acontis technologies GmbH

SOFTWARE

Hypervisor-Quickstart-Guide

acontis Real-time Hypervisor Quickstart

Version 1.0

Edition: November 3, 2022

Table of Contents

This Quickstart Guide shows how to install the acontis Hypervisor and how to run an EtherCAT demo application.

1 Getting Started - step-by-step tutorial

This manual will guide you through the following steps:

- Hypervisor Hardware/Software setup
- Assigning an `Intel PRO/1000` network adapter for real-time use (connected with EtherCAT slaves).
- Scan and configure the EtherCAT network.
- Bring the EtherCAT network into **OPERATIONAL** mode.

Hint: `sudo` is similar to Windows `run as administrator` and is valid for a single command only.

Hint: `sudo -s` switches the user to `root` (administrator), all following commands then will be executed with elevated rights.

1.1 Prerequisites

- PC/IPC with at least `1` Ethernet adapter.
  - CPU: at least `2` cores. This guide assumes a `4` cores CPU is used.
  - HDD: at least `10GB`. Additional `50GB` in case a Windows guest shall be used.
  - One Ethernet adapter to be used for TCP/IP (e.g. to remotely connect to the Hypervisor). This adapter has to be connected with you company network.
  - One additional `Intel Gigabit` Ethernet adapter (like `I210`), as some of the shipped demo scripts are preconfigured to this type of network adapter.
- Several EtherCAT slaves connected with the `Intel Gigabit` network adapter

Caution: You must disable Secure boot in the BIOS, otherwise the Hypervisor will not work correctly!

Caution: In this tutorial the Hypervisor will be **installed** onto the hard disk and **all** data will be **overwritten**. Assure required data **is saved** before installation!

- Recommended BIOS settings (for real-time operation)
  - USB Legacy mode has to be *disabled*
  - Hyper-Threading shall be *disabled*
  - Power saving settings have to be *disabled* (C-States, Speedstep, …)

2 Installation and basic configuration

The Hypervisor has to be installed onto an empty installation media. It is directly booted from the BIOS. Side-by-side installation with an existing Windows or Linux is not recommended, but possible. If an existing Windows or Linux system shall be kept, assure a free partition for booting the Hypervisor is available. The minimum required size for the installation media is 10 GByte.

2.1 USB stick boot media creation

You may use the free Rufus tool to create the boot media.

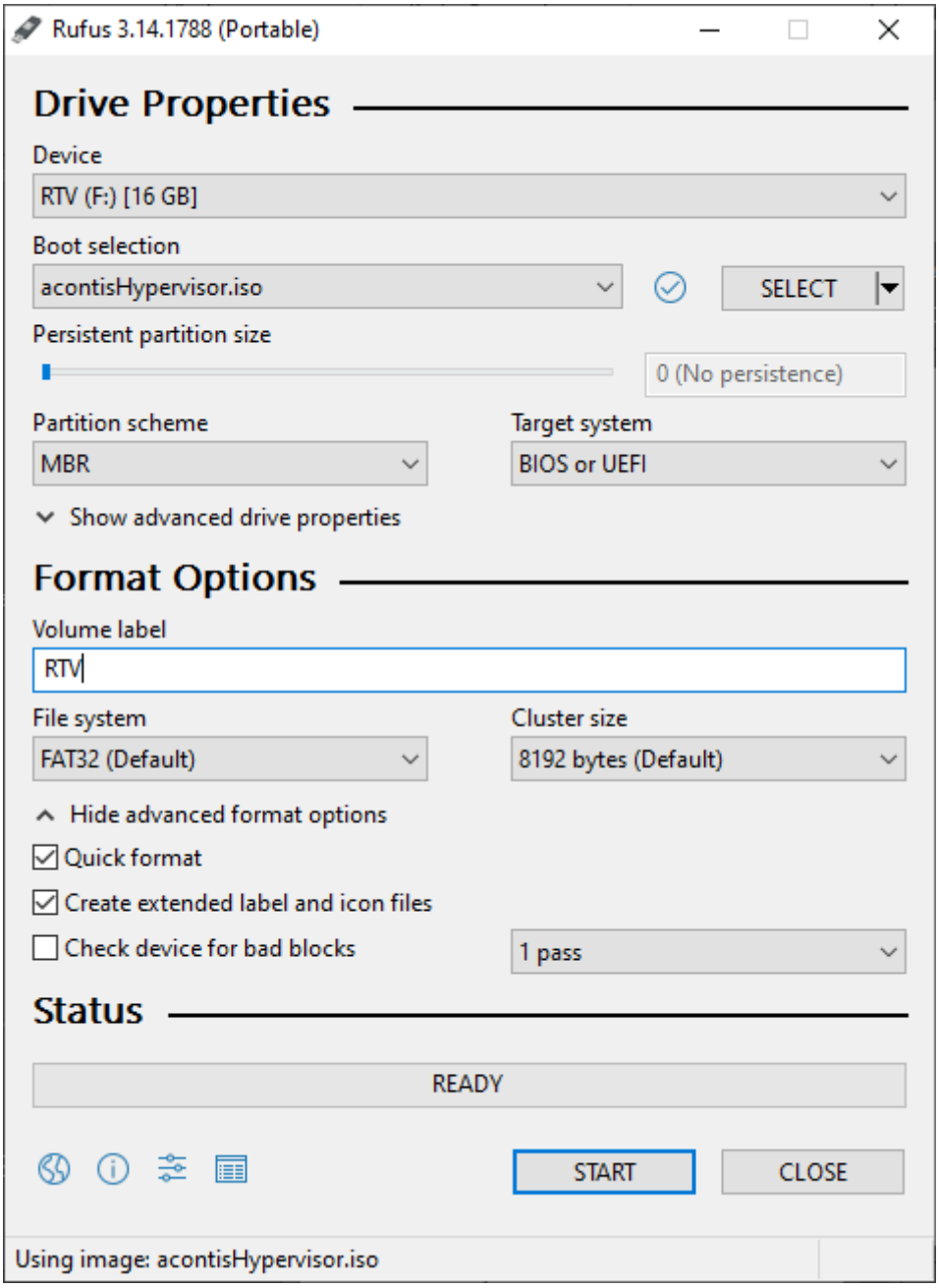


Fig. 2.1: Settings to create the acontis Hypervisor BOOT stick.

As the ISO is a hybrid ISO, Rufus will ask for the mode. Use the default ISO Image mode.

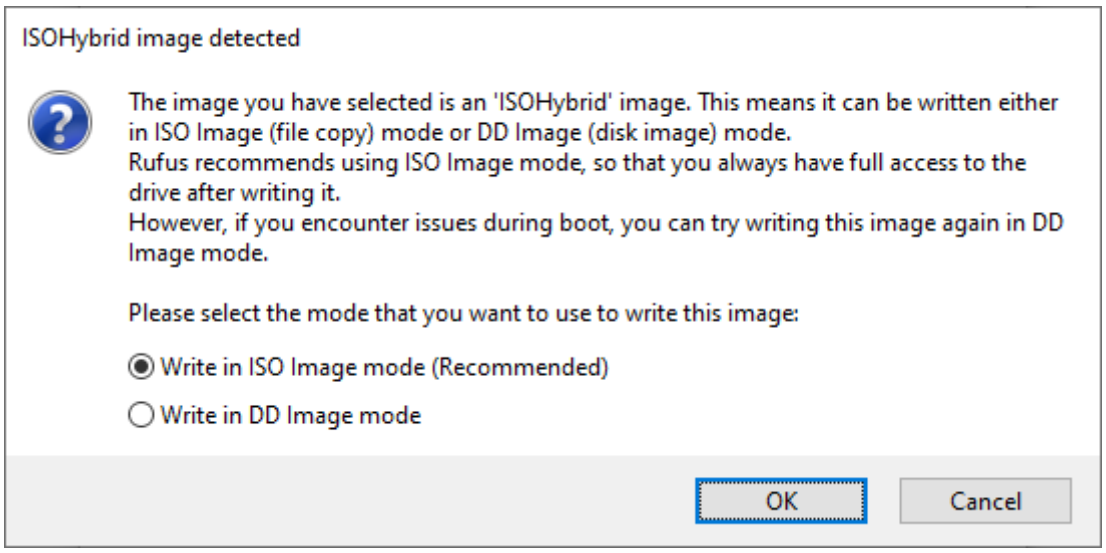


Fig. 2.2: Image mode selection.

2.2 Installation

Currently, a light-weight Ubuntu derivate (Xubuntu) is used as the Hypervisor service operating system.

- Select `Install acontis RTOSVisor` to install the Hypervisor, other options are not supported.
- Follow the installation instructions.
- Do not keep any existing Windows or Linux installation, let the installer erase previously installed OSes.
- A user account **must** be created while installing the Hypervisor.

Note: A password **MUST** be defined!

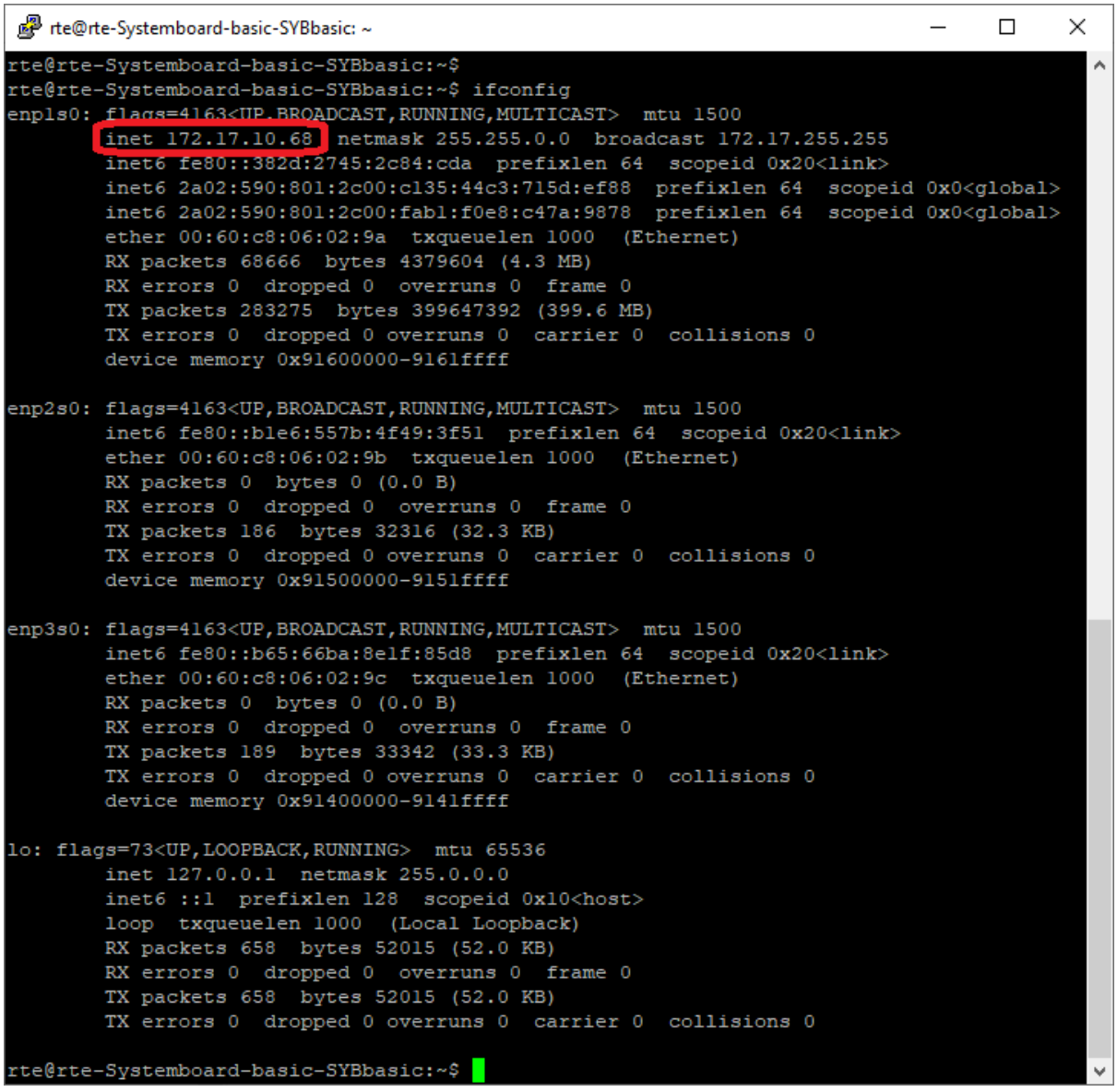
- After the installation has finished, you have to reboot the system.
- When the BIOS startup finished, a boot menu is shown and the default entry `Ubuntu` is selected, keep this and do **not** boot the `Hypervisor` entry. This entry will be activated automatically **after** the *first-time* configuration.
- After the first login, you may be asked to upgrade the Xubuntu version. Select `Don't Upgrade`

2.3 Remote Access to the Hypervisor using Remote Desktop Connection

When working with Windows for software development, it is recommended to remotely connect to the Hypervisor using a Remote Desktop Connection (using the RDP protocol). On Xubuntu, the `xrdp` application will provide such access. When using Remote Desktop you may simply copy paste commands from this manual into the Hypervisor shell.

First you need to determine the IP address of the Hypervisor. Open a shell (right click on desktop and select *'Open Terminal here'* **or** press `CRTL` + `ALT` + `T`)

```
$ ifconfig
```



Now its possible to access the Hypervisor system through Windows Remote Desktop using the previously determined IP address.

Hint: Use the user account, which was created when installing the Hypervisor.

Caution: To log-in remote into the Hypervisor through `xrdp`, **no** other user must be logged-in into the Hypervisor. You must log off before connecting.

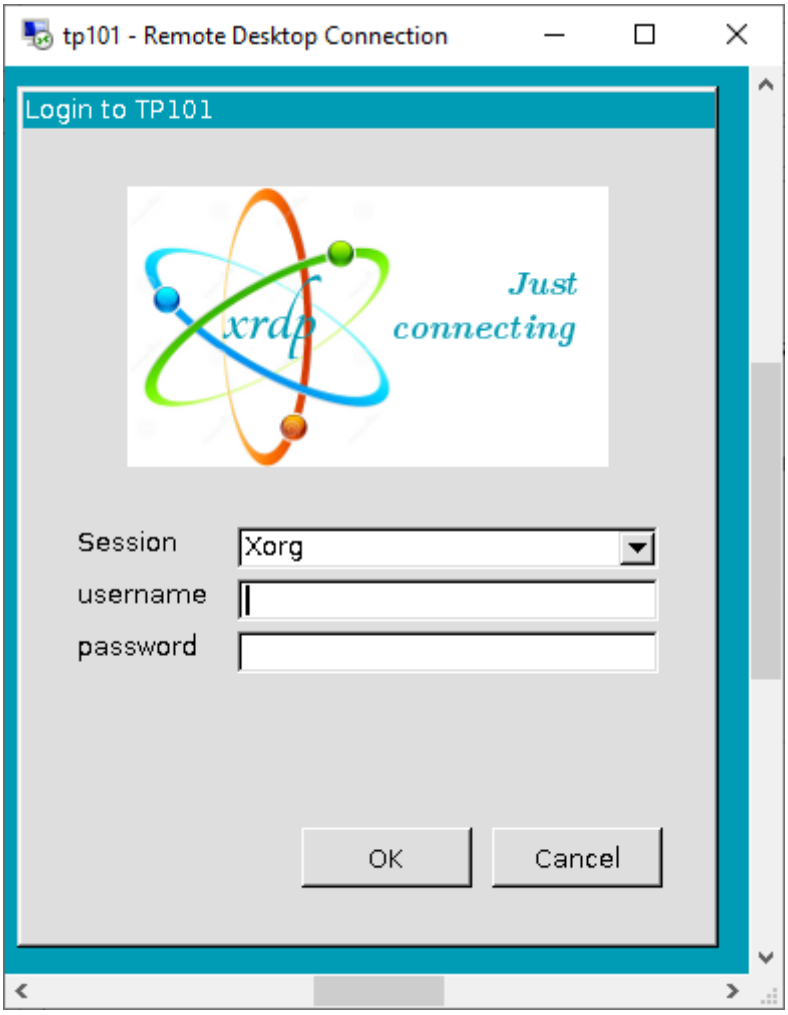


Fig. 2.3: xrdp remote login screen.

2.4 Basic system configuration

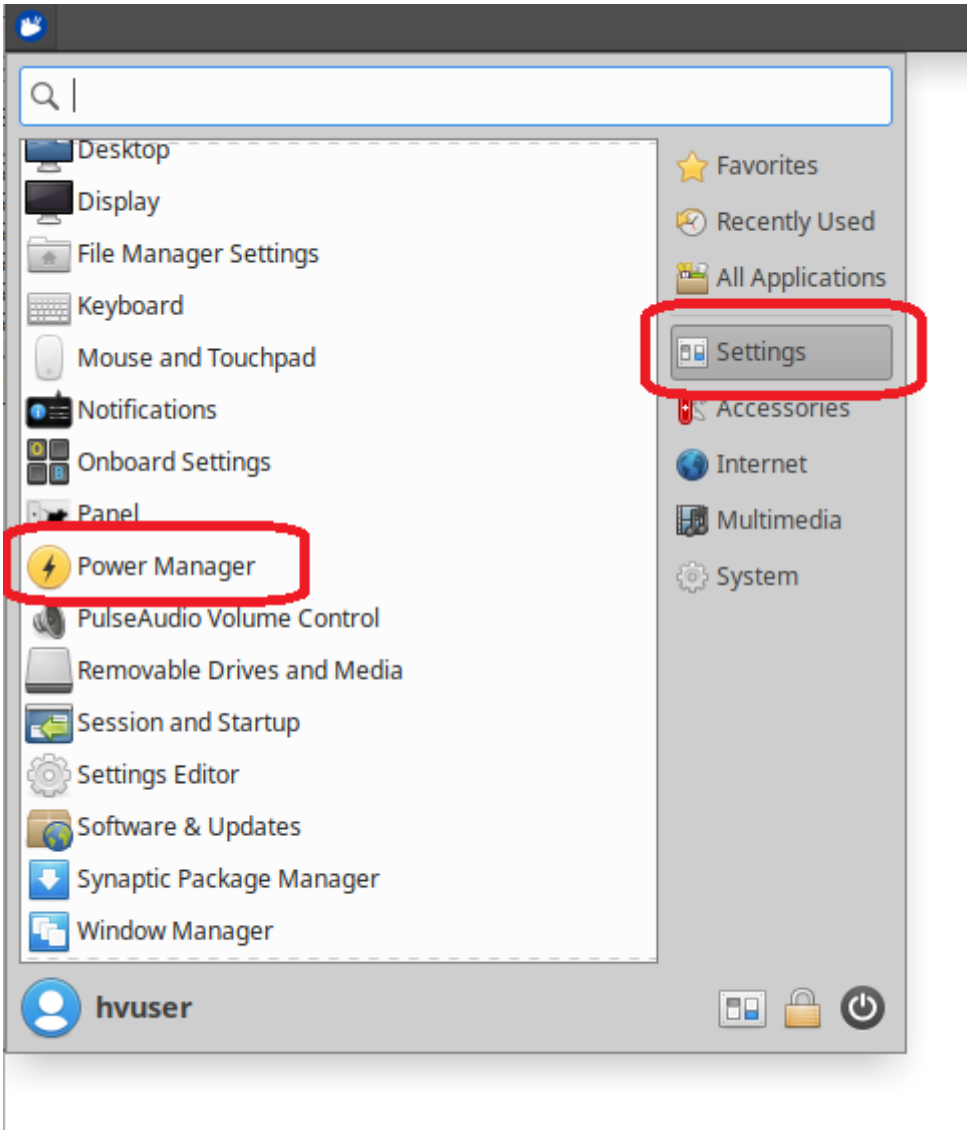
2.4.1 Power Management

To assure deterministic real-time behavior it is necessary to disable all power saving settings.

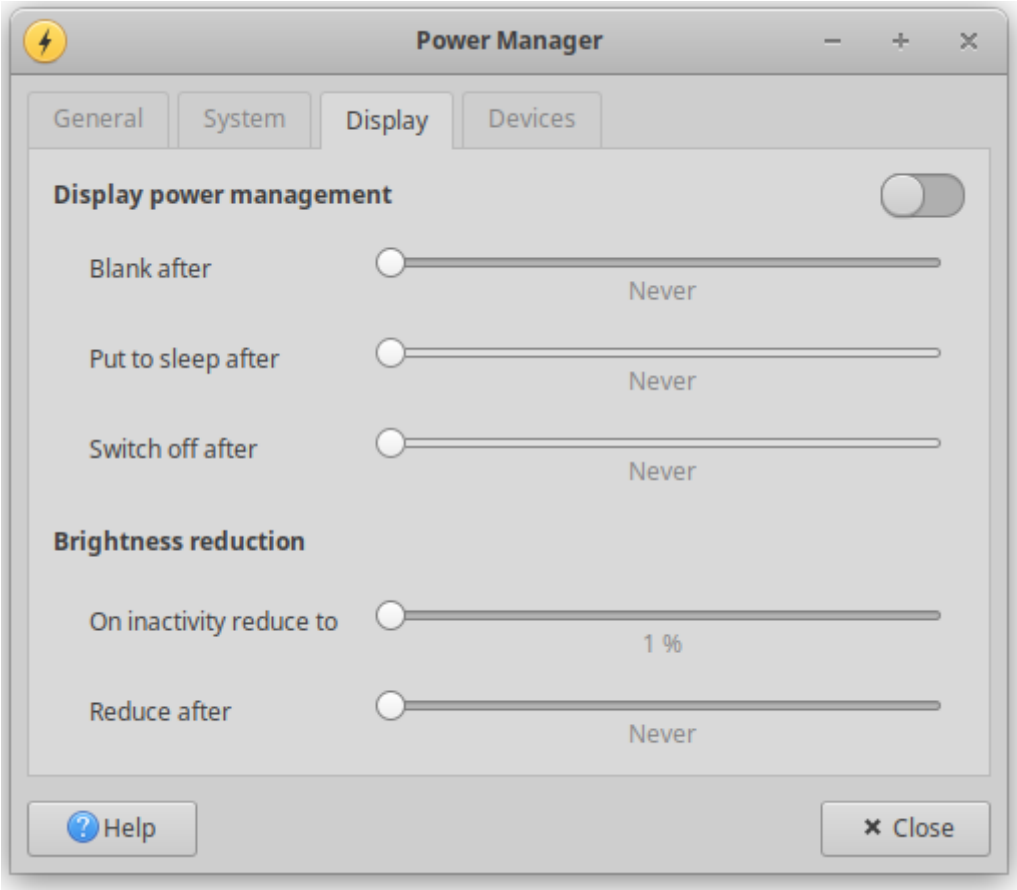
Several settings in the PC BIOS will have to be adjusted. You may take a look at the acontis website for example BIOS settings or pre-validated hardware BIOS settings

You need also to assure the Display Power Management is turned off in the hypervisor host.

First, open the Power Manager



Then assure the Display Power Manager is turned off



2.4.2 hardware partitioning

Open a shell (right click on desktop and select *'Open Terminal here'* **or** press `CRTL` + `ALT` + `T`)

Change to the configuration directory:

```
$ cd /hv/config
```

In the next step, the memory configuration for the real-time guest OS is set (this setting is **rtos-dependent!**). Use the `inithv` script:

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

The first parameter of the `inithv.sh` script will set the amount of memory to be assigned to the real-time guest OS, the second parameters will define the shared memory pool size in `MB`.

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, a second run of the `inithv` script may fix this issue.

Hint:

If you want to change the memory configuration at a later time, please use the `adjmemconf.sh` script. The following example shows how to change the configuration to use 768 MByte RAM for the real-time guest and 32 MByte RAM for the shared memory pool.

```
$ cd /hv/config
$ sudo ./adjmemconf.sh 768 32
$ sudo reboot
```

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

```
$ sudo reboot
```

2.5 Validate the configuration

The last step now is to validate if the system is correctly configured. For that purpose, a real-time demo application running in the real-time guest will be executed. The demo scripts to be used for this purpose will automatically start a console connection to show the output of the guest OS. When starting a Real-time Linux guest, you will have to log in first and start the demo manually. On RTOS-32 the demo is started automatically.

- **Real-time Linux**:

  ```
  $ cd /hv/lx
  $ sudo ./lx.sh
  $ ./dbgcon.sh
  ```

  Log in into Real-Time Linux and run the demo:

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

- **On Time RTOS-32**:

  ```
  $ cd /hv/rtos-32
  $ sudo ./realtimedemo.sh
  ```

To terminate the console connection to the real-time guest and stop showing the respective output messages: `CTRL` + `C`

Finally, stop the Real-time guest OS:

- **Real-time Linux**:

  ```
  $ cd /hv/lx
  $ sudo ./stopall.sh
  ```

- **On Time RTOS-32**:

```
$ cd /hv/rtos-32
$ sudo ./stopall.sh
```

Caution: **Always** stop the Real-time guest **before** starting again!

3 Ethernet Device Assigment

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`.

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.

3.1.1 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=5.15.0-41-generic 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'.

3.1.2 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`.

3.2 PCIe Device Assignment

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 script `./addeth.sh`. 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`.

```
$ cd /hv/config
$ sudo ./addeth.sh enp2s0 rtos_eth1 1
```

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

In case additional network devices shall be assigned:

```
$ sudo ./addeth.sh [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 `addeth.sh` script. 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!

3.3 Legacy PCI Device Assignment

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 `addeth.sh` script is required.

```
$ sudo ./addeth.sh [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.

3.4 Adjust the main Real-time guest configuration files

After creating the device configuration file `<RTOS device name>.config`, it needs to be included in the main configuration file. In this tutorial we use `rtos_eth1` as the unique `<RTOS device name>`. Please adjust the RTOS-32 configuration file as well because we will use the RTOS-32 guest for device assignment verification later.

- **RTOS-32 guest**:

  ```
  $ cd /hv/rtos-32
  $ gedit ./rtos-32demo.config
  ```

- **Real-time Linux guest**:

  ```
  $ cd /hv/lx
  $ gedit ./hv.config
  ```

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

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

```
;-----------------------------------------------------------------------
; includes
;-----------------------------------------------------------------------
#include "../config/membase.config"
#include "../config/memory.config"
#include "../config/cpu.config"
#include "../config/hwbase.config"
#include "../config/hwdevbase.config"
;#include "../config/hwdevbase_rtos2.config"
#include "../config/vmf.config"
;#include "../config/debug.config"
#include "../config/windowsvm.config"
#include "../config/rtos_eth1.config"
;#include "../config/pcidev1.config"
;#include "../config/pcidev2.config"
#include "linux.config"
```

3.5 Adjust the hypervisor partitioning script `/hv/config/hvpart.sh`

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

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

The `hvpart.sh` file should contain at least the following string **after** editing: `source $scriptDir/rtos_eth1.sh add`

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

```
$ #!/bin/bash
$ source /hv/config/rtos_eth1.sh add
```

Please reboot the system to make the change effective.

```
$ sudo reboot
```

Hint: The `hvpart.sh` script will be automatically started via the *systemd* service controlled via `/etc/systemd/system/hv_part.service`. 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
```

3.6 Device verification

3.6.1 Xubuntu de-assignment check

In a first step, check if the Xubuntu *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
```

(continues on next page)

(continued from previous page)

```
        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*).

3.6.2 Real-time guest assignment check

With the `RTOS-32Demo` it's possible to verify, if the assigned netwok adapter is visible to the rtos part:

```
$ cd /hv/rtos-32
$ sudo ./rtos-32demo.sh
```

The output will looks 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  -      -
```

(continues on next page)

(continued from previous page)

```
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/rtos-32
$ sudo ./stopall.sh
```

4 Running the acontis EtherCAT Master Demo

In this section we will get the EtherCAT master demo application running. In the first step, a simple network scan will be made. The network will **not** be set into **OPERATIONAL** mode.

4.1 Adjust the configuration

In this tutorial we use `rtos_eth1` as the unique `<RTOS device name>`. Please adjust the respective configuration file for the EtherCAT demo. You may also have to adjust the parameters for the master stack demo.

RTOS-32

```
$ gedit /hv/rtos-32/ecmasterdemo.config
```

Add or uncomment `#include "../config/rtos_eth1.config"`.

Assure the following command line is used (you will find this at the end of the file). `"CommandLine"="-i8254x 1 1 -v 3 -perf -t 0 -b 1000 -sp"`

Real-time Linux

```
$ gedit /hv/lx/hv.config
```

Add or uncomment `#include "../config/rtos_eth1.config"`.

4.2 Run the demo

RTOS-32

```
$ cd /hv/rtos-32
$ sudo ./ecmasterdemo.sh
```

Real-time Linux

```
$ cd /hv/lx
$ sudo ./lx.sh
$ ./dbgcon.sh
```

Start the EC-Master (*without* an `ENI`):

```
$ vmf64 login: root
$ password:  root
$ cd /mnt/rtfiles/ecat
$ ./noeni.sh
```

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

`CTRL + C`

```
$ sudo ./stopall.sh
```

4.3 Using Network configuration file (ENI)

In this section we will run the EtherCAT network into **OPERATIONAL** mode. You need to generate a network configuration files (`ENI`) prior to executing this tutorial. If you are not yet familiar with ENI files you should skip this chapter for now and return on demand.

RTOS-32

Copy your `ENI` file to `/hv/rtos-32/rtfiles/ecmasterdemo/ENI.xml`.

```
$ gedit ./hv/rtos-32/ecmasterdemo.config
```

Locate the `[ecat]` entry `"CommandLine"` and append `-f ENI.xml` to its value. Now you can run the demo as before.

Real-time Linux

Copy your `ENI` file to `/hv/lx/rtfiles/ecat/ecmasterdemo/ENI.xml`.

```
$ cd /hv/lx/rtfiles/ecat
$ cp ./noeni.sh ./eni.sh
$ gedit ./eni.sh
```

Add `-f ENI.xml` to the command line parameters entry. Now you can run the demo with ENI using `./eni.sh` instead of `./noeni.sh`.

5 Automatic Startup

In this section we will show how the RTOS and an application will be started automatically (*RT-Linux only*) after the Hypervisor has finished booting.

5.1 Automatic RTOS start

To automatically start RT-Linux after the hypervisor finished booting, please adjust the file

```
$ gedit /hv/lx/rtos_start.sh
```

and uncomment the line `./lx.sh`.

Hint:

The `rtos_start.sh` script will be automatically started via the `systemd` service controlled via `/etc/systemd/system/hv_rtos_start.service`.
This service can be enabled (default) or disabled:

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

5.2 Autostart RT-Linux Applications

Automatic startup of RT-Linux applications is configured in `/hv/lx/linux.config` (section `[Rtos\Autostart\1]`). By default automatic start is enabled.

If this autostart section is active, the autostart script `/hv/lx/rtfiles/autostart.sh` will be executed after RT-Linux has booted.

You may adjust both files according to your needs.