual network connection (static bridge)

If you want to communicate between a KVM guest and a RTOS guest, you can use the virtual network. This is either possible using the communication subsystem as described above. Alternatively, a static bridge can be used, then avoiding the need to use the communication subsystem.

You need to set `rtosvnet_nw` to a value of `1` to accomplish this.

The name of the statically created bridge will be `br-rtosvnet`.

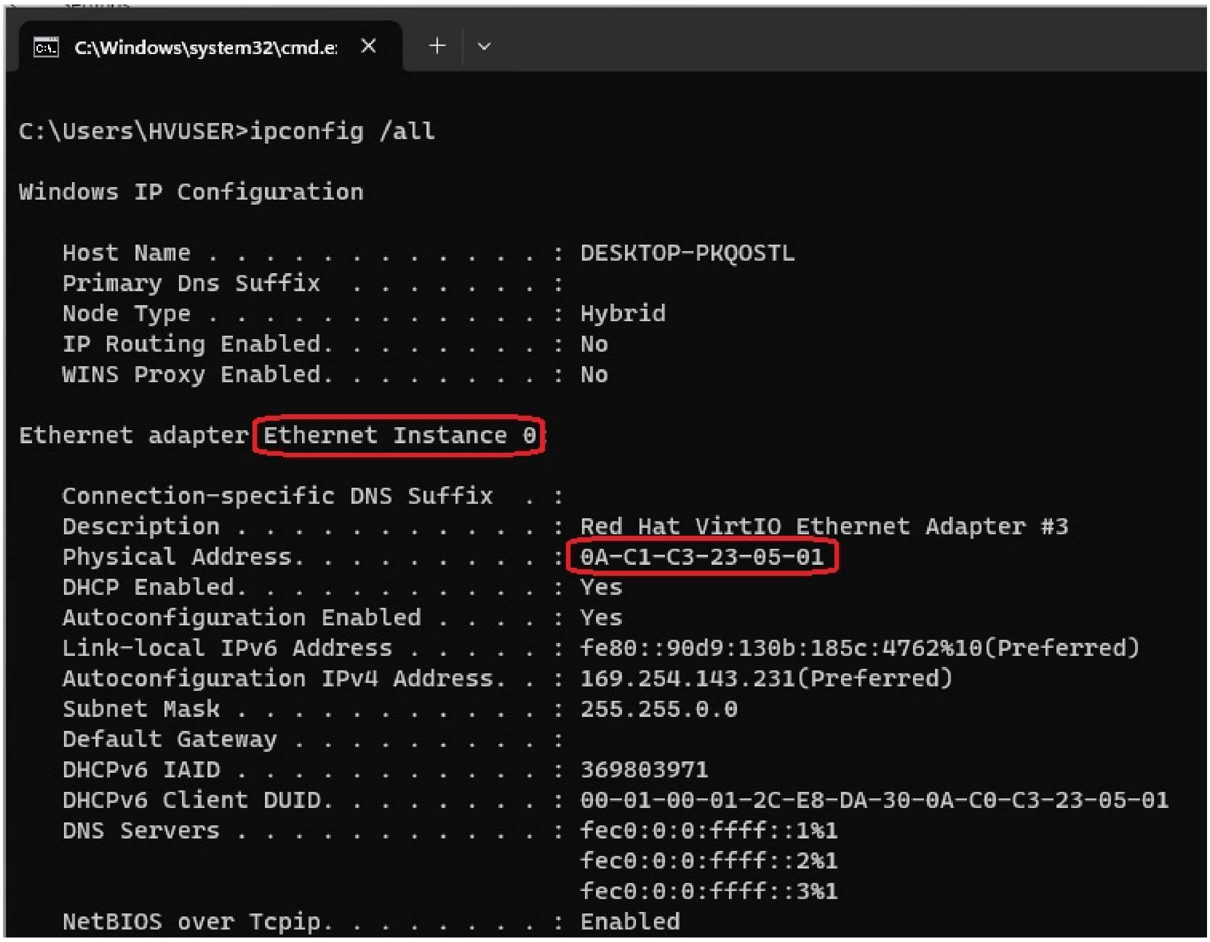
To establish a connection from the KVM guest with the virtual network, an IP address must be assigned to the corresponding Ethernet adapter in the KVM Guest. To modify the address of the virtual network, you can change the value of rtosvnet_br_ip in the configuration file usr_guest_config. By default, the address is set to 192.168.157.1.

Here is an example for a Windows guest:

Query the MAC address with brctl showmacs **after** the Windows guest has finished booting.

```
$ $ brctl showmacs br-rtosvnet
$ port no  mac addr           is local?    ageing timer
$   1      00:60:c8:00:00:00  yes          0.00
$   1      00:60:c8:00:00:00    yes          0.00
$   2      0a:c1:c3:23:05:01    no          0.13          #---- MAC address
↪ needed for configuration
$   2      ea:e8:fd:54:69:ee    yes          0.00
$   2      ea:e8:fd:54:69:ee    yes          0.00
```

Use the MAC address with the 'is local = no' entry for configuring the network card. The network card in the KVM guest can be identified by running ipconfig /all in the console. Set the network address in the Windows Network Connections.



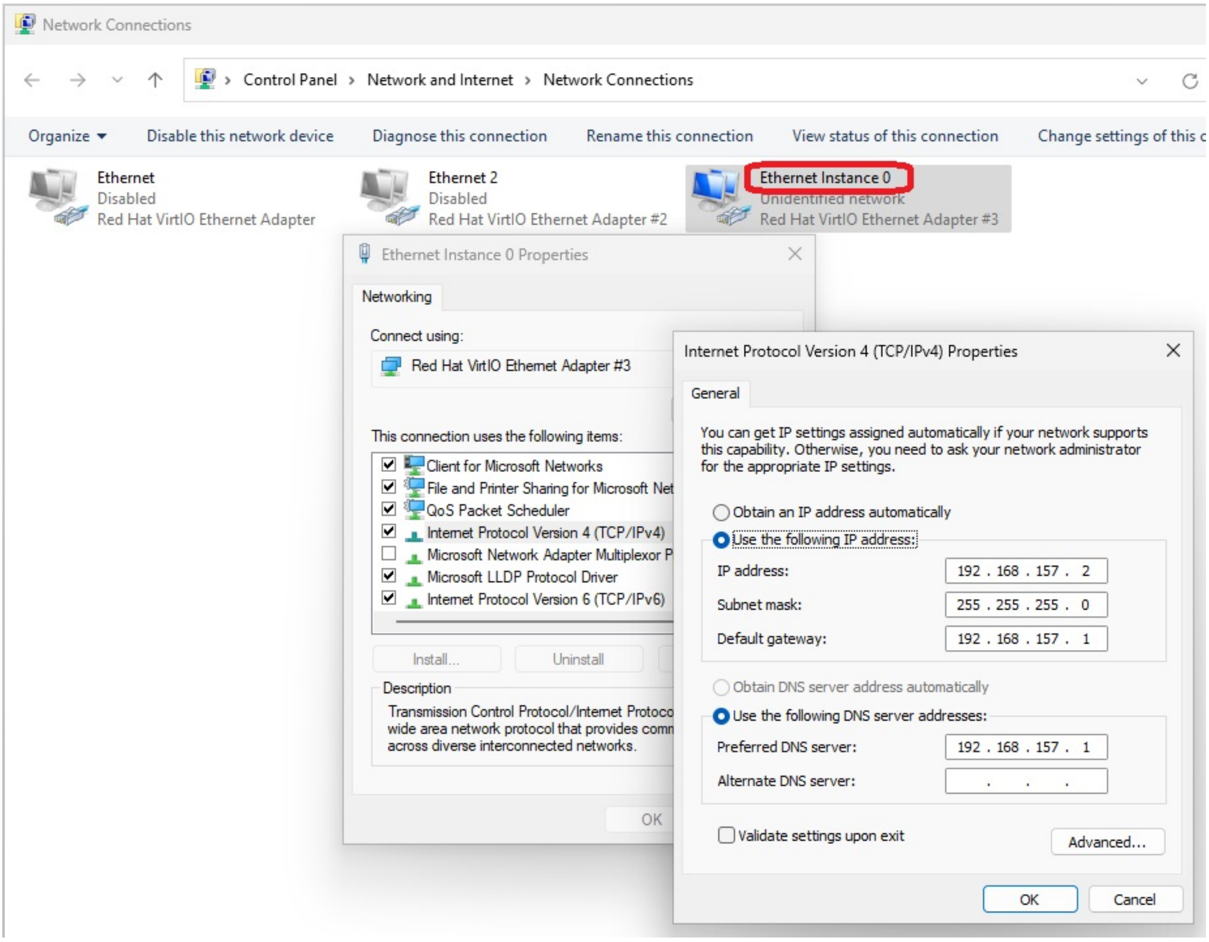
```
C:\Users\HVUSER>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : DESKTOP-PKQOSTL
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet Instance 0

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Red Hat VirtIO Ethernet Adapter #3
   Physical Address. . . . . . . . . : 0A-C1-C3-23-05-01
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::90d9:130b:185c:4762%10(Preferred)
   Autoconfiguration IPv4 Address. . : 169.254.143.231(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :
   DHCPv6 IAID . . . . . . . . . . . : 369803971
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2C-E8-DA-30-0A-C0-C3-23-05-01
   DNS Servers . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                       fec0:0:0:ffff::2%1
                                       fec0:0:0:ffff::3%1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```


Network Connections
Control Panel › Network and Internet › Network Connections
Organize ▾
Disable this network device
Diagnose this connection
Rename this connection
View status of this connection
Change settings of this c
Ethernet
Disabled
Red Hat VirtIO Ethernet Adapter
Ethernet 2
Disabled
Red Hat VirtIO Ethernet Adapter #2
Ethernet Instance 0
Unidentified network
Red Hat VirtIO Ethernet Adapter #3
Ethernet Instance 0 Properties
Networking
Install...
Uninstall
OK
Internet Protocol Version 4 (TCP/IPv4) Properties
General
Advanced...
OK
Cancel

Connect using:
Red Hat VirtIO Ethernet Adapter #3

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ File and Printer Sharing for Microsoft Net
- ☑ QoS Packet Scheduler
- ☑ Internet Protocol Version 4 (TCP/IPv4)
- ☐ Microsoft Network Adapter Multiplexor P
- ☑ Microsoft LLDP Protocol Driver
- ☑ Internet Protocol Version 6 (TCP/IPv6)

Description
Transmission Control Protocol/Internet Protoco
wide area network protocol that provides comn
across diverse interconnected networks.

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

IP address:	192 . 168 . 157 . 2
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 157 . 1

○ Obtain DNS server address automatically
● Use the following DNS server addresses:

Preferred DNS server:	192 . 168 . 157 . 1
Alternate DNS server:	. . .

☐ Validate settings upon exit

Caution: By default, the Windows firewall prevents responding to a ping from the Hypervisor Host. You must disable the firewall or set up an appropriate exception to be able to ping from the Hypervisor Host to the KVM guest.

7.6 Bridged virtual network connection (dynamic bridge)

If you want to communicate between a KVM guest and a RTOS guest, you can use the virtual network. This is either possible using the communication subsystem as described above. Alternatively, a dynamic bridge can be used, then avoiding the need to use the communication subsystem.

You need to set `vnet_nw` to a value of `1` to accomplish this.

The name of the dynamically created bridge will be `vmvnetbridge`.

Caution: If you have configured static bridges you have to remove all bridges before setting up the dynamic bridge:

```
$ hv_brdelall
```

To establish a connection from the KVM guest to the virtual network, an IP address must be assigned to the corresponding Ethernet adapter in the KVM guest.

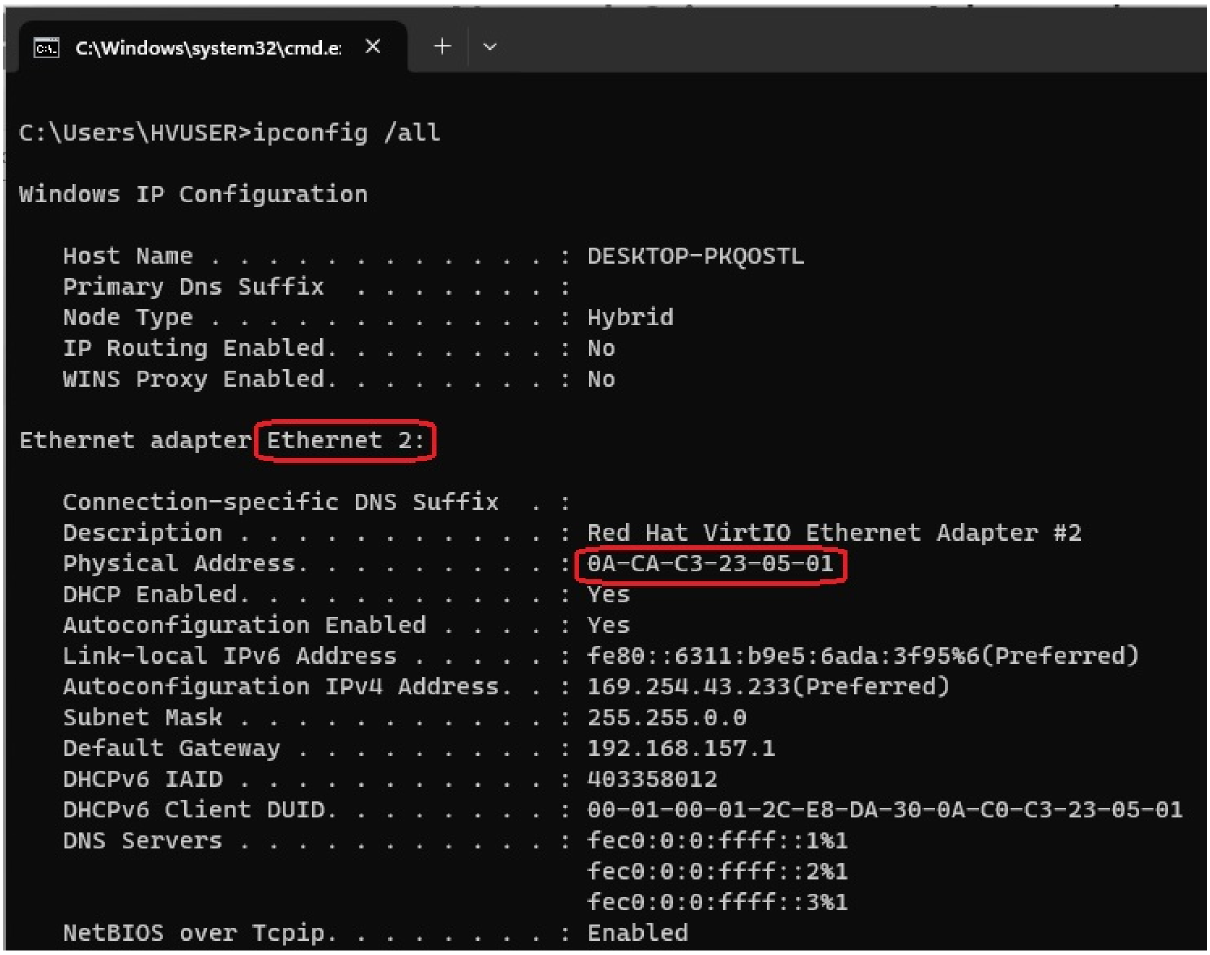
Here is an example for a Windows guest:

You will find the MAC address in the output when starting the KVM guest.

```
$ $ hv_guest_start -view
$ verify and prepare VM parameters
$ external network MAC = 0A:C0:C3:23:05:01
$ vnet network MAC = 0A:CA:C3:23:05:01     #---- virtual Network MAC needed
↪  for configuration
$ private network MAC = 0A:C1:C3:23:05:01
$ start VM
```

Use the `vnet network MAC` address for configuring the network card in the KVM guest. The corresponding network card can be identified by running `ipconfig /all` in the console. The network address is then set in the Windows `Network Connections`. By default, the address of the Hypervisor Host in the virtual network is 192.168.157.1. You can verify the IP Adress on the Hypervisor Host:

```
$ $ ip addr show vmvnetbridge
$    vmvnetbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
↪  state UP group default qlen 1000
$     link/ether 00:60:c8:00:00:00 brd ff:ff:ff:ff:ff:ff
$     inet 192.168.157.1/24 brd 192.168.157.255 scope global vmvnetbridge
```



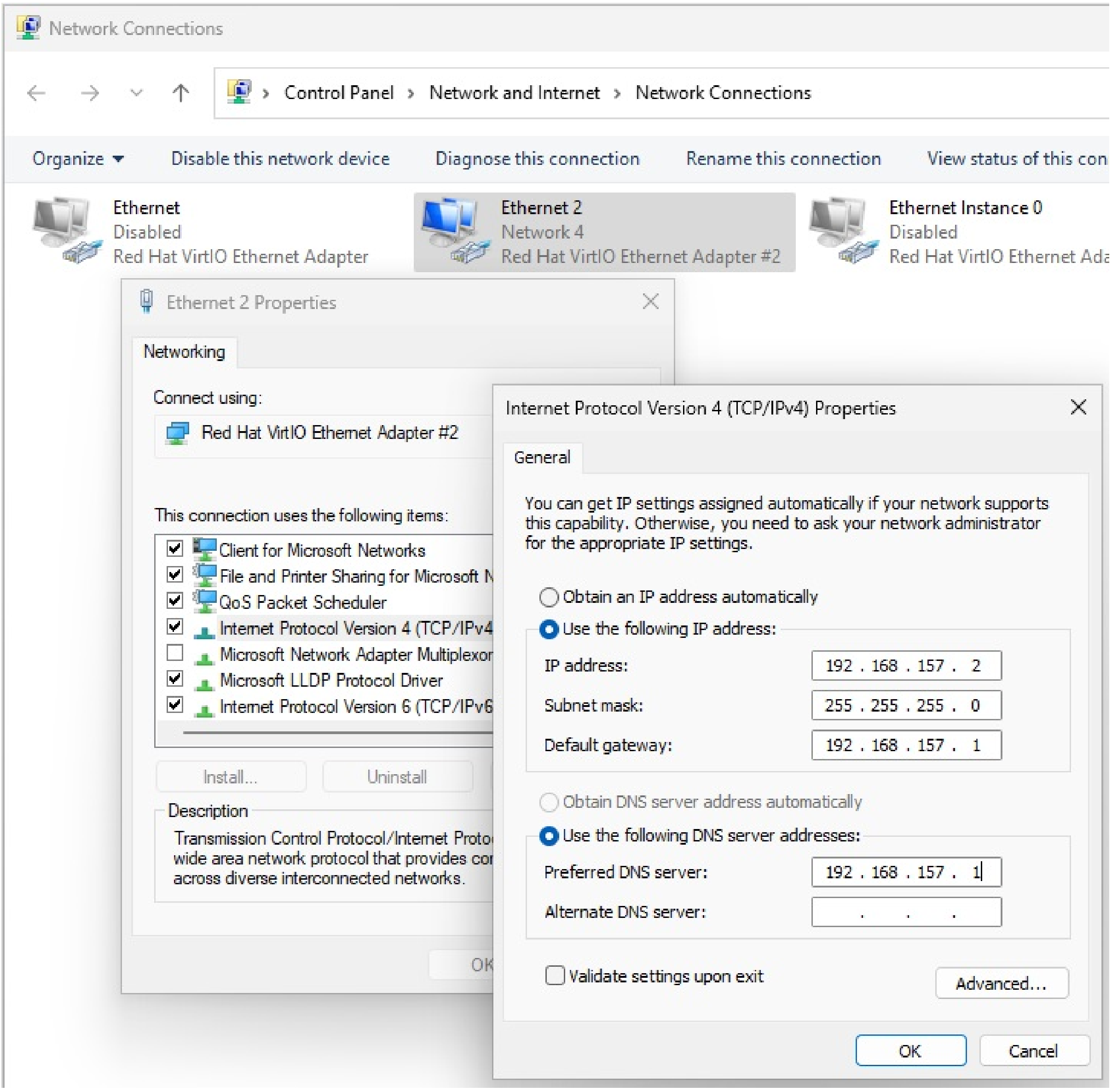
```
C:\Users\HVUSER>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : DESKTOP-PKQOSTL
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Red Hat VirtIO Ethernet Adapter #2
   Physical Address. . . . . . . . . : 0A-CA-C3-23-05-01
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::6311:b9e5:6ada:3f95%6(Preferred)
   Autoconfiguration IPv4 Address. . : 169.254.43.233(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 192.168.157.1
   DHCPv6 IAID . . . . . . . . . . . : 403358012
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2C-E8-DA-30-0A-C0-C3-23-05-01
   DNS Servers . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                       fec0:0:0:ffff::2%1
                                       fec0:0:0:ffff::3%1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```



Network Connections

Control Panel › Network and Internet › Network Connections

Organize ▾ | Disable this network device | Diagnose this connection | Rename this connection | View status of this con

Ethernet
Disabled
Red Hat VirtIO Ethernet Adapter

Ethernet 2
Network 4
Red Hat VirtIO Ethernet Adapter #2

Ethernet Instance 0
Disabled
Red Hat VirtIO Ethernet Ada

Ethernet 2 Properties

Networking

Connect using:

Red Hat VirtIO Ethernet Adapter #2

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ File and Printer Sharing for Microsoft N
- ☑ QoS Packet Scheduler
- ☑ Internet Protocol Version 4 (TCP/IPv4
- □ Microsoft Network Adapter Multiplexor
- ☑ Microsoft LLDP Protocol Driver
- ☑ Internet Protocol Version 6 (TCP/IPv6

Install... | Uninstall

Description

Transmission Control Protocol/Internet Proto
wide area network protocol that provides co
across diverse interconnected networks.

OK

Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 192 . 168 . 157 . 2

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 192 . 168 . 157 . 1

○ Obtain DNS server address automatically

● Use the following DNS server addresses:

Preferred DNS server: 192 . 168 . 157 . 1

Alternate DNS server: . . .

□ Validate settings upon exit

Advanced...

OK | Cancel

Caution: By default, the Windows firewall prevents responding to a ping from the Hypervisor Host. You must disable the firewall or set up an appropriate exception to be able to ping from the Hypervisor Host to the KVM guest.

7.7 KVM guest internal network (static bridge)

For guest internal TCP/IP communication it is possible to set up a guest internal network. This network has no connection to the outside physical network.

You need to set `guest_nw` to a value of `1` to enable this network. The Hypervisor Host may also be connected to this network by setting the `guest_br_ip` parameter to an appropriate IP address and network mask. In the default case, the Hypervisor Host is not connected to this network (`guest_br_ip` set to `no`).

The name of the statically created bridge will be `br-guest`.

7.8 Guest Ethernet MAC addresses

In the GUEST_FOLDER the script GUEST_NAME_setmac.sh defines the Ethernet MAC address for the virtual network adapters inside the guest. By default, if this file does not exist, this script file is automatically generated using random local administered addresses. You will have to change these addresses by an official address related to your company.

The content of this file looks as follows:

```
$ export ethmacVM1=0A:C0:3E:89:08:04
$ export ethmacVM2=0A:C1:3E:89:08:04
$ export ethmacVM3=0A:C2:3E:89:08:04
$ export ethmacVM4=0A:C3:3E:89:08:04
$ export vnetethmacVM1=0A:CA:3E:89:08:04
$ export ethmacPHYS
$ ethmacPHYS[0]=0A:D0:3E:89:08:04
$ ethmacPHYS[1]=0A:D1:3E:89:08:04
$ ethmacPHYS[2]=0A:D2:3E:89:08:04
$ ethmacPHYS[3]=0A:D3:3E:89:08:04
$ ethmacPHYS[4]=0A:D4:3E:89:08:04
$ ethmacPHYS[5]=0A:D5:3E:89:08:04
$ ethmacPHYS[6]=0A:D6:3E:89:08:04
$ ethmacPHYS[7]=0A:D7:3E:89:08:04
$ ethmacPHYS[8]=0A:D8:3E:89:08:04
$ ethmacPHYS[9]=0A:D9:3E:89:08:04
```

The ethmacVM1 address belongs to the bridged Ethernet controller (bridged to the external network).

The ethmacVM2 address belongs to the private (internal) Ethernet controller (using NAT).

The ethmacVM3 address belongs to the statically bridged Virtual network.

The ethmacVM4 address belongs to the guest internal network.

The vnetethmacVM1 address belongs to the dynamically bridged Virtual network.

The ethmacPHYS[#] addresses belong to the statically bridged external network (if MAC replication is turned off).

8 Host SDKs

Each part of the hypervisor supplies one or more SDKs, which can be used by custom-compiled applications. The SDKs can be categorized by Hypervisor Host, realtime guest and general purpose guest SDKs.

8.1 Hypervisor Host SDK

Applications, which are running at the **Hypervisor Host** and should communicate with the guests, needs the `Hypervisor Host SDK` as api.

The `Hypervisor Host SDK` is located at `/hv/sdk` folder. It contains currently an `/inc`, a `/lib` and an `/examples` sub-directory.

8.2 RTOS Guest SDK

- RT-Linux:
- VxWorks: [–]
- RTOS-32: [–]

8.3 Windows Guest SDK

The SDK's are available in the **Windows** guest **after** installing `RTOSVisor.exe`.

- Install `/hv/guests/files/RTOSVisor.exe`
- Location of the SDK directory: `%Program Files%\acontis_technologies\Hypervisor\SDK`

Hint: The supplied product `zip` package contains also this SDK at `/Hypervisor/SDK/...`.

8.4 Linux Guest SDK

The SDK's are available **after** installing the `hvUbuntuGuestSdkPackage_1.0-0_amd64.deb` package. This is done by calling the installation script `install_sdk.sh`. Both files are located on the Hypervisor Host in the directory `/hv/guests/files/LinuxTools`.

It is possible to install the SDK within the Linux guest or a separate development PC with Linux installed.

8.4.1 Install SDK in Linux guest

The installation is provided in a mountable directory on the Hypervisor Host. To access this directory from the Linux guest we need to install the network client software `cifs-utils` and mount the SMB drive from QEMU where the installation is located.

Open a `console` prompt in the Ubuntu guest and enter the following:

```
$ sudo apt-get update
$ sudo apt-get install cifs-utils
$ sudo mkdir /mnt/qemu
$ sudo mount -t cifs //10.0.2.4/qemu /mnt/qemu -o guest
$ sudo /mnt/qemu/files/LinuxTools/install_sdk.sh
```

With the installation process, the directory `/hv/sdk` is created. This is where all the essential files required for the SDK are stored.

8.4.2 Install SDK on a Linux development PC

To install the SDK on a separate development PC the `hvUbuntuGuestSdkPackage_1.0-0_amd64.deb` and `install_sdk.sh` files have to be copied from the Hypervisor Host `/hv/guests/files/LinuxTools` to the development PC. With calling the script `install_sdk.sh` the SDK will be installed in the directory `/hv/sdk`.

9 Automatic Startup

9.1 Hypervisor Host autologin

If you want to enable automatic login to the Hypervisor Host after booting has finished, follow the next steps.

```
$ sudo -e gedit
↪  /usr/share/lightdm/lightdm.conf.d/60-lightdm-gtk-greeter.conf
```

```
# autologin-settings to be appended
autologin-user=insert_your_username_here
autologin-user-timeout=0
```

9.2 Automatic guest startup

9.2.1 Enable autostart

In this section we will show how one or multiple guests can be started automatically after the Hypervisor has finished booting.

Introduction

To automatically start guests after the hypervisor finished booting, you need to adjust the /hv/config/usr_guest_autostart.config file.

```
$ gedit /hv/config/usr_guest_autostart.config
```

All guests that shall be started automatically have to be included here by inserting its respective guest foldername. The following example configuration will automatically start 4 guests, located in the folder guest0001 to guest0004.

```
/hv/guests/guest0001
/hv/guests/guest0002
/hv/guests/guest0003
/hv/guests/guest0004
```

The configuration settings will only become effective after running the command hv_set_autostart. This command requires the hypervisor root configuration file name as parameter. The root configuration file used by the System Manager is located in /hv/config/hv.config. The root configuration file used by the RT-Linux example guest configuration is located in /hv/guests/examples/rt-linux/hv.config.

```
$ hv_set_autostart -enable HYPERVISOR_ROOT_CONFIG_FILENAME
```

Hint:

This command will copy, rename and adjust the hv_guest_autostart.service template located in the guest folders into the /hv/services/hv_guest_autostart-GUEST.service (where GUEST is the guest foldername).

Then it will enable a respective `systemd` service located in `/etc/systemd/system`. This service will automatically start the guest after booting the system.

Caution: If you change the settings in `usr_guest_autostart.config` you need to first disable the autostart configuration and then re-enable the new configuration:

```
$ hv_set_autostart -disable
$ hv_set_autostart -enable HYPERVISOR_ROOT_CONFIG_FILENAME
```

Automatic startup of example guests (RT-Linux)

Uncomment the line `/hv/guests/examples/rt-linux show_console` in `usr_guest_autostart.config` to automatically configure starting the example RT-Linux guest. You may also uncomment the line `/hv/guests/examples/windows show_console` to automatically configure starting the example virtual KVM guest (e.g. Windows or Ubuntu).

```
;###################################################
; user specific guest autostart configuration script
;###################################################

; autostart example RT-Linux guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/examples/rt-linux show_console

; autostart example Windows guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/examples/windows show_console

; autostart example Ubuntu guest
; if guest console shall be shown automatically, use parameter show_console
; /hv/guests/examples/ubuntu show_console

; autostart system manager created guests (to be removed if system manager␣
↪supports autostart handling)
; if guest console shall be shown automatically, use parameter show_console
```

Then enable autostart:

```
$ hv_set_autostart -enable /hv/guests/examples/rt-linux/hv.config
```

Automatic startup of System Manager managed guests

Currently, the System Manager does not support setting autostart configuration from within the GUI. It is required to add additional lines `/hv/guests/guestXXXX show_console` in `usr_guest_autostart.config` to automatically configure starting System Manager generated guests.

```
;###################################################
; user specific guest autostart configuration script
;###################################################

```

(continues on next page)

(continued from previous page)

```
; autostart example RT-Linux guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/examples/rt-linux show_console

; autostart example Windows guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/examples/windows show_console

; autostart example Ubuntu guest
; if guest console shall be shown automatically, use parameter show_console
; /hv/guests/examples/ubuntu show_console

; autostart system manager created guests (to be removed if system manager␣
↪supports autostart handling)
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/guest0001
/hv/guests/guest0002
```

When enabling autostart, the /hv/config/hv.config file needs to be used:

```
$ hv_set_autostart -enable /hv/config/hv.config
```

Standard console mode

The show_console parameter following the guest foldername will automatically launch a console window to the guest when logging in into the Hypervisor Host.

The following example configuration will automatically start 3 guests, located in the folder guest0001 to guest0003. A console window will be displayed for guest 1 and 3.

```
/hv/guests/guest0001 show_console
/hv/guests/guest0002
/hv/guests/guest0003 show_console
```

Hint:

Here, the hv_guest_autostart_xfce.sh template will also be copied, renamed and adjusted into /hv/services/hv_guest_autostart_xfce-GUEST.sh.
When rebooting and after logging in, the Hypervisor will read the autostart.desktop file located in the home folder in ~/.config/autostart. This file points to a script file /hv/bin/xfce-hv_autostart.sh which launches all console windows that are configured in usr_guest_autostart.config.

Standalone console mode

The standalone_console parameter following the guest foldername will automatically launch a full screen console window to the guest when logging in into the Hypervisor Host.

The following example configuration will automatically start 3 guests, located in the folder guest0001 to guest0003. A full screen console window will be displayed for guest 3.

```
/hv/guests/guest0001
/hv/guests/guest0002
/hv/guests/guest0003 standalone_console
```

By default, the monitor/display is configured to never turn off.
This can be adjusted by editing the script
/hv/bin/guest_autostart_standalone_dispconf.sh and commenting out the respective commands.

```
# disable DPMS (Energy Star) features
# sudo xset -dpms >>$HV_BIN/guest_autostart_standalone_dispconf.log 2>&1

# turn off screen saver
# sudo xset s off >>$HV_BIN/guest_autostart_standalone_dispconf.log 2>&1
```

Caution: You should disable the Hypervisor Host desktop to assure the standalone console mode works properly. See section *Disable Hypervisor Host desktop* how to disable the Hypervisor Host desktop.

Caution: In the current RTOSVisor version, the standalone_console only works for the guest which is started last.

Caution: The standalone_console must not be used in conjunction with graphics passthrough.

Hint:

This command will copy, rename and adjust the hv_guest_autostart_standalone.service template located in the guest folders into the
/hv/services/hv_guest_autostart_standalone-GUEST.service (where GUEST is the guest foldername).
Then it will enable a respective systemd service located in /etc/systemd/system.
This service will automatically start the guest after booting the system.

Caution: Only one single KVM guest (no RTOS guest) can use the standalone console. Furthermore, if one guest is using the standalone console, no other guests can use the standard console anymore.

9.2.2 Disable autostart

To disable automatic start of guests, run the command

```
$ hv_set_autostart -disable
```

A complete reset of the autostart configuration can be done using the following command (it will not change the configuration settings in `usr_guest_autostart.config`):

```
$ hv_set_autostart -reset
```

9.2.3 Verify autostart

To show the current autostart configuration, run the command

```
$ hv_set_autostart -show
```

9.3 Hypervisor Host services

The hypervisor is using systemd services to automatically start guests, drivers etc. All services are located in the `/hv/services` folder. Active services will have a link set in `/etc/systemd/system` which point to the respective service file in `/hv/services`

The following services exist:

```
hv_sysmgr.service                - start the graphical System Manager web
 ↪  server backend
hv_loaddriver.service            - load the basic hypervisor drivers
hv_part.service                 - hypervisor partitioning, e.g. assigns PCI
 ↪  Ethernet Cards to a RTOS
hv_usbip_expose.service          - expose USB devices to Real-time guests
hv_vmf_autostart.service         - start the Virtual Machine Framework (the
 ↪  RTOS-VM Hypervisor) and load the hypervisor configuration
hv_guest_autostartXxxx.service   - start one specific guest
```

9.4 Autostart RT-Linux Applications

Automatic startup of RT-Linux applications is configured in the `guest.config` file located in the guest folder (section `[Rtos\Autostart\1]`). By default automatic start is enabled.

If this autostart section is active, the autostart script `autostart.sh` will be executed after RT-Linux has booted. This file is located in the `files` subfolder of the guest folder.

You may adjust both files according to your needs.

10 RTOS Devices (Partitioning)

10.1 Overview

By default, all devices are assigned for *non* Real-time guests. To assign a specific device to Real-time guests, the RTOSVisor has to be configured appropriately. For each device, a device assignment script has to be created which assigns such device to RTOS guests in general. In addition, a device configuration file for RTOS guests must be created which assigns such device to a specific RTOS guest.

Caution: The current script files only support device assignment to a single RTOS guest.

10.2 PCI/PCIe Devices

Hint: This section is valid for all kinds of PCI/PCIe devices. For assigning PCI/PCIe Ethernet devices you should follow the steps described in *PCI/PCIe Ethernet Devices*

10.2.1 Detect pci bus info

With `lspci` command a list of all available pci devices will be presented. The first number sequence (f.ex. `00:00.0`) is the main part of the desired `pci info`. The number sequence is the `vendor:device.function` information. An additonal `-D` parameter shows the domain number.

```
$ sudo lspci -k
```

Following an example output:

```
hvuser@HV-TP106:~$ sudo lspci -k
00:00.0 Host bridge: Intel Corporation Xeon E3-1200 Processor Family DRAM␣
↪Controller (rev 09)
        DeviceName:  Realtek 8211
        Subsystem: Fujitsu Technology Solutions Xeon E3-1200 Processor␣
↪Family DRAM Controller
        Kernel driver in use: ie31200_edac
        Kernel modules: ie31200_edac
00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network␣
↪Connection (Lewisville) (rev 04)
        Subsystem: Fujitsu Technology Solutions 82579LM Gigabit Network␣
↪Connection (Lewisville)
        Kernel driver in use: e1000e
        Kernel modules: e1000e
00:1a.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset␣
↪Family USB Enhanced Host Controller #2 (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family USB Enhanced Host Controller
        Kernel driver in use: ehci-pci
00:1c.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family␣
↪PCI Express Root Port 1 (rev b4)
```

(continues on next page)

(continued from previous page)

```
        Kernel driver in use: pcieport
00:1c.6 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family␣
↪PCI Express Root Port 7 (rev b4)
        Kernel driver in use: pcieport
00:1c.7 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family␣
↪PCI Express Root Port 8 (rev b4)
        Kernel driver in use: pcieport
00:1d.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset␣
↪Family USB Enhanced Host Controller #1 (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family USB Enhanced Host Controller
        Kernel driver in use: ehci-pci
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev a4)
00:1f.0 ISA bridge: Intel Corporation C202 Chipset LPC Controller (rev 04)
        Subsystem: Fujitsu Technology Solutions C202 Chipset LPC Controller
        Kernel driver in use: lpc_ich
        Kernel modules: lpc_ich
00:1f.2 SATA controller: Intel Corporation 6 Series/C200 Series Chipset␣
↪Family 6 port Desktop SATA AHCI Controller (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family 6 port Desktop SATA AHCI Controller
        Kernel driver in use: ahci
        Kernel modules: ahci
00:1f.3 SMBus: Intel Corporation 6 Series/C200 Series Chipset Family SMBus␣
↪Controller (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family SMBus Controller
        Kernel driver in use: i801_smbus
        Kernel modules: i2c_i801
02:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network␣
↪Connection
        Subsystem: Fujitsu Technology Solutions 82574L Gigabit Network␣
↪Connection
        Kernel driver in use: e1000e
        Kernel modules: e1000e
03:00.0 VGA compatible controller: Matrox Electronics Systems Ltd. MGA␣
↪G200e [Pilot] ServerEngines (SEP1) (rev 05)
        Subsystem: Fujitsu Technology Solutions MGA G200e [Pilot]␣
↪ServerEngines (SEP1)
        Kernel driver in use: mgag200
        Kernel modules: mgag200
```

Example output with additional `-D` parameter:

```
hvuser@HV-TP106:~$ sudo lspci -kD
0000:00:00.0 Host bridge: Intel Corporation Xeon E3-1200 Processor Family␣
↪DRAM Controller (rev 09)
        DeviceName:  Realtek 8211
        Subsystem: Fujitsu Technology Solutions Xeon E3-1200 Processor␣
↪Family DRAM Controller
        Kernel driver in use: ie31200_edac
        Kernel modules: ie31200_edac
0000:00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit␣
↪Network Connection (Lewisville) (rev 04)
        Subsystem: Fujitsu Technology Solutions 82579LM Gigabit Network␣
↪Connection (Lewisville)
```

(continues on next page)

(continued from previous page)

```
        Kernel driver in use: e1000e
        Kernel modules: e1000e
...
```

10.2.2 Case 1: PCIe Devices

In case a *non-ethernet PCI Express* (PCIe) device should be assigned to the real-time part, the supplied `hv_addpcidev` command should be used. The device `<bus info>` identified must be used as the first parameter of the `hv_addpcidev` command. The second (**mandatory**) parameter is a unique device name used on the Real-time side. The third (**mandatory**) parameter is a unique number (to be increased sequentially) for each assigned device, starting with `1`.

```
$ hv_addpcidev 0000:02:00.0 rtos_pcidev1 1
```

This call creates two files: `/hv/config/rtos_pcidev1.sh` and `/hv/config/rtos_pcidev1.config`.

In case additional devices shall be assigned:

```
$ hv_addpcidev [DETECTED_BUS_INFO] rtos_pcidev2 2
```

Hint: The RTOS device name is **mandatory**. This parameter **must** be a **unique** name that is used to identify the device. This name will also be used in `filenames` that are created by the `hv_addpcidev` command. In this tutorial the *default* name used is `rtos_pcidev1`.

Caution: If devices with **same** name are assigned to the **same** Rtos the names of the keys in `rtos_pcidev1.sh`, `rtos_pcidev1.config`, etc. **must** be **altered**!

10.2.3 Case 2: Legacy PCI Devices

In case a *non-ethernet Legacy* PCI device should be assigned to the real-time part, the method described in this section has to be applied (all other information described for *non-ethernet* PCIe devices are valid, though). Compared with *non-ethernet* PCIe devices, a fourth parameter of the `hv_addpcidev` command is required.

```
$ sudo hv_addpcidev [DETECTED_BUS_INFO] rtos_pcidev1 1 <interrupt type>
```

Possible values of `<interrupt type>` are **legacy** or **no_interrupt**. If this parameter is **not** provided, the default `MSI interrupt` is used which is **not** feasible for Legacy PCI devices.

10.3 PCI/PCIe Ethernet Devices

Hint: All commands to be executed in the following guide have to be input via the shell. To open the shell right click on the desktop and select *'Open Terminal here'* **or** press `CRTL` + `ALT` + `T`.

10.3.1 Device Identification

In a first step, it is required to determine the Ethernet device that shall be used by the Real-time geust. There are several ways how to detect the desired adapter.

Identify by hardware information

An easy way to identify an adapter is its hardware information:

```
$ lshw -class network
```

returns

```
*-network:1
    description: Ethernet interface
    product: 82545EM Gigabit Ethernet Controller (Copper)
    vendor: Intel Corporation
    physical id: 6
    bus info: pci@0000:02:06.0
    logical name: enp2s0
    version: 01
    serial: 00:0c:29:94:bb:c3
    size: 1Gbit/s
    capacity: 1Gbit/s
    width: 64 bits
    clock: 66MHz
    capabilities: bus_master cap_list rom ethernet physical logical tp␣
↪10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation
    configuration: autonegotiation=on broadcast=yes driver=e1000␣
↪driverversion=5.15.0-88-acontis duplex=full ip=172.17.10.26 latency=0␣
↪link=yes mingnt=255 multicast=yes port=twisted pair speed=1Gbit/s
    resources: irq:16 memory:fd580000-fd59ffff memory:fdfe0000-fdfeffff␣
↪ioport:2080(size=64) memory:fd520000-fd52ffff
```

We can see many information helping on identification: The network adapter `enp2s0` is an `'Intel'` type `'82545EM'` with MAC-ID `'00:0c:29:94:bb:c3'` and current link state `'link=yes'`.

Hint: If `RT-Linux` will be used as RTOS, than remember the currently used driver (in this example: `driver=e1000`), as *this* driver can **also** be used in `RT-Linux` with the *attached* device. This could be achived by a `modprobe [driver_name]` call in the rtos console window.

Identify by link-status change

Having identical devices the link status change can be used for identification. First unplug the cable of the desired Ethernet port and then list the current states:

```
$ ifconfig -a
```

```
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ether 00:0c:29:94:bb:c3  txqueuelen 1000  (Ethernet)
    RX packets 6063  bytes 705772 (705.7 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 292  bytes 62080 (62.0 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now you (re-)connect the port (to a live network) and list the states again.

```
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.17.10.26  netmask 255.255.0.0  broadcast 172.17.255.255
    inet6 fe80::28d1:40ed:dd73:b97a  prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:94:bb:c3  txqueuelen 1000  (Ethernet)
    RX packets 6125  bytes 717920 (717.9 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 350  bytes 71344 (71.3 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Both outputs are similar except the wanted adapter's `<link>` state has changed.

In this example the name of the network adapter is `enp2s0`. The can also be used to identify an adapter by its MAC-ID - in this case `00:0c:29:94:bb:c3`.

10.3.2 Case 1: PCIe Ethernet Devices

In case a *PCI Express* (PCIe) device is assigned, the method described in this section has to be applied. The device name identified above must be used as the first parameter of the `hv_addeth` command. The second (**mandatory**) parameter is a unique device name used on the Real-time side. The third (**mandatory**) parameter is a unique number (to be increased sequentially) for each assigned device, starting with `1`.

```
$ hv_addeth enp2s0 rtos_eth1 1
```

This call creates two files: `/hv/config/rtos_eth1.sh` and `/hv/config/rtos_eth1.config`.

In case additional network devices shall be assigned:

```
$ hv_addeth [DETECTED_NAME] rtos_eth2 2
```

Hint: The RTOS device name is **mandatory**. This parameter **must** be a **unique** name that is used to identify the device. This name will also be used in `filenames` that are created by the `hv_addeth` command. In this tutorial the *default* name used is `rtos_eth1`.

Caution: If devices with **same** name are assigned to the **same** Rtos the names of the keys in `rtos_eth1.sh`, `rtos_eth1.config`, etc. **must** be altered!

10.3.3 Case 2: Legacy PCI Ethernet Devices

In case a *Legacy* PCI device is assigned, the method described in this section has to be applied (all other information described for PCIe devices are valid, though). Compared with PCIe devices, a fourth parameter of the `hv_addeth` command is required.

```
$ sudo hv_addeth [DETECTED_NAME] rtos_eth 1 <interrupt type>
```

Possible values of `<interrupt type>` are **legacy** or **no_interrupt**. If this parameter is **not** provided, the default `MSI interrupt` is used which is **not** feasible for Legacy PCI devices.

10.4 Legacy Serial (COM) Devices (non PCI)

Run the following command to get an overview of the COM devices on your hardware.

```
$ sudo cat /proc/tty/driver/serial | grep -v "uart:unknown"
```

You will then get a list of devices, in the example below the hardware provides 2 serial ports, COM1 and COM2.

```
serinfo:1.0 driver revision:
0: uart:16550A port:000003F8 irq:4 tx:0 rx:0
1: uart:16550A port:000002F8 irq:3 tx:0 rx:0
```

The device names for the COM devices are ttyS0 (COM1), ttyS1 (COM2) etc. Before using a legacy serial device (COM1, COM2 etc.) in the RTOS, you need to unbind the Hypervisor Host driver. In a first step, determine the device numbers for the serial port, e.g. for COM1:

```
$ sudo tree /sys/class/tty/ttyS0/device/driver
```

You will then get the following information.

```
/sys/class/tty/ttyS0/device/driver
├── 00:02 -> ../../../../devices/pnp0/00:02
├── bind
├── uevent
└── unbind
```

The device numbers in the above example are “00:02”.

To unbind the driver for COM1, you need to run the following command.

```
$ echo -n "00:02" | sudo tee -a /sys/class/tty/ttyS0/device/driver/unbind
```

If you want to use the COM1 port in the Hypervisor Host again, unbind it.

```
$ echo -n "00:02" | sudo tee -a /sys/bus/pnp/drivers/serial/bind
```

You may add the following lines in `/hv/config/usr_hvpart.sh` to automatically bind/unbind the COM1 port:

```
[ "$cmd" == "add" ] && echo -n "00:02" | sudo tee -a /sys/class/tty/ttyS0/
↪device/driver/unbind
[ "$cmd" == "delete" ] && echo -n "00:02" | sudo tee -a /sys/bus/pnp/
↪drivers/serial/bind
```

10.5 RTOS assignment (general)

To assign a specific device to Real-time guests in general, the partitioning script `/hv/config/usr_hvpart.sh` must be adjusted.

The device assignment scripts `<RTOS device name>.sh` usually shall be executed **automatically** on system startup. To accomplish this, you need to add the respective `<RTOS device name>.sh` calls into the file `/hv/config/usr_hvpart.sh`. In our tutorial we use `rtos_eth1` as the unique `<RTOS device name>`, as mentioned earlier.

```
$ gedit /hv/config/usr_hvpart.sh
```

The `usr_hvpart.sh` file should contain at least the following string **after** editing: `source $HV_CONFIG/rtos_eth1.sh $cmd`

The example below shows how the device with the *unique* name `rtos_eth1` is assigned.

```
#!/bin/bash

cmd="add"
[ $1 == "delete" ] && cmd="delete"

# unbind devices (assign to RTOS)
source $HV_CONFIG/rtos_eth1.sh $cmd
```

Please run the `hv_hvpart` command with the parameter `add` or reboot the system to make the change effective.

```
$ hv_hvpart add
```

You may use the `delete` parameter to assign all RTOS devices back to the Hypervisor Host.

```
$ hv_hvpart delete
```

Hint: The `$HV_BIN/hvpart.sh` script will be automatically started via the *systemd* service controlled via `/etc/systemd/system/hv_part.service`. This script will call the `usr_hvpart.sh` script which includes user specific partitioning commands. This service can be *enabled* or *disabled* as shown below (by default, it is enabled)

```
$ sudo systemctl enable /hv/services/hv_part.service
$ sudo systemctl disable hv_part
```

Caution: The System Manager configuration tool will write the device assignment into `/hv/bin/hvpart.sh`, you must **not** change this file. Effectively, all devices listed in `/hv/bin/hvpart.sh` and `/hv/config/usr_hvpart.sh` will be assigned to the RTOS guests.

10.5.1 Verification

de-assignment check using Hypervisor Host

In a first step, check if the Hypervisor Host's *original* driver is **not** used in conjunction with the devices assigned to the Real-time guest.

```
$ lspci -k
```

The output will look similar like the following excerpt:

```
    :         :        :         :        :         :        :         :        :         :
    :         :        :         :        :         :        :         :        :         :
    :         :        :         :        :         :        :         :        :         :
01:00.0 Ethernet controller: Intel Corporation I210 Gigabit Network Connection (rev 03)
        Subsystem: Intel Corporation I210 Gigabit Network Connection
        Kernel driver in use: igb
        Kernel modules: igb
02:00.0 Ethernet controller: Intel Corporation I210 Gigabit Backplane Connection (rev 03)
        Subsystem: Intel Corporation I210 Gigabit Backplane Connection
        Kernel driver in use: pci-stub
        Kernel modules: igb
03:00.0 Ethernet controller: Intel Corporation I210 Gigabit Backplane Connection (rev 03)
        Subsystem: Intel Corporation I210 Gigabit Backplane Connection
        Kernel driver in use: pci-stub
        Kernel modules: igb
    :         :        :         :        :         :        :         :        :         :
    :         :        :         :        :         :        :         :        :         :
    :         :        :         :        :         :        :         :        :         :
```

In the above example, the instance `01:00.0` is used by Ubuntu (driver: *igb*, *e1000e* etc.) and the instances `02:00.0` and `03:00.0` are assigned to a Real-time guest (driver: *pci-stub*).

10.6 Assign a device to a specific Real-time guest

After creating the device configuration file `<RTOS device name>.config`, it needs to be included into the guest configuration file to become effective for the respective guest. The next steps describe how to add this device to the example Real-time Linux guest, we use `rtos_eth1` as the unique `<RTOS device name>`.

```
$ cd /hv/guests/examples/rt-linux
$ gedit ./usr.config
```

Add the corresponding `rtos_eth1.config` entry to the `includes` section of the configuration file.

The following example shows the 'modified' `usr.config` file:

```
RtosConfig
;------------------------------------------------------------------------
; acontis technologies GmbH
;
; Guest user configuration
;------------------------------------------------------------------------

#include "/hv/config/rtos_eth1.config"

;------------------------------------------------------------------------
; End of file
;------------------------------------------------------------------------
```

Please adjust the RTOS-32 guest configuration file as well because we will use an RTOS-32 application for device assignment verification later.

```
$ cd /hv/guests/examples/rtos-32
$ gedit ./usr.config
```

10.6.1 Real-time guest assignment check

Real-time Linux

After assigning a device to the RT-Linux a driver could be loaded for that device.

Run the rtos:

```
$ cd /hv/guests/examples/rt-linux
$ hv_guest_start -view
```

Log in into Real-Time Linux and run `modprobe [driver_name]`:

```
$ vmf64 login: root
$ password:  root
$ modprobe e1000
```

Hint: In the default case the *previously* used Hypervisor Host driver can **still** be used in RT-Linux.

RTOS-32

We use the `RTOS-32Demo` application to quickly verify, if the assigned netwok adapter is visible to the rtos part.

Adjust the guest configuration setting to prepare starting the *RTOS-32Demo*:

```
$ cd /hv/guests/examples/rtos-32
$ gedit usr_guest_config.sh
```

and add the following line:

```
export osImage=$HV_ROOT/guests/examples/rtos-32/Loader.bin
```

Adjust the link to the demo application:

```
$ cd /hv/guests/examples/rtos-32
$ rm rtos32app.dlm
$ ln -s /hv/guests/examples/rtos-32/files/RTOS-32Demo.dlm rtos32app.dlm
```

Run the demo:

```
$ cd /hv/guests/examples/rtos-32
$ hv_guest_start -view
```

The output should look like this for the used network adapter (`8086:1533`) in this tutorial:

```
PCI BIOS Information

Vendor Device Bus Dev Func Class  Int IRQ TLat MSI Type
-------------------------------------------------------
FFFE   FFFE     0   0   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0   2   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  20   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  22   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  26   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  27   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  28   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  29   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  30   0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  31   0  000000  -    0    0  -  Invalid class code
8086   1533     1   0   0  020000  A    5    0  X  Network controller
FFFE   FFFE     2   0   0  000000  -    0    0  -  Invalid class code
10EC   8169     3   5   0  020000  A    4   64  -  Network controller
Current Date/time: 01/01/2018 00:00:00
Current Date/time: 01/01/2018 00:00:01
Current Date/time: 01/01/2018 00:00:02
Current Date/time: 01/01/2018 00:00:03
Current Date/time: 01/01/2018 00:00:04

List of threads:
Name            Prio State      R.Del FStck  MStck Scheds  CTime  CT%
---------------------------------------------------------------------
Main Task          5 Current          63060  61276     11  -      -
Idle Task          0 Ready             2164   2164    526  -      -
Comm               5 Delaying       1 32832  31724    527  -      -
IPTASK             6 BlckdWait         4128   3940      2  -      -
IPTIMER            6 Delaying      11  4120   3940     44  -      -

Interrupts:
 IRQ    Calls  FreeStack  Doubles  Time
-------------------------------------------
   0      526        744        0  -

Network:
 Ping will respond at IP address 192.168.157.2
```

In the above example, one `8086:1533` device at `BDF 1,0,0` is assigned to the Real-time guest.

Finally, terminate the console connection to the real-time guest and stop the Real-time guest OS:

CTRL + C

```
$ cd /hv/guests/examples/rtos-32
$ hv_guest_stop
```

10.7 RTOS de-assignment

To de-assign a specific device from Real-time guests, the partitioning script `/hv/config/usr_hvpart.sh` must be adjusted.

In a first step, please de-assign **all** devices from RTOS guests using the `delete` parameter of the hv_hvpart command (assign all RTOS devices back to the Hypervisor Host):

```
$ hv_hvpart delete
```

Then you need to remove or uncomment the respective `<RTOS device name>.sh` calls from the file `/hv/config/usr_hvpart.sh`. Here we use `rtos_eth1` as the unique `<RTOS device name>`, as mentioned earlier.

```
$ gedit /hv/config/usr_hvpart.sh
```

The `usr_hvpart.sh` file should contain at least the following string **after** editing: `source $HV_CONFIG/rtos_eth1.sh $cmd`

The example below shows how the device with the *unique* name `rtos_eth1` is assigned.

```
#!/bin/bash

cmd="add"
[ $1 == "delete" ] && cmd="delete"

# unbind devices (assign to RTOS)
# source $HV_CONFIG/rtos_eth1.sh $cmd       --> uncomment or remove this line
```

Finally, run the `hv_hvpart` command with the parameter `add` or reboot the system to assign the remaining devices to RTOS guests again.

```
$ hv_hvpart add
```

11 Network

11.1 Virtual network

The hypervisor provides a virtual network. The Hypervisor Host and all the guests can be connected to this virtual network.

The IP addresses are set to fixed values.

The default IP addresses are:

Hypervisor Host: `192.168.157.1`

First RTOS: `192.168.157.2`

Windows example guest: `192.168.157.3`

Hint: The Hypervisor Host virtual network IP address is initially set when calling the `/hv/bin/inithv.sh` script. If the IP addresses shall be changed, the `hv_netconf` command can be used to adjust the Hypervisor Host IP address, for more details, see *Hypervisor Host network configuration*. The IP addresses of the RTOS and KVM guests have to be adjusted accordingly.

11.2 Network Forwarding from external computer to the RTOS

11.2.1 Hypervisor Host preparation

If the RTOS (or any other OS connected to the virtual network) shall be accessed via TCP/IP from a **single** external system, traffic can be forwarded to the virtual network. Execute the following steps to forward traffic from a specific external computer to the RTOS:

- enable network forwarding in the Hypervisor Host:

  ```
  $ sudo sysctl -w net.ipv4.ip_forward=1
  ```

- determine the IP address of the Hypervisor Host. You can use the `ifconfig` command to accomplish this.

Caution:

Assure the Default Gateway in the RTOS is set to the Hypervisor Host virtual network IP address `(192.168.157.1)!`

For RT-Linux it is set by default. For other RTOS you need to check the RTOS documentation how to accomplish this.

11.2.2 Forwarding from external Windows computer

- open a Command Window with Administrator rights on your Windows PC
- run the following command (replace AAA.BBB.CCC.DDD with the appropriate IP address of the Hypervisor Host):

```
route add 192.168.157.0 mask 255.255.255.0 AAA.BBB.CCC.DDD
```

11.2.3 Forwarding from external Linux computer

- open a Terminal Window on your Linux PC
- run the following command (replace AAA.BBB.CCC.DDD with the appropriate IP address of the Hypervisor Host):

```
ip route add 192.168.157.0/24 via AAA.BBB.CCC.DDD
```

11.3 Bridge virtual and physical network

If the RTOS (or any other OS connected to the virtual network) shall be accessed via TCP/IP from **any** external system, the virtual network and the respective physical network have to be bridged.

In the folder `/hv/config` you can find the template configuration file `brvnetconfig.sh` for the bridge configuration. Note, the IP address of the virtual network inside the RTOS guest need to be adjusted appropriately, see below for more details.

11.3.1 Bridge configuration

First step: determine, which network adapter should be bridged. Search for `<link>` entry and get the adapter name.

```
$ ifconfig -a
```

In this case it's `enp2s0`. The current `$IP$` address of `enp2s0` is `inet 172.17.10.53` and the network mask is `255.255.0.0`.

```
rtv@rtv-TEST:~$ ifconfig -a
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.17.10.53  netmask 255.255.0.0  broadcast 172.17.255.255
    inet6 2a02:590:801:2c00:7170:3747:f835:a1cb  prefixlen 64  scopeid 0x0
↪<global>
    inet6 fe80::fe6f:c5f8:c5cd:e3cd  prefixlen 64  scopeid 0x20<link>
    inet6 2a02:590:801:2c00:96b0:b8a:2c58:6c91  prefixlen 64  scopeid 0x0
↪<global>
    ether 90:1b:0e:18:c9:83  txqueuelen 1000  (Ethernet)
    RX packets 116751  bytes 22127837 (22.1 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 74453  bytes 551331072 (551.3 MB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp3s5: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
```

(continues on next page)

(continued from previous page)

```
        ether 74:ea:3a:81:4b:1d  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 201  bytes 14798 (14.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 201  bytes 14798 (14.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vnet0: flags=99<UP,BROADCAST,NOTRAILERS,RUNNING>  mtu 1500
        inet 192.168.157.1  netmask 255.255.255.0  broadcast 192.168.157.255
        ether 00:60:c8:00:00:00  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59  bytes 10381 (10.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Next step: determine the default gateway.

```
$ ip route ls
```

You will get an output like `default via 172.17.5.2 dev enp2s0 proto dhcp metric 100`.

Next step: determine the DNS server.

```
$ resolvectl status | grep "Current DNS Server"
```

You will get an output like `Current DNS Server: 172.17.5.9`.

Next step: Adjust `brvnetconfig.sh` with the detected values of `ifconfig`:

```
$ gedit /hv/config/brvnetconfig.sh
```

Values:

- `netif="enp2s0"`
- `defaultgw="172.17.5.2"`
- `dns="172.17.5.9"`
- `vnetbrip="172.17.10.53"`
- `vnetbrnm="255.255.0.0"`
- `#vnetbrmac=` comment in and adjust value only if there are collisions with **'same'** `MAC-IDs` on the network.

```
#!/bin/bash

# Ethernet network interface to bridge with VM.
# ethernet interface to bridge with vnet
netif="enp2s0"
```

(continues on next page)

(continued from previous page)

```
# Default gateway
# How to determine the default gateway:
#       Use the command ip route ls
#           default via 172.17.5.2 dev enp2s0 proto dhcp metric 100
#           172.17.0.0/16 dev enp2s0 proto kernel scope link src 172.17.
↪10.4 metric 100
#       The default gateway here is "172.17.5.2"
defaultgw="172.17.5.2"       # default gateway

# DNS server
# How to determine the default gateway:
#       Use the following command: resolvectl status | grep "Current DNS␣
↪Server"
#            Current DNS Server: 172.17.5.9
#       The DNS server here is "172.17.5.9"
dns="172.17.5.9"

# Bridge settings
# The bridge replaces the former network device used by the hypervisor to␣
↪connect to the network.
# See above results provided by the ifconfig -a command
#       enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
#               inet 172.17.10.53  netmask 255.255.0.0  broadcast 172.17.
↪255.255
#       In this example, the bridge IP address is 172.17.10.53 and the␣
↪network mask is 255.255.0.0
vnetbrip="172.17.10.53"
vnetbrnm="255.255.0.0"

# the values below are default values, typically not to be changed
vnetip="192.168.157.1"
vnetnm="255.255.255.0"
#vnetbrmac="54:52:00:ac:30:10       # by default, the MAC address of the␣
↪physical network is used
vnet="vnet0"
vnetbr="vnetbr"
```

RT-Linux Guest IP address settings

If the network is bridged, the IP address of RT-Linux must be adjusted properly. The settings are stored in `/hv/guests/examples/rt-linux/linux.config`. The IpAddress has to be set to a unique address in your company network, assure the entries are uncommented! The MacAddress has to be adjusted to a unique value only if more than one RT-Linux guest is bridged, in that case, please adjust the last value from `12` to `13`, `14` etc.

```
; This must be set correctly if the vnet device is bridged in the␣
↪Hypervisor Host
[Rtos\Vnet\0]
    "IpAddress"="172.17.10.239"
    "MacAddress"="AA:BB:CC:DD:E0:12"
```

Bridge activation

- After configuring the brige parameters, you can create the bridge:
  - First, start the RTOS to assure the virtual network is available.
  - Run the `hv_brvnetset` command.
- Remove the bridge - Run the `hv_brvnetclr` command.

11.4 Hypervisor Host network configuration

The Hypervisor Host network can be configured using automatic IP address configuration (DHCP), manual IP address configuration or disabled network. To simplify the process, the `hv_netconf` command is provided.

Caution:

The Hypervisor Host network configuration has to match with the KVM guest network settings (e.g. Windows or Ubuntu guest)
One of the guest network settings is determined by parameter netif_mode in the guest configuration file (e.g. `usr_guest_config.sh` located in the `GUEST_FOLDER`).
If the Hypervisor Host and guest settings do not match, the behaviour is undefined.

11.4.1 Automatic network configuration

This is the default mode. Nothing has to be changed if this mode shall be used. In case the networking had been adjusted manually, you can switch back to the automatic configuration as follows.

```
$ hv_netconf -auto
```

A single network interface can be set into automatic mode as follows.

```
$ hv_netconf %DEVICE% -auto
```

For example:

```
$ hv_netconf enp1s0 -auto
```

11.4.2 Manual network configuration

If you want the Hypervisor Host to be configured manually, you need to adjust the settings accordingly.

```
$ hv_netconf %DEVICE% -manual IP-address netmask-bits gateway-IP dns-IP
```

For example:

```
$ hv_netconf enp1s0 -manual 192.168.178.188 24 192.168.178.1 8.8.8.8
```

Then, configure the guest.

```
$ cd GUEST_FOLDER
$ gedit usr_guest_config.sh
```

Change the respective configuration values.

```
netif_mode=0
netif_m=...
defaultgw_m=...
dnsgw_m=...
brip_m=...
brnm_m=...
```

11.4.3 Disabled network

If you want the Hypervisor Host not to use the network, you need to adjust the settings accordingly. In case the PC currently is connected with the LAN and you want to use this connection for a guest, you need to determine the device name before disabling the Hypervisor Host network. You may use the `ifconfig` command for that purpose.

```
$ sudo ifconfig
```

Next, disable the network of the Hypervisor Host.

```
$ hv_netconf -off
```

You need to turn off IPv6 as follows.

```
$ sudo gedit /etc/sysctl.conf
```

Insert the following lines at the bottom of this file:

```
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Make this effective:

```
$ sudo sysctl -p
```

Then, configure the guest.

```
$ cd GUEST_FOLDER
$ gedit usr_guest_config.sh
```

Change the respective configuration values.

```
netif_mode=2
netif_m=...
```

Caution: You must define the network device that shall be used in the guest by setting the parameter `netif_m` to the name you have determined above.

Hint: To re-enable the Hypervisor Host network in automatic mode, run the `hv_netconf` command again:

```
$ hv_netconf -auto
```

And change the `netif_mode` configuration value.

```
$ cd GUEST_FOLDER
$ gedit usr_guest_config.sh
```

```
netif_mode=1
```

11.5 SMB (Windows) file share

If you want to exchange data between the hypervisor and external (Windows) computers, you may want to create a network share.

In a first step, the SAMBA package needs to be installed:

```
$ sudo apt-get install samba
```

Then you need to configure the SMB server properly. It is recommended to use the same username for the network share as you are using for the Hypervisor Host. To determine the user, you may run:

```
$ whoami
```

In this document we assume, the username is hvuser.

If you want use a different user for the SMB share, this user must also be configured for the Hypervisor Host. For example, to add a new user smbuser, run the following command:

```
    $ sudo adduser smbuser
```

To create a file share which is accessible from a remote Windows computer, go to the SAMBA configuration file:

```
$ sudo gedit /etc/samba/smb.conf
```

In the below example, we will create a share with the name guests ([guests]) which will share the folder /hv/guests.

Add the following section to the end of the smb.conf file and save:

```
[guests]
  comment = guests share
  path = /hv/guests
  browseable = yes
  valid users = hvuser
  guest ok = yes
  read only = no
```

If you encounter issues with the file share, you may also adjust the following section in the smb.conf file:

```
[global]
map to guest = never
```

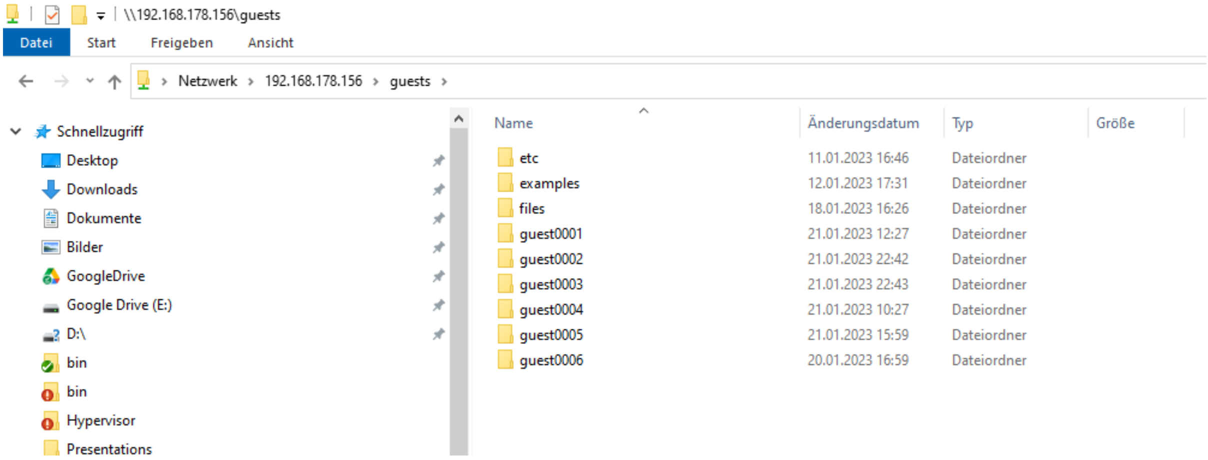
Then you need to provide network share access for the user hvuser:

```
$ sudo smbpasswd -a hvuser
```

Restart the SAMBA service:

```
$ sudo systemctl restart smbd.service nmbd.service
```

To check, if the share is active, try to access the SAMBA share from the Windows explorer. You may have to use the IP address of the hypervisor.



Name	Änderungsdatum	Typ	Größe
etc	11.01.2023 16:46	Dateiordner	
examples	12.01.2023 17:31	Dateiordner	
files	18.01.2023 16:26	Dateiordner	
guest0001	21.01.2023 12:27	Dateiordner	
guest0002	21.01.2023 22:42	Dateiordner	
guest0003	21.01.2023 22:43	Dateiordner	
guest0004	21.01.2023 10:27	Dateiordner	
guest0005	21.01.2023 15:59	Dateiordner	
guest0006	20.01.2023 16:59	Dateiordner	

Fig. 11.1: Access to share from Windows file explorer.

12 USB Devices

12.1 USB device access for RT-Linux guests

A single USB device can be accessed from within one single RT-Linux guest using usbip.
This will expose a USB device from the Hypervisor Host to the RT-Linux guest via the virtual network.

12.1.1 Hypervisor Host usbip package

In a first step, the respective software package for the RTOSVisor may have to be installed. Run the following command to check if the package is already available:

```
$ which usbipd
```

The package is available if you get the following result: `/usr/bin/usbipd`.

acontis linux kernel comes with preinstalled linux-extra- package, so it already includes usbip. If usbip package is not avaialble, please contact your technical support.

12.1.2 RT-Linux

The standard RT-Linux image shipped with the hypervisor does not support usbip. You need to exchange this image by the separately provided `rtlinux515.x64-usbip.bin` image file. This file is part of the CODESYS package and located in `/hv/guests/etc/rt-linux/files/codesys/rtlinux`. You need to adjust the configuration:

```
$ cd /hv/guests/examples/rt-linux
$ gedit guest_config.sh
```

Adjust the following line:

```
export osImage=$HV_ROOT/guests/etc/rt-linux/files/codesys/rtlinux/
↪rtlinux515.x64-usbip.bin
```

12.1.3 Prepare USB device exposal

The USB device which shall be exposed to RT-Linux needs to be prepared for exposal. In a first step you will need to determine the vendor and device ID of the USB device you want to expose. Insert the USB device and execute the following commands.

```
$ hv_hvusbip -init
$ hv_hvusbip -list
```

Hint: The call with the -init parameter is only required once.

A list of USB devices will be shown.

```
device list
***********
 - busid 3-2 (046d:c52b)
   Logitech, Inc. : Unifying Receiver (046d:c52b)

 - busid 3-3 (046d:c52f)
   Logitech, Inc. : Unifying Receiver (046d:c52f)

 - busid 3-5 (0e8d:0608)
   MediaTek Inc. : unknown product (0e8d:0608)

 - busid 3-8 (064f:2af9)
   WIBU-Systems AG : CmStick (HID, article no. 1001-xx-xxx) (064f:2af9)

 - busid 4-4 (18a5:0243)
   Verbatim, Ltd : Flash Drive (Store'n'Go) (18a5:0243)
```

In this example, we will expose the WIBU CmStick device to the guest.
According to the above output, its vendor device id is **064f:2af9**

Then you need to insert the vendor device ID into the respective configuration file.

```
$ cd /hv/config
$ gedit usbip_exposed_device.sh
```

Insert the appropriate vendor device ID.
Don’t miss to remove the comment at the beginning of the line!

```
# one single device currently can be exposed to the RTOS
export usb_vendev=064f:2af9   # WIBU CodeMeter USB dongle
```

Hint: After booting, the `hv_usbip_expose.service` located in `/hv/services` will prepare USB device exposal.

12.1.4 Expose USB device to the guest

You need to copy the `/hv/config/usbip_exposed_device.sh` into the respective guest folder to expose the USB device.

```
$ cd GUEST_FOLDER
$ cp /hv/config/usbip_exposed_device.sh .
```

Hint:

When starting RT-Linux, the `GUEST_FOLDER/usbip_gen_init.sh` script will dynamically create the `GUEST_FOLDER/files/usbip_init.sh` script.

This sub script will be called when RT-Linux is booted via the `GUEST_FOLDER/files/autostart.sh` script.

Caution: You must not adjust the `GUEST_FOLDER/files/usbip_init.sh` script manually, it will be overwritten when RT-Linux is started!

You may insert specific autostart activities when the RT-Linux guest starts.

```
$ cd GUEST_FOLDER
$ gedit usbip_gen_init.sh
```

At the very bottom the required autostart activity for a specific WIBU Codemeter USB device is shown. Please adjust the script `GUEST_FOLDER/usbip_gen_init.sh` according to your needs.

```
# insert usbip specific autostart activities here
if [ $usb_vendev == "064f:2af9" ]; then
    # echo "start codemeter daemon"
    echo "start-stop-daemon --start --quiet --chuid root --exec /usr/sbin/
↪CodeMeterLin" >>./rtfiles/usbip_init.sh
    echo "to verify CodeMeter status run 'cmu --cmdust'" >>./rtfiles/usbip_
↪init.sh
fi
```

Then, the usbip expose service needs to be enabled which will start the usbip server on the Hypervisor Host:

```
$ sudo systemctl enable /hv/services/hv_usbip_expose.service
```

Finally you should start the service to make your changes effective. You may also reboot to do so.

```
$ sudo systemctl start hv_usbip_expose
```

Hint:

This service will run the script /hv/config/usbip_expose.sh which will finally expose the device.

You can run the command 'systemctl status hv_usbip_expose.service' to verify if exposing worked correctly.

Caution:

If the USB device is removed and re-inserted, it has to be exposed again.

Furthermore, in RT-Linux the appropriate steps (executed in `usbip_init.sh`) also have to be executed again.

After you have started the service, you can check if the device is correctly exposed - before starting RT-Linux. Connect the USB device and then run the following command.

```
$ usbip list -r 127.0.0.1
```

A list of exportable USB devices will be shown. The one that you have set before should be shown.

```
Exportable USB devices
======================
 - 127.0.0.1
        3-3: WIBU-Systems AG : CmStick (HID, article no. 1001-xx-xxx)␣
↪(064f:2af9)
           : /sys/devices/pci0000:00/0000:00:14.0/usb3/3-3
           : (Defined at Interface level) (00/00/00)
           :  0 - Human Interface Device / No Subclass / None (03/00/00)
```

Finally you can start RT-Linux, the USB device should be visible then. To verify this, you can run the following command (inside RT-Linux!).

```
$ usbip port
```

A list of imported USB devices will be shown. The one that you have set before should be shown.

```
Imported USB devices
====================
Port 00: <Port in Use> at Full Speed(12Mbps)
       WIBU-Systems AG : CmStick (HID, article no. 1001-xx-xxx) (064f:2af9)
       1-1 -> usbip://192.168.157.1:3240/3-3
           -> remote bus/dev 003/007
```

Hint:

To stop exposing USB devices, you have to comment the device in `/hv/config/usbip_exposed_device.sh`.

```
# one single device currently can be exposed to the RTOS
# export usb_vendev=064f:2af9  # WIBU CodeMeter USB dongle
```

Then you should disable the service.

```
$ sudo systemctl disable hv_usbip_expose
```

12.1.5 Hypervisor Host usbip low level services

The usbip low level services are used by the scripts and services of the Hypervisor Host.
This background information may be helpful for debugging and diagnosis purposes.

Loading kernel modules:

```
$ sudo modprobe usbip_host
$ sudo modprobe usbip_core
```

Start the usbip daemon:

```
$ sudo usbipd -D
```

Show all USB connected devices:

```
$ usbip list -l
```

Expose a specific USB device:

```
$ sudo usbip bind -b busid
```

Show all currently exposed USB devices that are not in use:

```
$ usbip list -r 127.0.0.1
```

Stop exposing a specific USB device:

```
$ sudo usbip unbind -b busid
```

12.1.6 RT-Linux usbip low level services

The usbip low level services are used by the scripts and services of the RT-Linux guest.
This background information may be helpful for debugging and diagnosis purposes.

Loading kernel modules:

```
$ modprobe vhci-hcd
```

Show exposed devices:

```
$ usbip list -r 192.168.157.1
```

Attach exposed device:

```
$ usbip attach -r 192.168.157.1 -b "Device bus ID"
```

Show ports in use by attached devices:

```
$ usbip port
```

Detach a specific port:

```
$ usbip detach -p "port"
```

12.2 Windows/Linux USB guest access (non automatic mode)

In case the USB device is plugged in while the guest is already running, you need to determine some information about the USB device.

Using the lsusb command, a list of connected USB devices can be found:

```
lsusb
  Bus 002 Device 002: ID 0b95:1790 ASIX Electronics Corp. AX88179 Gigabit␣
↪Ethernet
  Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
  Bus 001 Device 003: ID 8087:0025 Intel Corp. Wireless-AC 9260 Bluetooth␣
↪Adapter
  Bus 001 Device 006: ID 064f:2af9 WIBU-Systems AG CmStick (HID, article␣
↪no. 1001-xx-xxx)
  Bus 001 Device 005: ID 046d:c050 Logitech, Inc. RX 250 Optical Mouse
  Bus 001 Device 004: ID 046a:0011 Cherry GmbH G83 (RS 6000) Keyboard
  Bus 001 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
  Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Then you need to run the hv_guest_monitor command.

```
$ cd GUEST_FOLDER
$ hv_guest_monitor
```

If you want to dynamically connect the first USB device (Bus 002, Device 002), theyp in the following command:

```
device_add usb-host,id=MyUsbDevice,hostbus=2,hostaddr=2
```

The id value can be selected freely but it has to be unique in case multiple USB devices are connected.

Hint: More information can be found here:

- https://github.com/qemu/qemu/blob/master/docs/usb2.txt
- https://unix.stackexchange.com/questions/426652/connect-to-running-qemu-instance-with-qemu-monitor/476617
- https://wiki.archlinux.de/title/QEMU#USB_Peripherie

12.3 Windows/Linux USB guest access (passthrough mode) for non Real-time guests

If a specific physical USB port shall be permanently used by the guest, this can be accomplished using USB passthrough mode.
In this case, the hypervisor will monitor a specific USB port and automatically passthrough a USB device (for ex. an USB 3.0 Stick) to an active guest.

In the first step, you need to determine the USB hostbus and hostport value pairs for the selected physical USB port.

Caution: Depending on the USB type (USB1/2 or USB3) there will be different values even if the same physical port is used.

Caution:

If a USB mouse is connected to a specific USB port, you must not passthrough this USB port unless you are using graphics passthrough mode.
This means, the physical USB port where such mouse is connected must not be used for other devices.
Also, the USB mouse must not be connected to any physical USB port which is passed through to the guest.
If this port is passed through, the mouse pointer will not be visible!

Hint: More information can be found here:

- https://qemu.weilnetz.de/doc/6.0/system/usb.html

- https://qemu-project.gitlab.io/qemu/system/devices/usb.html
- https://www.kraxel.org/blog/2018/08/qemu-usb-tips/

12.3.1 USB1/2 devices

Connect an USB2 stick to the USB port you want to passthrough and execute the following on the Hypervisor Host:

```
$ lsusb -t
```

You will get a result similar to

```
1  /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2  /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
3      |__ Port 4: Dev 18, If 0, Class=Mass Storage, Driver=usb-storage, 480M
4      |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
5      |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB hostbus value for this physical port using USB1/USB2 devices is 1 (see line 2)
The USB hostport value for this physical port using USB1/USB2 devices is 4 (see line 3)

12.3.2 USB3 devices

Connect an USB3 stick to the USB port you want to passthrough and execute the following on the Hypervisor Host:

```
$ lsusb -t
```

You will get a result similar to

```
1  /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2      |__ Port 5: Dev 18, If 0, Class=Mass Storage, Driver=usb-storage, 5000M
3  /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
4      |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
5      |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB hostbus value for this physical port using USB3 devices is 2 (see line 1)
The USB hostport value for this physical port using USB3 devices is 5 (see line 2)

12.3.3 USB hubs

If devices are connected behind an USB hub, you will see multiple nested ports behind which the device can be found.

Example 1: USB3 device behind a hub

```
1 /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2     |__ Port 1: Dev 15, If 0, Class=Hub, Driver=hub/4p, 5000M
3         |__ Port 2: Dev 16, If 0, Class=Vendor Specific Class, Driver=ax88179_178a, 5000M
4 /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
5     |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
6     |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB hostbus value for this physical port behind a hub is 2 (see line 1)

The USB hostport value for this physical port behind a hub is 1.2 (see lines 2 and 3)

Example 2: USB2 device behind two hubs

```
1 /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2 /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
3     |__ Port 1: Dev 26, If 0, Class=Hub, Driver=hub/4p, 480M
4         |__ Port 1: Dev 29, If 0, Class=Hub, Driver=hub/4p, 480M
5             |__ Port 2: Dev 30, If 0, Class=Mass Storage, Driver=usb-storage, 480M
6     |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
7     |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB hostbus value for this physical port behind a hub is 1 (see line 2)

The USB hostport value for this physical port behind a hub is 1.1.2 (see lines 3,4 and 5)

Caution: Some USB devices contain an internal USB hub to expose multiple USB device instances. Here, the same rules apply.

12.3.4 Guest configuration

The USB hostbus and hostport value pairs need to be used in the guest configuration file `usr_guest_config.sh`.

Below you will find the required entries for the above examples.

```
# USB host passthrough (automatic passthrough for any device connected to these ports).
#   Note: on the same physical USB port, different values for hostbus, hostport pairs will show up for different USB speed!
```

(continues on next page)

(continued from previous page)

```
export USB_HOST_ADAPTER1_PASSTHROUGH=""
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=1,hostport=4"
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=2,hostport=5"
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=2,hostport=1.2"
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=1,hostport=1.1.2"
```

Launch the guest and try to physically connect and disconnect a USB devices to the configured ports. The guest then should recognize such device automatically.

13 Graphics (Desktop) configuration

What graphical output the user will see can be configured according to the respective needs.
While during development showing the Hypervisor Host desktop (the Linux desktop) is very convenient this typically is not wanted when the system is shipped to customers.

13.1 Display both, Host and Guest Desktop

This is the default configuration after installation of the RTOSVisor.

13.2 Display Guest Desktop only

When shipping the system to customers you may want to only display the guest desktop in full screen (= kiosk) mode. If kiosk mode is enabled, the Windows (or Ubuntu) guest will run in full screen mode without the option to switch back to the Hypervisor Host desktop.

13.2.1 Guest autostart

In a production environment, the whole application should be started automatically after powering on the system.

- How to automatically start guests, see section *Automatic guest startup*.
- The section *Standalone console mode* describes how to automatically start one of the guests showing its desktop in full screen mode (standalone console).
- If you want to use guest autostart with Hypervisor Host desktop enabled, you must enable host autologin. The chapter *Hypervisor Host autologin* describes how to *autologin* into the Hypervisor Host in desktop mode.

13.2.2 Disable Hypervisor Host desktop

When using guest autostart with the standalone console (full screen desktop view of the guest), the Hypervisor Host desktop is no longer required and should be disabled. In this case the guest is also started faster after power on.

Execute the following command to configure the Hypervisor Host into a console-only mode.

```
$ sudo systemctl set-default multi-user.target
```

After rebooting `$ sudo reboot` and configuring guest autostart in standalone mode according to *Standalone console mode*, the guest should be started automatically in kiosk mode without showing the Hypervisor Host desktop.

Hint: If you want to **reactivate the Hypervisor Host desktop**, run

```
$ sudo systemctl set-default graphical.target
$ sudo reboot
```

If you want to **temporary start the Hypervisor Host desktop**, run

```
$ startx
```

It is even possible to completely **uninstall the Hypervisor Host GUI** to save disk space.
Please note, if the GUI is uninstalled, the interaction with the Hypervisor Host can be performed only via a SSH connection.
As an alternative, you may configure a separate Grub Menu Entry or Press Ctrl-Alt-F2 to switch to a tty2 console
The following command will completely uninstall the GUI component:

```
$ sudo apt-get purge lightdm
```

13.2.3 Monitor Mirroring (Windows guests)

If running a Windows guest in standalone console mode and more than one monitor is connected to the PC, by default the main display will be mirrored to all other displays. The display resolution will be set automatically. Depending on the connected monitors, it may be necessary to adjust the display resolution.

You will have to determine the display names and the related resolutions of the connected monitor.

```
$ export DISPLAY=:0
$ sudo xrandr
```

Below you can see a typical example where at HDMI-1 the main monitor is connected and at DP-1 the secondary one is connected.

```
$ Screen 0: minimum 320 x 200, current 1920 x 1080, maximum 16384 x 16384
$ HDMI-1 connected primary 1366x768+0+0 (normal left inverted right x axis y
↪  axis) 410mm x 230mm
$    1366x768      59.79*+
$    1920x1080     60.00    59.94
$    1280x1024     75.02    60.02
$    1280x720      60.00    59.94
$    1024x768      75.03    60.00
$    800x600       75.00    60.32
$    720x480       60.00    59.94
$    640x480       75.00    72.81    66.67    60.00    59.94
$    720x400       70.08
$ DP-1 connected 1920x1080+0+0 (normal left inverted right x axis y axis)
↪  531mm x 299mm
$    1920x1080     60.00*+
$    1680x1050     59.95
$    1280x1024     75.02    60.02
$    1280x960      60.00
$    1152x864      75.00
$    1024x768      75.03    60.00
$    832x624       74.55
$    800x600       75.00    60.32    56.25
$    640x480       75.00    59.94
$    720x400       70.08
```

If the resolution shall be adjusted, this can be set in `/hv/config/mirrormon.config`.

To set the resolution of the monitor connected at DP-1 to the same value as for HDMI-1, you may enter the following line into `mirrormon.config`:

```
DP-1:1366x768
```

Depending on the monitors connected, you may be able to set a resolution that the monitor originally did not support (in the above example, the monitor connected to DP-1 did not support 1366x768).

> **Caution:** You should use the same resolution on all connected monitors and the resolution should by default be supported by the monitor!

13.2.4 Monitor Switching (Linux/Debian guests)

Use case: A system which is configured to automatically start the guest in full-screen Kiosk mode after the Hypervisor Host boots. Two physical monitors are connected to the hardware with identical or different graphics resolution and **a single power source**. One of the two monitors is powered on when the VM starts. This monitor can be powered off while the second one is powered on (power switch), the guest display then will be automatically shown at the second monitor.

A template script, that has to be modified is located in `/hv/guests/etc/multi-mon/switchmon-1mon.py` When the script detects that one display is connected and the first one is disconnected, it executes commands on the guest VM. Therefore, it is necessary to configure the guest VM to allow the Hypervisor Host root user to execute commands via an SSH connection without a password. Make sure to establish an SSH key exchange between the Hypervisor Host and guest. If the guest has no external access and is configured for NAT Network, the Hypervisor Host should be set up for port forwarding.

Follow the steps below to configure this script:

- Install the spice-vdagent Debian package on the Linux VM.
- Save this script to a directory within the KVM guest, e.g., /hv/guests/guest0001.
- Make the script executable: chmod 755 multimon.py.
- Modify the VM_USER variable to a user in your VM. The Hypervisor Host will execute commands in the VM using this user.

Port Forwarding

This script assumes that the VM is isolated from the external network and operates through NAT, without any external IP address. If your VM has an IP address, you need to edit and adapt this script appropriately.

Port forwarding refers to the scenario where the Hypervisor Host connects to itself through a specific port, and this connection is then forwarded to a VM. To enable port forwarding, make the following changes to /hv/bin/kvm_guest.sh:

Locate the following lines:

```
if [ $private_nw -eq 1 ]; then
...
USERNET=$USERNET" -netdev user,id=networkusr,smb=$HV_ROOT/guests
```

Replace them with:

```
if [ $private_nw -eq 1 ]; then
...
USERNET=$USERNET" -netdev user,id=networkusr,smb=$HV_ROOT/guests,
↪hostfwd=tcp:127.0.0.0
```

Hypervisor Host - Guest SSH Key Exchange

To establish a secure connection between the Hypervisor Host and VM, it is necessary to exchange cryptographic keys. Follow these steps on the Hypervisor Host:

- Switch to the root user: sudo -i.
- Generate host keys: ssh-keygen -t rsa. When prompted to provide an “Enter passphrase,” press ENTER without entering a password.
- Copy the host key to the VM by executing the following commands:

```
ssh username@127.0.0.1 -p 8822 "mkdir -p .ssh"
cat .ssh/id_rsa.pub | ssh username@127.0.0.1 -p 8822 'cat >> .ssh/
↪authorized_keys'
```

- Replace “username” with the appropriate VM user.
- Test the connection to the VM using the following command:

```
ssh username@127.0.0.1 -p 8822
```

Verify

Perform the following steps to verify, if the script works properly:

- Start the VM.
- Execute the script: cd /hv/guests/guest0001 && sudo ./switchmon.py.
- Switch the power source and video cable from one monitor to another.
- The script’s output should resemble the following

```
2023-05-03 16:18:04,990 Switchmon started
2023-05-03 16:18:05,012 Active monitor: DP-2
2023-05-03 16:18:07,040 Active monitor resolution: 1920x1200
2023-05-03 16:18:27,194 Active monitor: NONE
2023-05-03 16:18:31,978 Active monitor: DP-2
2023-05-03 16:18:34,007 Active monitor resolution: 1920x1200
2023-05-03 16:18:40,165 Active monitor: NONE
2023-05-03 16:18:44,937 Active monitor: HDMI-1
2023-05-03 16:18:46,962 Active monitor resolution: 1280x1024
2023-05-03 16:18:49,075 Active monitor: NONE
2023-05-03 16:18:53,130 Active monitor: DP-2
2023-05-03 16:18:55,222 Active monitor resolution: 1920x1200
```

Automatic start

To start the script automatically in the background, make the following change to /hv/bin/kvm_start.sh:

Add /hv/guests/guest0001/multimon.py& before the remote-viewer line.

13.2.5 Suppress boot stage messages

Boot log

You may completely hide all logging messages while the computer boots. Adjust the files /etc/grub.d/40_custom and /etc/grub.d/41_custom. You need to edit the linux kernel boot line which includes the text “linux /boot/vmlinuz- “.

Add the following parameters to the bootline:

```
"quick splash console=ttyS0".
```

Save the file, update the grub menu and reboot.

```
$ sudo update-grub
$ sudo reboot
```

Login prompt

It is also possible to completely hide the login/password prompt that is displayed before the VM Guest window is shown in the kiosk mode. Execute the following command:

```
$ "systemctl disable getty@tty1.service"
```

and reboot.

You should see no login prompt anymore. The screen should remain black and then the VM Desktop should be displayed. If you wish to log in, press the hot key combination Ctrl-Shift-F2..F6.

Grub boot menu

If you want to suppress the grub boot menu, you need to edit the grub menu which is set in /etc/default/grub

```
$ sudo gedit /etc/default/grub
```

Adjust these entries:

```
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=3
```

Then you need to apply the changes:

```
$ sudo update-grub
$ sudo reboot
```

Then, during boot there will be a 3 seconds period where you have the chance to press the ESC key to show the boot menu if needed.

13.3 Guest Desktop via Pass-Through

Using graphics pass-through mode the integrated graphics hardware is assigned to the Windows guest and no longer available for the Hypervisor Host. This is described in the `Graphics Pass-Through Guide`.

14 Hypervisor Boot Customization

14.1 Splash Screen

To disable the “Xubuntu Splash Screen” when booting, please edit /etc/default/grub and update the bootloader.

```
$ cd /etc/default
$ sudo gedit grub
```

```
remove "splash quiet" from GRUB_CMDLINE_LINUX_DEFAULT
```

and then run

```
$ sudo update-grub
```

14.2 Brand Labeling

You can find some information about how to brand label the product *here*.

15 Clone an existing Installation

You may want to clone an existing RTOSVisor installation which is prepared for production. Clone tools like bugzilla (https://clonezilla.org/) can be used for such purpose.

Before cloneing the installation, you need to remove the following files in all KVM guest folders:

- dns_conf.sh
- *_setmac.sh
- all *.log and *.bak files

16 Remote Debugging

16.1 RT-Linux (Visual Studio)

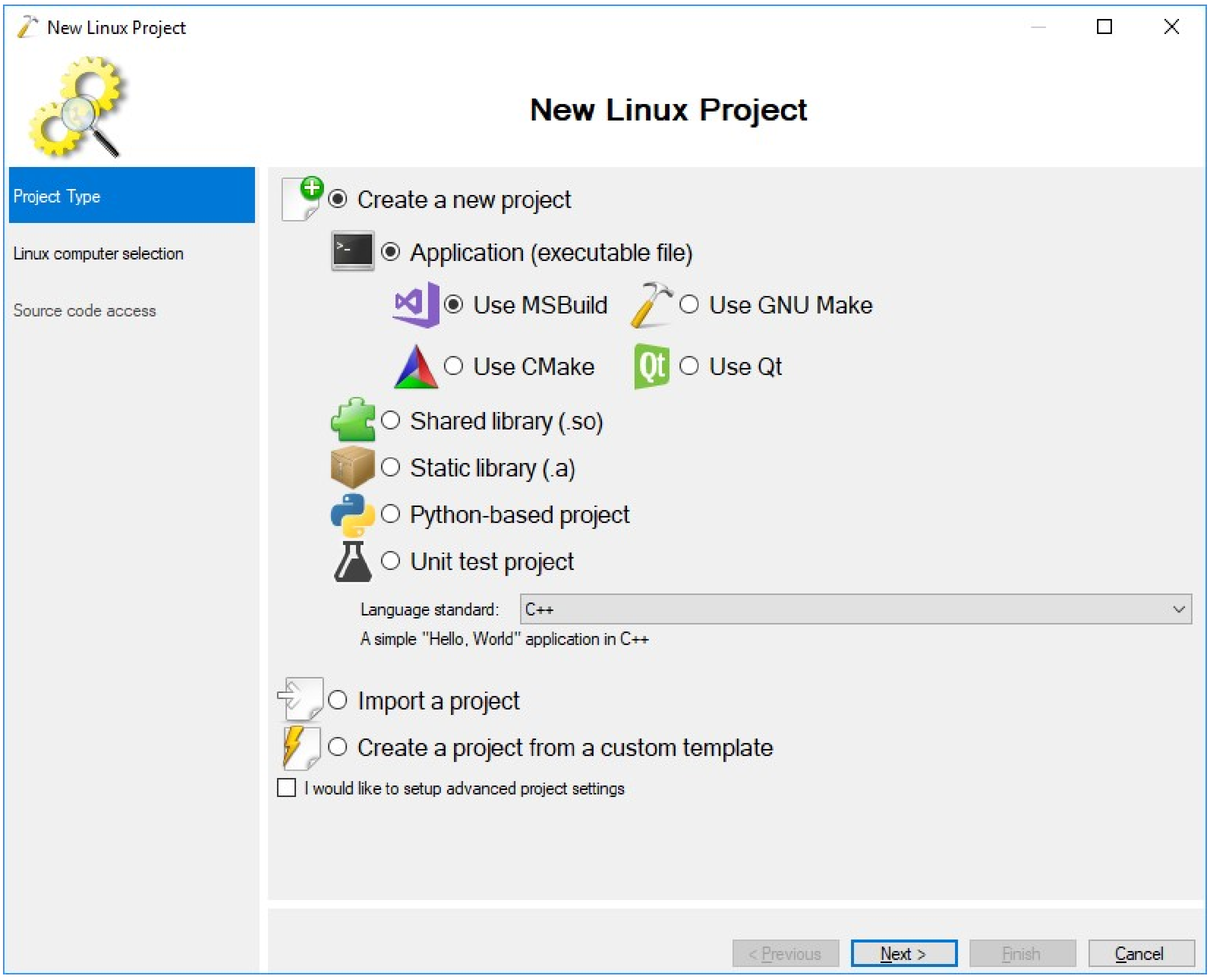
You may use the third party VisualGDB solution for development and debugging of RT-Linux applications using Microsoft Visual Studio.
An evaluation license can be obtained here: http://visualgdb.com/download

After installing VisualGDB restart Visual Studio to get the latest VisualGDB package updates.
The toolchain to be used can be downloaded from http://software.acontis.com/LxWin/MinGW.zip
For x64 development download the following toolchain
http://software.acontis.com/LxWin/MinGw64.zip

> Extract the zip file into `C:\` (If you want extract into another directory, ensure there are no blanks!)

16.1.1 Create a new project

- Start the Hypervisor Host, configure to run the Linux RTOS and start RT-Linux (this is described in the Quick Start Tutorial).
- Set up network bridging and assure you can reach the RT-Linux OS from your Windows development machine. See chapter *Bridge virtual and physical network* for details. Alternatively you can also use network forwarding, see *Network Forwarding from external computer to the RTOS*
- Start Visual Studio
- Create a new VisualGDB project by using the `Linux Project Wizard`
- Set up the project as Application and use MSBuild
- Set the `Language standard` to `C++`



New Linux Project

New Linux Project

● Create a new project

● Application (executable file)

● Use MSBuild ○ Use GNU Make

○ Use CMake ○ Use Qt

○ Shared library (.so)

○ Static library (.a)

○ Python-based project

○ Unit test project

Language standard: C++

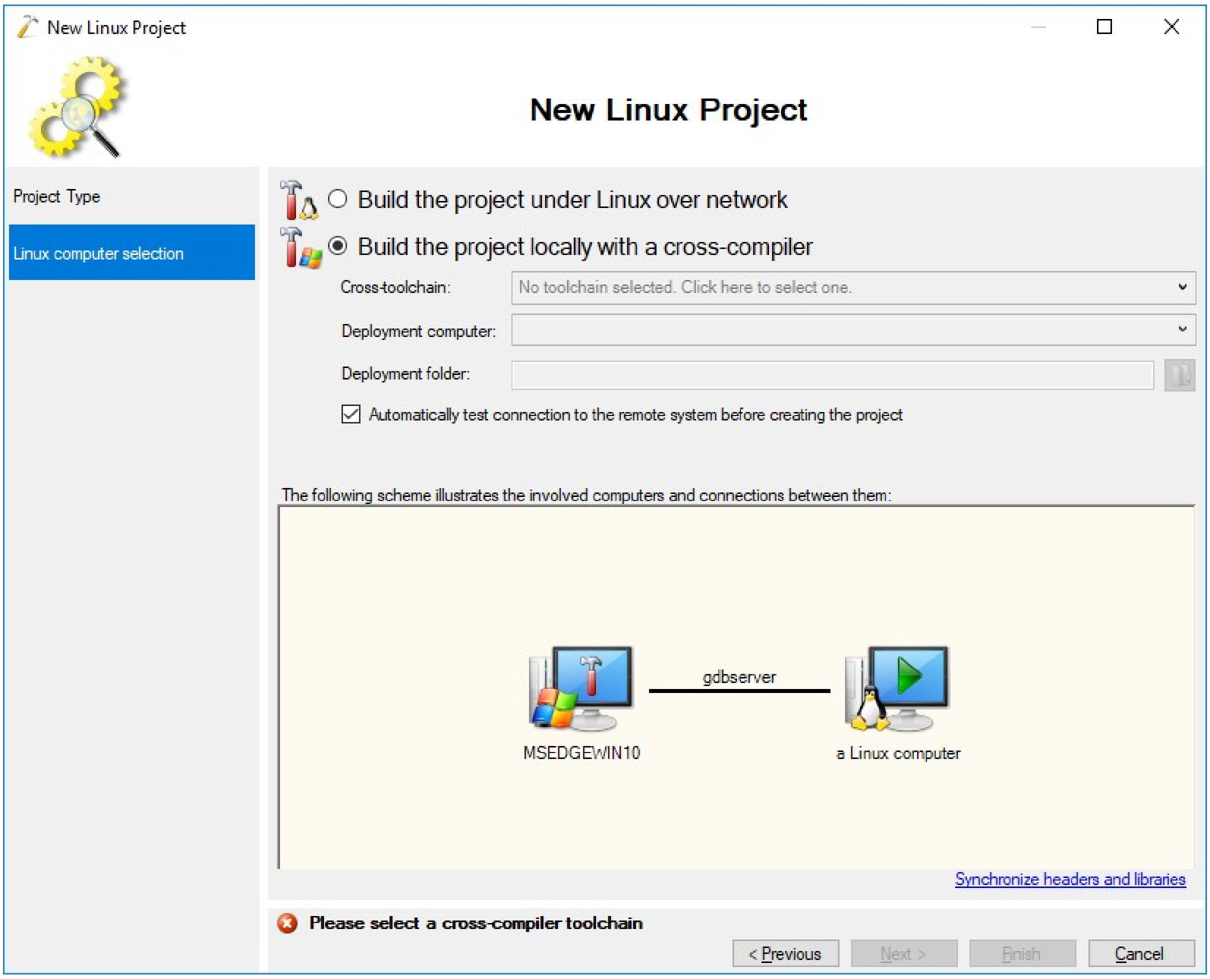
A simple "Hello, World" application in C++

○ Import a project

○ Create a project from a custom template

□ I would like to setup advanced project settings

- Select `Build the project locally with a cross-compiler`



New Linux Project

New Linux Project

○ Build the project under Linux over network

● Build the project locally with a cross-compiler

Cross-toolchain: No toolchain selected. Click here to select one.

Deployment computer:

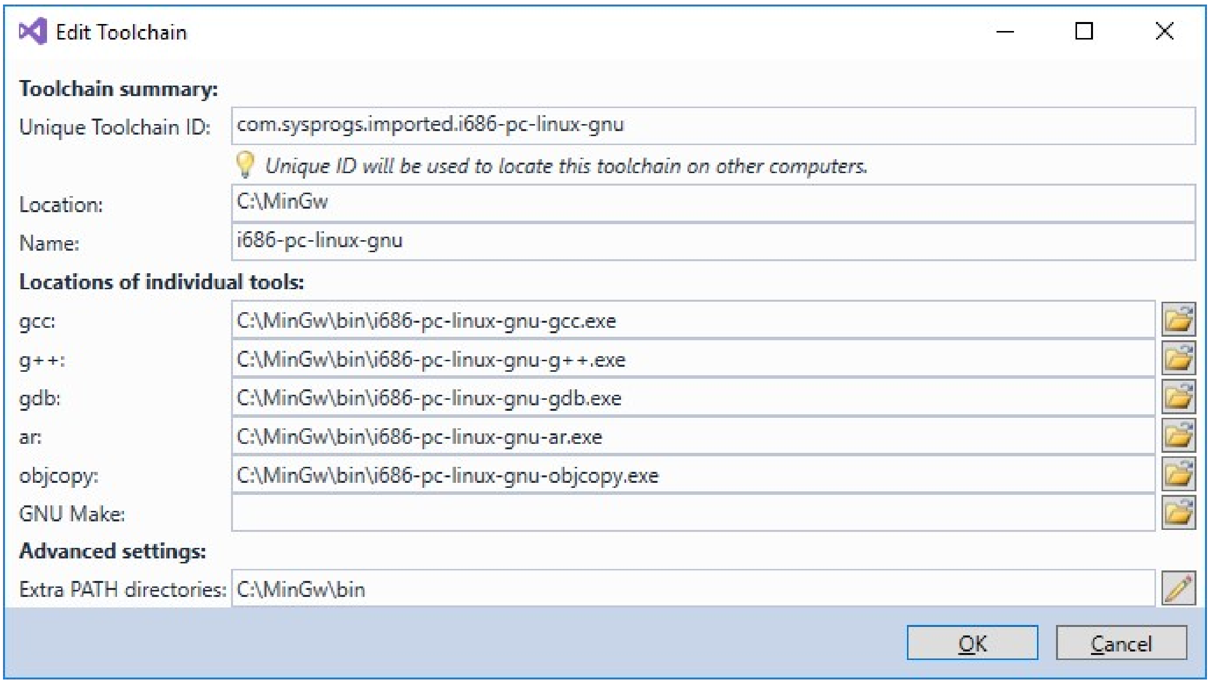
Deployment folder:

☑ Automatically test connection to the remote system before creating the project

The following scheme illustrates the involved computers and connections between them:

Please select a cross-compiler toolchain

- In the Cross-toolchain field select `Locate a cross-toolchain by finding its gdb.exe` and select `C:\MinGw\bin\i686-pc-linux-gnu-gdb.exe`
- For the 64 bit (x64) toolchain select `C:\MinGw64\bin\x86_64-pc-linux-gnu-gdb.exe`
- Edit the Toolchain dialog looks like:



Edit Toolchain

Toolchain summary:

Unique Toolchain ID: com.sysprogs.imported.i686-pc-linux-gnu

Unique ID will be used to locate this toolchain on other computers.

Location: C:\MinGw

Name: i686-pc-linux-gnu

Locations of individual tools:

gcc: C:\MinGw\bin\i686-pc-linux-gnu-gcc.exe

g++: C:\MinGw\bin\i686-pc-linux-gnu-g++.exe

gdb: C:\MinGw\bin\i686-pc-linux-gnu-gdb.exe

ar: C:\MinGw\bin\i686-pc-linux-gnu-ar.exe

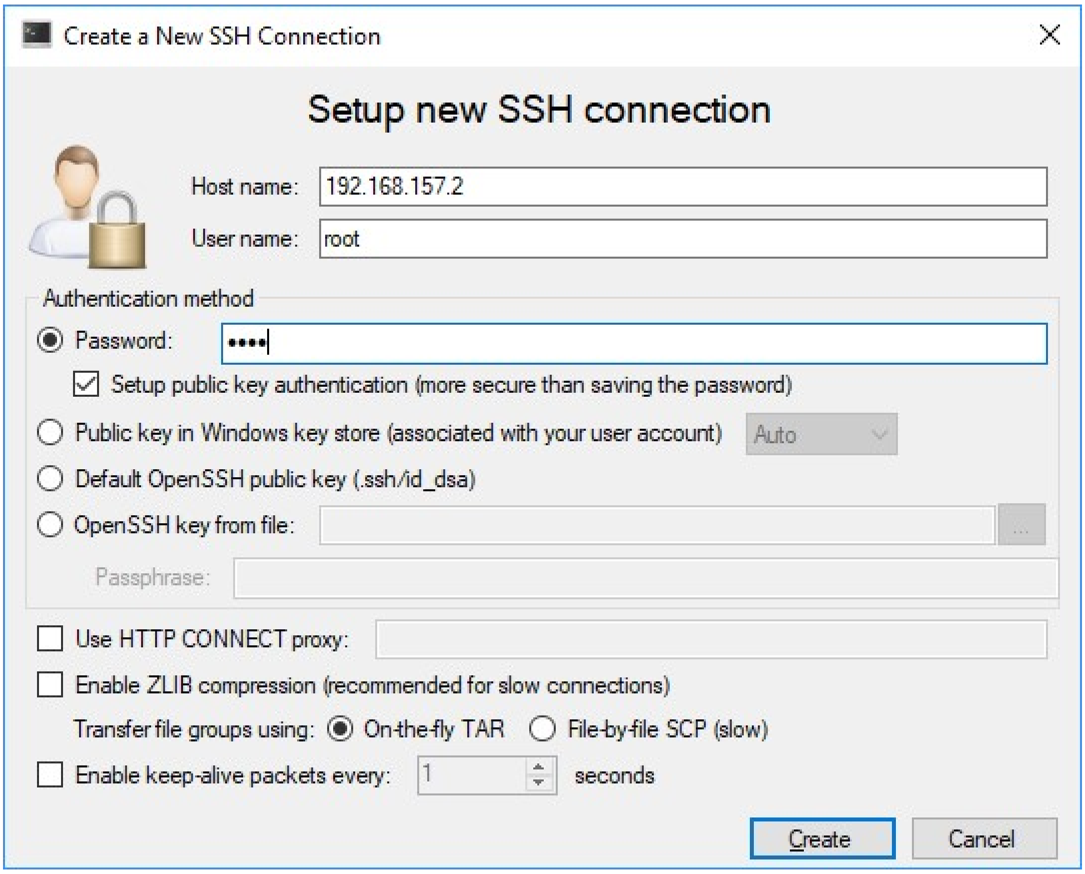
objcopy: C:\MinGw\bin\i686-pc-linux-gnu-objcopy.exe

GNU Make:

Advanced settings:

Extra PATH directories: C:\MinGw\bin

- In the `New Linux Project`-View, click the drop-down-field `Deployment computer` to create a new SSH connection
- Assure RT-Linux is started before you create the SSH connection! As host name use the IP address of the RT-Linux. User name and password are both `root`.



Create a New SSH Connection

Setup new SSH connection

Host name: 192.168.157.2

User name: root

Authentication method

● Password: ••••

☑ Setup public key authentication (more secure than saving the password)

○ Public key in Windows key store (associated with your user account) Auto

○ Default OpenSSH public key (.ssh/id_dsa)

○ OpenSSH key from file:

Passphrase:

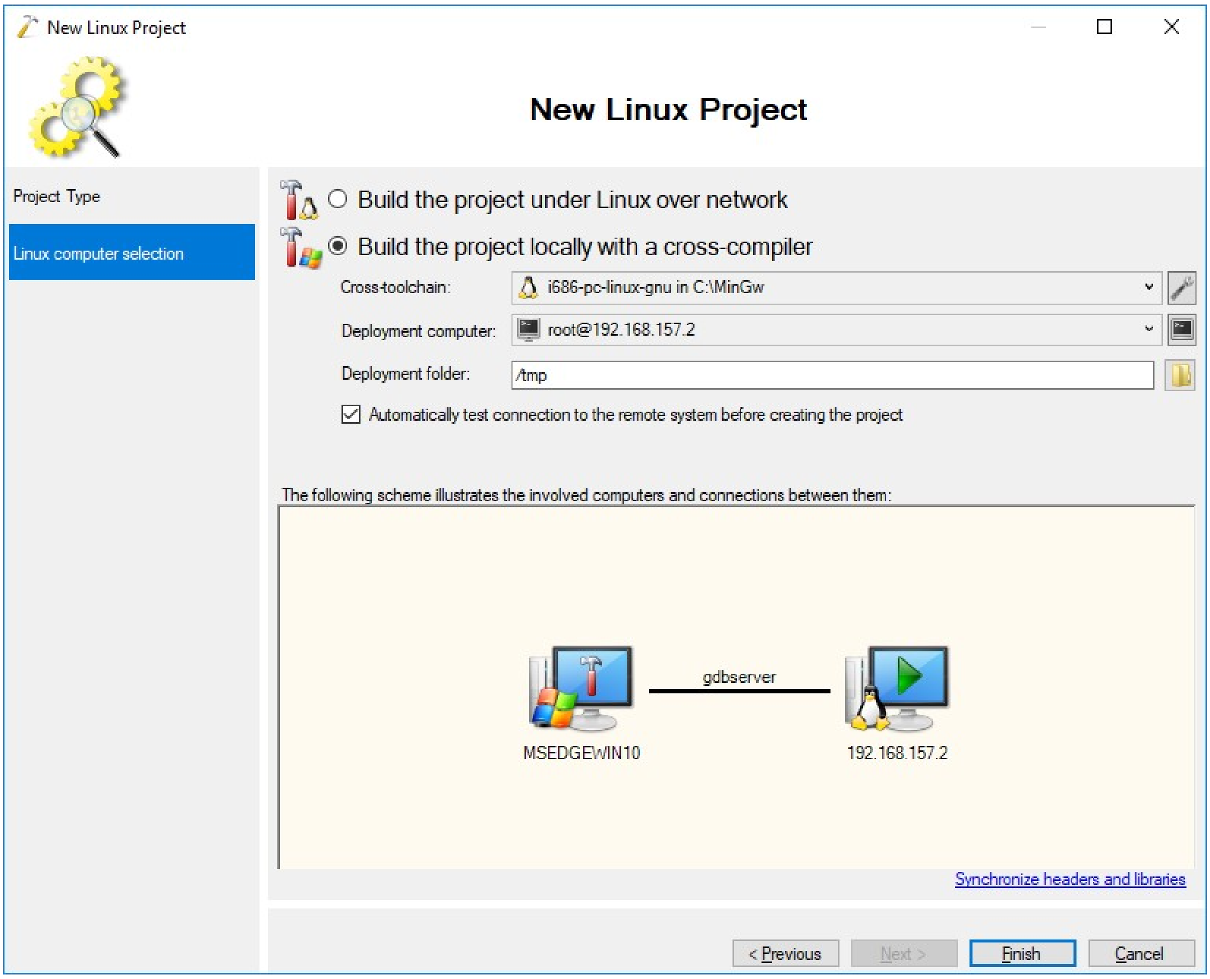
□ Use HTTP CONNECT proxy:

□ Enable ZLIB compression (recommended for slow connections)

Transfer file groups using: ● On-the-fly TAR ○ File-by-file SCP (slow)

□ Enable keep-alive packets every: 1 seconds

- Assure, before finishing the New Linux Project dialog looks like this:



New Linux Project

○ Build the project under Linux over network

● Build the project locally with a cross-compiler

Cross-toolchain: i686-pc-linux-gnu in C:\MinGw

Deployment computer: root@192.168.157.2

Deployment folder: /tmp

☑ Automatically test connection to the remote system before creating the project

The following scheme illustrates the involved computers and connections between them:

- Then press Finish. Accept the `Mismatching environment detected` message with the OK Button.
- Now you can debug the project

16.2 On Time RTOS-32

There are separate tutorials which describe how to set up remote debugging for On Time RTOS-32.

17 Miscellaneous

17.1 Error Report

In case of errors or technical questions, you may generate an error report which will collect various system informations. This error can be sent to the acontis technical support for further investigation.

Run the following command to create the error report:

```
$ hv_gen_error_report
```

17.2 KVM Guests with SecureBoot

17.2.1 UEFI and Legacy BIOS

17.2.2 UEFI

What is SecureBoot

Secure Boot is an interface between UEFI and Operating System. When SecureBoot is activated, it prevents the loading of unsigned boot loaders or drivers

Read More: https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface

Preparing keys for SecureBoot

The key thing in SecureBoot is a platform key (Platform). It establishes relationship between a platform owner and a platform firmware. PK is a self-generated certificated owned by OEM.

Another important key is a KEK key. This key is obtained from an OS manufacturer (for ex. Microsoft) and is used to establish trust relationship between the firmware and OS.

Generating the platform key:

```
openssl req -newkey rsa:2048 -nodes -keyout PKpriv.key -x509 -days 365 -
↪out PK.crt
Generating a 2048 bit RSA private key
....+++
.+++
writing new private key to 'PKpriv.key'
-----
You are about to be asked to enter information that will be incorporated␣
↪into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Bayern
```

(continues on next page)

(continued from previous page)

```
Locality Name (eg, city) []:Munic
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Beer Inc
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: BayernBeer
Email Address []:
```

OVMF supports keys in DER format only. So we need to convert t:

```
$ openssl x509 -in PK.crt -outform der -out PK.der
```

Download Key Exchange Key (KEK): `MicCorKEKCA2011_2011-06-24.crt` https://go.microsoft.com/fwlink/p/?linkid=321185

Download Signature Database (allows Windows to boot): `MicWinProPCA2011_2011-10-19.crt` https://go.microsoft.com/fwlink/?LinkId=321192

Download Microsoft signer for third party UEFI binaries via DevCenter: `MicCorUEFCA2011_2011-06-27.crt` https://go.microsoft.com/fwlink/p/?LinkID=321194

Building OVMF with SecureBoot support

By default, OVMF is built without SecureBoot support.

So it is recommended to fetch this project from its repository and build OVMF yourselves.

Install required packages:

```
$ sudo apt-get install build-essential git uuid-dev iasl nasm -y
$ sudo apt-get install iasl -y
$
$ git clone git://github.com/tianocore/edk2.git
$ cd edk2
```

Prepare build tools:

```
$ git submodule update -init
$ make -C BaseTools
$ .edksetup.sh
$ make -C ./BaseTools
$ export EDK_TOOLS_PATH=/home/rte/edk2/BaseTools
$ .edksetup.sh BaseTools
```

Edit `Conf/target.txt`:

```
ACTIVE_PLATFORM = OvmfPkg/OvmfPkgX64.dsc
TARGET_ARCH = X64
TOOL_CHAIN_TAG = GCC5
```

Build OVMF with SecureBoot support:

```
$ OvmfPkg/build.sh

    -a IA32 -a X64

    -D SMM_REQUIRE -D SECURE_BOOT_ENABLE

    -D FD_SIZE_2MB -D EXCLUDE_SHELL_FROM_FD
```

Binaries can be found in the `Build` directory.

Embedding SecureBoot keys to OVMF

Create a `OVMF-SecureBoot` directory and copy `Build/OvmfX64/DEBUG_GCC5/FV/OVMF_CODE.fd` and `Build/OvmfX64/DEBUG_GCC5/FV/OVMF_VARS.fd` to this directory.

Create a `hda` subdirectory and copy all generated and downloaded keys to this subdirectory.

Run qemu:

```
$ cd OVMF-SecureBoot
$ qemu-system-x86_64 -L .

    -drive if=pflash,format=raw,readonly,file=OVMF_CODE.fd

    -drive if=pflash,format=raw,file=OVMF_VARS.fd

    -hda fat:hda

    -net none
```

After booting you get to a UEFI shell. Type `exit`.

1. Go to Device Manager / Secure Boot Configuration / Secure Boot Mode and change from **Standard Mode** to **Custom Mode**.
2. PK Options / Enroll PK / Enroll PK Using File and choose `PK.der`
3. KEK Options / Enroll KEK / Enroll KEK Using File and choose `MicCorKEKCA2011_2011-06-24.crt`
4. DB Options / Enroll Signature / Enroll Signature Using File and choose `MicWinProPCA2011_2011-10-19.crt`
5. Repeat last step and choose `MicCorUEFCA2011_2011-06-27.crt`

The Secure Boot Mode should be **Enabled** now.

Exit from BIOS, shutdown the machine.

18 RTOS-32 Legacy information

Caution: This documentation is not valid for RTOSVisor V8.1!

The below documentation is an excerpt from RTOSVisor V8.0. Some of the files can be found in `/hv/guests/etc/rtos-32`.

18.1 On Time RTOS-32 container

- **Directories:**

  `/hv`
  this root directory contains all RTOSVisor files and executables.

  `/hv/rtos-32`
  RTOS-32 configuration files and start/stop scripts as well as OS binaries.

  `/hv/rtos-32/rtfiles`
  directory for your `.dlm` files.

- **Most important bash scripts:**

  `/hv/rtos-32/realtimedemo.sh`
  starts RTOS-32 VM and the acontis tool measuring context switch and interrupt latencies.

  `/hv/rtos-32/realtimedemo-debug.sh`
  starts debug monitor that awaits requests from a remote debugger on a Windows PC with installed EcWin, Visual Studio, and RTE Plugin.

  `/hv/rtos-32/ecmasterdemo.sh`
  starts RTOSVisor RTOS-32 VM and EtherCAT MasterStack demo.

  `hv_guest_stop`
  stops VM and RTOSVisor.

  `/hv/rtos-32/dbgcon.sh`
  opens interactive RTOS-32 VM console, so you can interact with your app here.

- **Specific command used for debugging:**

  `hv_brvnetset`
  creates a virtual network bridge on Hypervisor Host to forward debugger *TCP/IP/UDP* packets from `LAN1` to RTOS-32 VM. It is required to start this script if you need to perform remote debugging of a RTOS-32 app from another machine.

Hint: See chapter “Bridge virtual and physical network” in the Hypervisor Manual for details how to configure the bridge.

hv_vnetclr
deletes bridge, after the RTOS-32 VM has been stopped.

Start a RTOS-32 container using the `rtos-32.sh` script.

18.1.1 How to run a sample preconfigured RTOS-32 App (Realtimedemo)

RTOSVisor has a special tool (`RealtimeDemo`), which can accuratelly measure the real-time capabilities of a machine and the hypervisor. This tool can be also used as a first sample realtime app, to play with the RTOSVisor.

To start it, execute `/hv/rtos-32/realtimedemo.sh` script. This script loads/starts RTOSVisor VM, RTOS-32 OS Image and starts `RealtimeDemo.dlm`. Execute `hv_guest_stop` script to stop everything.

Most important parameters here are: *Interrupt Delay* and *Task Delay* (in microseconds). These parameters show the realtime capabilities of the Hypervisor Host and RTOSVisor.

You can press `Ctrl` + `C` to exit from the console into Linux Terminal. Please execute `dbgcon.sh` script to return to the RTOS-32 console again.

It is also easily possible to run the realtime demo in the background, completely detached from the console. So, use `dbgcon.sh` script in this case, to open interactive RTOS-32 terminal from a new console.

18.1.2 How to run `EcMasterDemo` that uses a network card and the EtherCAT stack

It is assumed that your Hypervisor Host has two ethernet ports `LAN1` and `LAN2`. The first one can be used for a *TCP/IP* traffic and/or to establish a debug connection with a DevPC.

It is also assumed, that your second `LAN` port was already assigned to RTOSVisor, as it is described in this `Hypervisor.pdf` in the **PCI/PCIe device assignment** section.

Execute the following steps:

1. Connect `LAN2` port to a EtherCAT slave.
2. `cd /hv/rtos32`
3. `sudo ./ecmasterdemo.sh`

All required files like `EcMasterDemo.dlm`, `EcMaster.dlm`, `emllRTL8169.dlm` are already in the `rtfiles/ecmaster` subdirectory.

18.1.3 How to remotely debug a RTOS-32 app from a Windows PC with Visual Studio

It is assumed, that you also have a second Windows PC (DevPC), where you installs Visual Studio + EcWin + RTE Plugin. DevPC is used to develop your RTOS-32 Application and perform a remote debugging of this app on the Hypervisor Host.

acontis has developed a special Visual Studio plugin that provides the possibility to create a RTOS-32 project, compile, debug it (local or remote) and configures a Visual Studio environment for you.

Requirements for a Windows PC:

* Visual Studio 2015 or newer should be installed

* acontis Rtos32Win/EcWin should be installed

Topics like creating a new RTOS-32 Project are out of scope of this document. Please refer to the official acontis RTOS-32 documentation to find details.

Before we start, it is important to understand, how RTOS-32 Apps work in the RTOS-32 VM. Every app consists of a Loader part (`/hv/rtos-32/Loader.bin`) and a dynamically loaded module in a `.dlm` format (`/hv/rtos-32/rtfiles/yourapp.dlm`). `Loader.bin` is provided with full source code and comes as a separate Visual Studio project. It is automatically generated by the RTE Visual Studio Plugin using the method described above.

But an application, that is is supposed to be debugged, should use a separate bootloader called **RTOS-32 Monitor** (`/hv/rtos-32/debug/Monvmf.bin`).

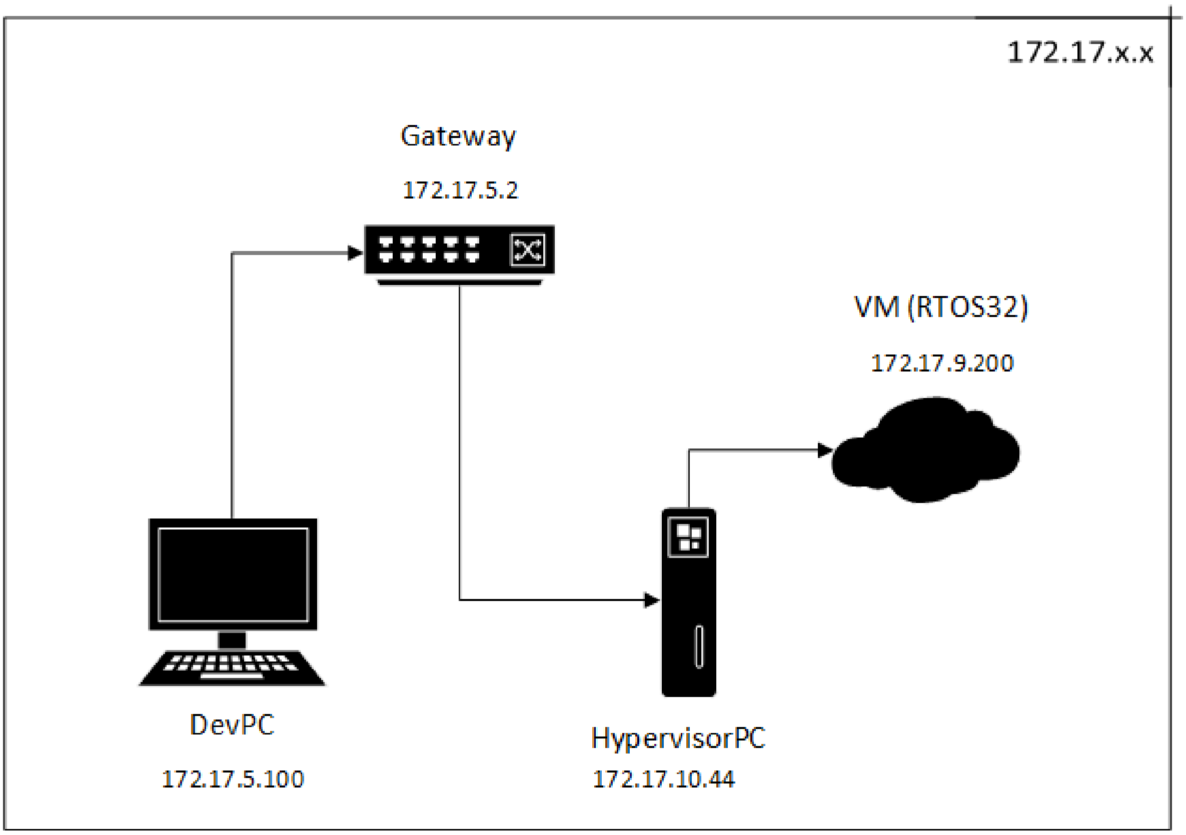
The monitor starts in RTOS-32 VM and listens to a *TCP/IP* traffic from the remote Visual Studio debugger.

The second important thing: all Monitor settings (like an *IP* address to listen to) are always embedded into the `Monvmf.bin` file. So, copy `Monvmf.bin` to the Hypervisor Host from DevPC every time when you change remote *IP* address in the System Manager (the part of EcWin/Rtos32Win).

Network Configuration:

The Hypervisor Host should be configured to peform the remote debugging; scripts and configuration files should contain specific *IP* addresses/masks/subnets. The requirement is that DevPC, Hypervisor Host and RtosVM are in the same subnet, in our example `172.17.x.x`.

Our sample network configuration:



All these *IP* addresses should be set here: `/hv/config/brvnetconfig.sh`

In our example, the gateway is a simple router with running DHCP server, that assigns static `172.17.5.\*` addresses to development PCs in the office and dynamic `172.17.10.\*` addresses to all other computers connected to the router.

VM with address `172.17.9.200` is in a separate subnet, not physically connected to the router. But Hypervisor Host accepts `MASK 172.17.255.255` on its virtual network bridge (read below) and can forward all traffic to and from a VM from the `172.17.x.x` network.

By simple words, the DevPC communicates always with VM. In the VM a special debug stub is started (called **RTOS-32 Debug Monitor**) that listents to the `172.17.9.200` *IP* address.

Open project of a sample demo app (we describe here how to debug `RealtimeDemo`):

1. DevPC: open System Manager and right click `RTOS #1\Application` and choose `Create New Application Project (Debug Only)`
2. Choose `RealtimeDemo` and click *OK* button. Visual Studio projects are generated now and copied to `RtFiles` and other subdirectories of your System Manager workspace.

Prepare a debugger bootloader (RTOS-32 Monitor):

1. DevPC: Select *Remote Debugging* and click *Settings* button near to a grayed *IP* text field.
2. Enter `172.17.9.200` in the **IP** text field and press *OK* button. Please note, now the new RTOS-32 Monitor binary (`Monvmf.bin`) is generated and the target *IP* addresss is embedded into the binary.

Compile the `.dlm` project:

1. DevPC: Click *Open Project with Visual Studio* button.
2. In the Visual Studio two projects are available in the solution explorer: `Loader` and `RealtimeDemo`.
3. Right click every project and click *Build*

Copy binaries to the Hypervisor Host:

1. Copy `RealtimeDemo.dlm` from the `RtFiles` directory in your workspace on DevPC to `/hv/rtos-32/rtfiles/debug/` directory.
2. Copy `projects/monvmf/Monvmf.bin` to `/hv/rtos-32/debug/` directory
3. Make sure `/hv/rtos-32/realtimedemo-debug.config` file has correct settings. It should have a valid file server directory (`hv/rtos-32/rtfiles/debug/`) and it should of course include `.config` file for a PCI Card for `LAN2` port, that is assigned to the RTOS VM.

Start RTOS-32 VM and the RTOS-32 Monitor (Hypervisor Host):

1. Execute `/hv/rtos-32/realtimedemo-debug.sh`. Please note, the `.dlm` module was already added to the `realtimedemo-debug.config` in the `Rtos\Loader` section.
2. You should see a RTOS-32 Monitor output. It listens to the IP address `172.17.9.200` to receive the remote debugger traffic.

Please check all your network setings here: `/hv/config/brvnetconfig.sh`

Please note, vnet0 virtual network adapter is created on Hypervisor Host, when the VM is started. Please consider this adapter as a virtual PCI network card inside the RTOS-32 VM, that is visible in Hypervisor Host as `vnet0`.

By default the `192.168.178.1` address is assigned to this adapter on a Linux side.

Configure a virtual bridge network adapter

Configure a virtual bridge network adapter to forward all incoming debugger traffic from `LAN1` to the virtual network adapter `vnet` in the RTOS VM.

Please execute `/hv/hvctl/brvnetset.sh` script. The virtbr network adapter is created on the linux side.

Start debugging (DevPC):

1. In Visual Studio open file `Loader.cpp`.
2. Locate `main()` function
3. Set a breakpoint.
4. Press `F5` key to start debug.

How to repeat/stop debugging:

1. There is no need to restart the RTOS-32 Monitor if you want to stop the debugging.

Simply click Stop in the Visual Studio and then press `F5` again.

1. It is not needed to copy the RTOS-32 Monitor binary (`Monvmf.bin`) every time you make changes in the loader or a `.dlm` project. Only when the *IP* address is changed.
2. **If you found a bug in your `.dlm` module and need to upload a new verison into the Hypervisor Host, please do the following:**

    1. compile project
    2. copy changed `.dlm` binary to the `/hv/rtos-32/rtfiles/debug/` directory.

3. **If you want completely stop your VM, make it in the following sequence:**

    1. Execute `/hv/hvctl/brvnetclr.sh` to remove the bridge
    2. Execute `hv_guest_stop`