acontis technologies GmbH

SOFTWARE

Hypervisor-MultiOS-Guide

acontis Real-time Hypervisor MultiOS Setup

Version 1.0

Edition: November 3, 2022

Table of Contents

1 Introduction

This guide describes setting up a multi-OS setup with `2` running RTOS-32 instances (`RTOS` + `RTOS2`), each running an EtherCAT master stack.

This manual will guide you through the following steps:

- Hardware/Software setup including the `Intel PRO/1000` network adapter, several EtherCAT slaves. (See Chapter *Prerequisites*)
- Assign the network adapters to the RTOS
- Get the multi-OS setup running
- Get the `2` EC-Master instances running *without* ENI

Hint: `sudo` equals Windows `run as administrator`.

Hint: `sudo -s` switches user to `root` (administrator).

1.1 Prerequisites

- PC/IPC with at least `2` network devices.
  - CPU: at least `4` cores. Default configuration is optimized for `4` cores.
  - HDD: at least `20GB`. Additional `50GB` in case of Windows-VM-Guest
  - preferably `2x Intel PRO/1000` (like `I210`), as the scripts are preconfigured to this network adapter.
- `2` EtherCAT slaves to attach to the network adapter

Caution: The PC/IPC will be used to install/co-install an UBUNTU derivate!

- BIOS settings
  - USB Legacy mode has to be disabled
  - Hyper-Threading shall be disabled
  - Power saving settings have to be disabled (C-States, Speedstepping, …)

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 used, assure a free partition for booting the hypervisor is required. The minimum required size for the installation media is 10 GByte.

2.1 USB stick boot medium creation

We simply recommend the Rufus tool to create the boot medium stick.

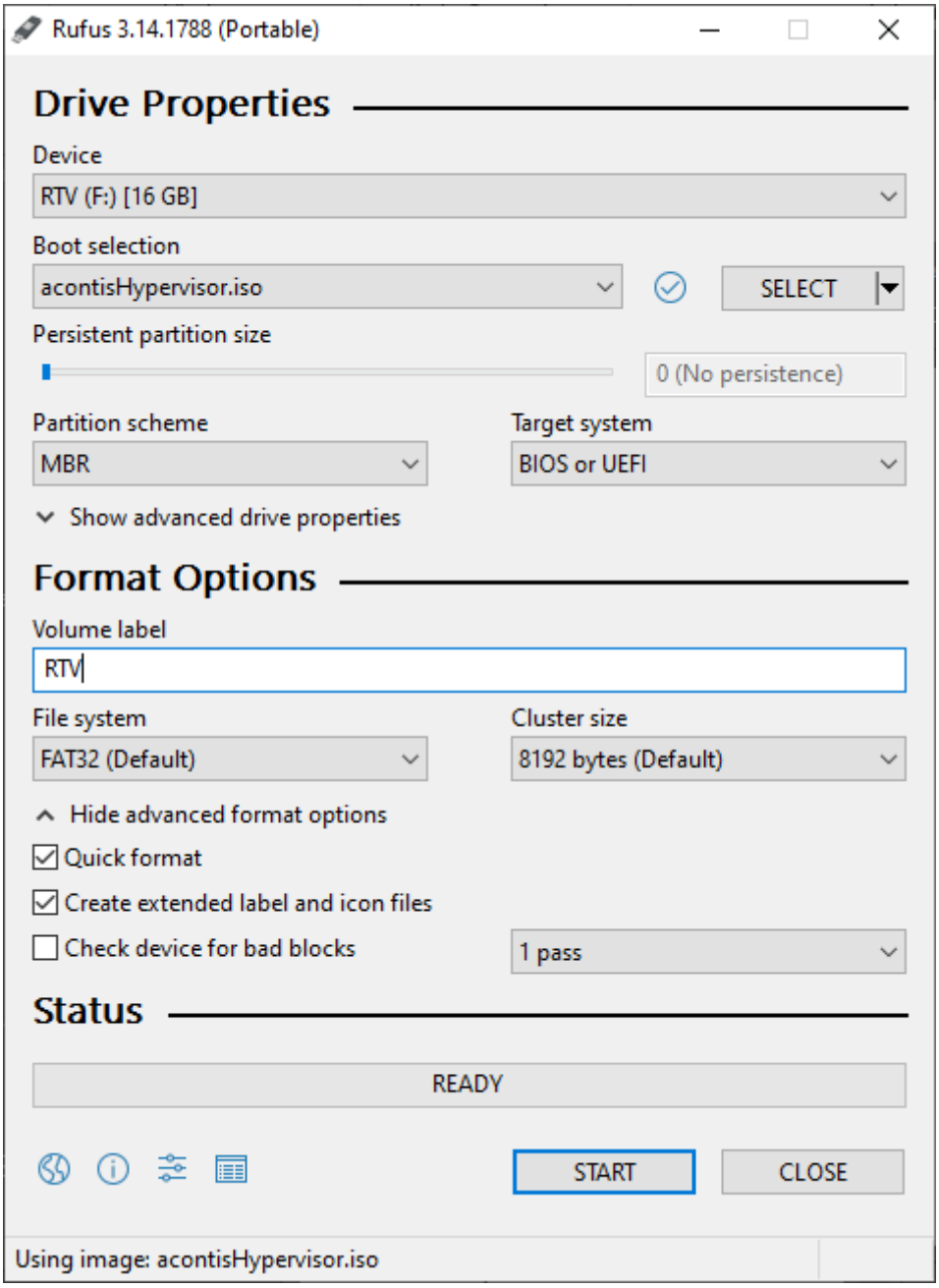

Fig. 2.1: Settings to create an acontis Hypervisor BOOT stick.

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 Access Hypervisor PC using Remote Desktop Connection

It's recommended to access the hypervisor system through `xrdp`, if a Windows development PC is used. Windows provides a feature `RDP` for remote desktop access. It's easier to copy-paste the successive shell commands through (x)RDP.

The following steps must be done in case of using `xrdp`:

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

Allow firewall to open required port. All other required packages are yet included.

```
$ sudo ufw allow 3389
```

Now its possible to access the hypervisor system through Windows RDP.

Hint: Use the user account, which was created at *install* state.

Caution: To log-in remote into the hypervisor through `xrdp`, **no** other user must be logged-in into the hypervisor.

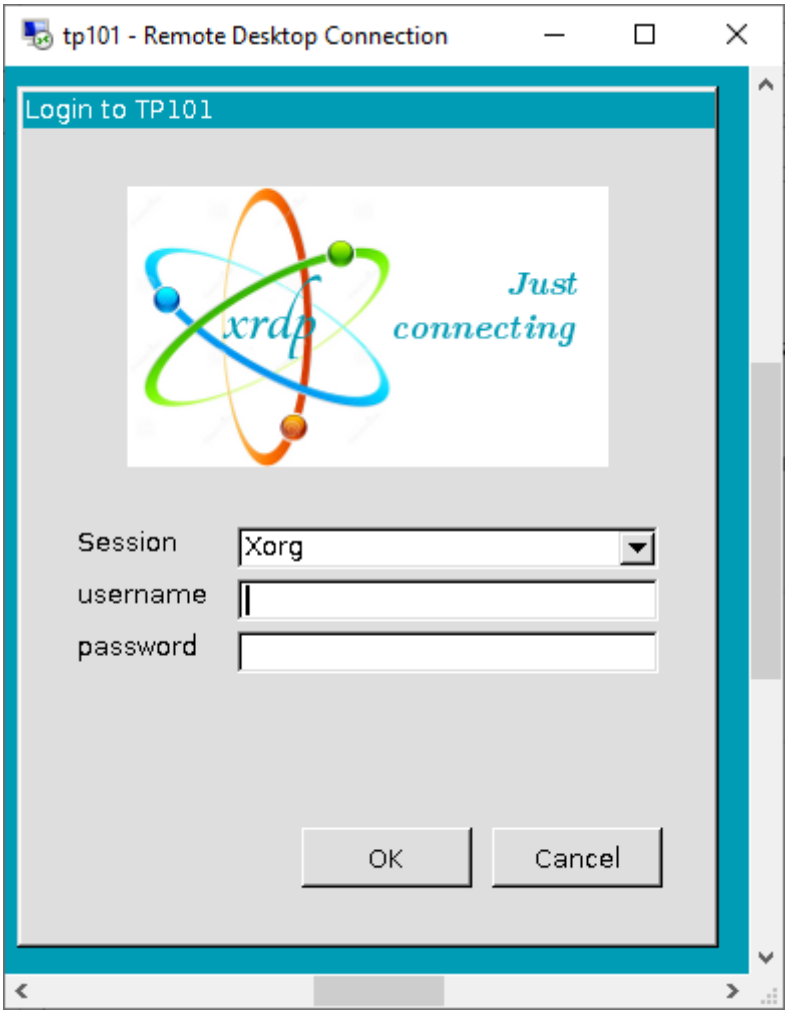

Fig. 2.2: xrdp remote login screen.

2.4 Basic system configuration

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

Next step we need the memory assignment (**rtos-depend!**) and the CPU count through the inithv script:

```
$ cd /hv/config
$ sudo ./inithv.sh 64 16 2
```

The first parameter of the `inithv.sh` describes the amount of assigned memory of the Rtos and the second one the shared memory pool size in `MB`. The third parameter (`2`) describes the number of CPU cores for the Rtos'es. In this example `2` cores are assigned to the rtos part.

Caution: If you get the message `WARNING: CPU frequency not stable...` through the execution of the `inithv.sh` script, you did not properly disable power settings in the BIOS. Please follow the respective instructions above (Prerequisites).

A second run of the inithv script may help in that case **after!** a reboot.

2.5 Setup processor count

The setup of our hypervisor assigns the **last** core *automatically* to the Rtos. In a multi OS enviroment more than the last core is needed by the Rtos'es. In this case for each **additional** Rtos `1` core is subtracted from the remaining count for the hypervisor.

Check grub configuration:

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

Search entry `linux /boot/...` and check parameter `maxcpus` if `maxcpus=2`. This is the *default* value in our example configuration (quad-core) in this tutorial. For a quad-core CPU the first `2` cores are assigned to the hypervisor and the next `2` cores each to `1` Rtos.

Update grub:

```
$ sudo update-grub
```

2.6 Setup `memory.config`

The file `memory.config` describes the memory distribution of the hypervisor and the guest OS. The supplied values are in `hex`.

```
$ gedit /hv/config/memory.config
```

Change entry `[Rtos] "MemorySize"` to value `2000000`

```
[Rtos]
    "MemorySize"=dword:2000000              ; 1st RTOS
[Rtos-End]
```

2.7 Setup `cpu.config`

The file `cpu.config` describes, which CPU core is assigned exclusivly to which Rtos. This is a bit mask (flags) and the supplied values are in `hex`.

```
$ gedit /hv/config/cpu.config
```

The *default* in this tutorial is based on a quad-core CPU. Core `1` and `2` are assigned to the *Hypervisor* `(0x3)`, core `3` to *RTOS2* `(0x4)` and core `4` to *RTOS* `(0x8)`.

```
;-----------------------------------------------------------------------
; HV guest and Windows guest
;-----------------------------------------------------------------------
[Windows]
  "ProcessorMask"=dword:3      ; 0x3 = 0011b = core 0 + core 1 to␣
↪hypervisor

;-----------------------------------------------------------------------
; RTOS
;-----------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
[Rtos]
  "ProcessorMask"=dword:8       ; 0x8 = 1000b = core 4 to RTOS       (bit␣
↪mask, bit 0 = core 0, bit 1 = core 1, ...)

[Rtos2]
  "ProcessorMask"=dword:4       ; 0x4 = 0100b = core 3 to RTOS2      (bit␣
↪mask, bit 0 = core 0, bit 1 = core 1, ...)
```

Caution: You need to uncomment the [Rtos2] entries!

2.8 Setup `hwdevbase_rtos2.config`

Caution: This section shouldn't be modified - **only** in special cases!

```
$ gedit /hv/config/hwdevbase_rtos2.config
```

Change all entries (at least `4` occurences) of `[...\Rtos2\...] "Destination"` to value `0x4`.

Example:

```
[Devices\Rtos2\RTOS Local Apic Timer\Interrupt\1]
    "Type"=dword:2
    "Id"=dword:1A
    "LocalApicType"=dword:0
    "DestinationFormat"=dword:0
    "Destination"=dword:4
    "Vector"=dword:F6
```

2.9 Reboot system

To get the changes effective, a reboot is required:

```
$ sudo reboot now
```

3 Running the RealtimeDemo Application

Start Rtos:

The `multiosdemo-realtime.sh` will start both rtos (`RTOS` + `RTOS2`) instances and show `2` output consoles.

```
$ cd /hv/rtos-32
$ sudo ./multiosdemo-realtime.sh
```

Stop output and return to console: `CTRL` + `C`

Stop Rtos:

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

> **Caution:** **Always** call stop before start again!

4 Ethernet Device Assigment

Hint: All user inputs should be done in a Terminal, or in case a file is edited in an editor.

4.1 Device identification

There are several ways to detect the desired network adapter(s), which should be assigned to the rtos part. This tutorial use as simple way to detect the desired network adapter. In our test case we need at least 2 network adapter, which are not connected at this stage.

Hint: In our example the PC has 3 network adapters. 1 *connected* to the company network and 2 *not-connected* anywhere. This 2 adapters will be used later by the EC-Master and will be connected to the EtherCAT slaves.

List all available adapters:

```
$ ifconfig -a
```

The output will be similar to the following output. The adapter, which is connect to the company network will contain a <link> entry. This adapter will be ignored. The other 2 adapters are desired. The lo with inet 127.0.0.1 is the localhost, which **isn't** desired also.

```
enp0s25: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.17.10.45  netmask 255.255.0.0  broadcast 172.17.255.255
    inet6 2a02:590:801:2c00:6fef:fc8f:977f:aa12  prefixlen 64  scopeid 0x0
↪<global>
    inet6 fe80::ce1d:c9c9:aa11:fa9  prefixlen 64  scopeid 0x20<link>
    inet6 2a02:590:801:2c00:bd83:9c95:38f0:6305  prefixlen 64  scopeid 0x0
↪<global>
    ether 00:19:99:de:98:cf  txqueuelen 1000  (Ethernet)
    RX packets 48668  bytes 60096879 (60.0 MB)
    RX errors 0  dropped 2  overruns 0  frame 0
    TX packets 28112  bytes 33952890 (33.9 MB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 20  memory 0xfbb00000-fbb20000

enp1s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ether 00:15:17:8a:bd:d9  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
    device interrupt 16  memory 0xfba40000-fba60000

enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ether 00:19:99:e0:e6:73  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
```

(continues on next page)

(continued from previous page)

```
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 18  memory 0xfb900000-fb920000

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 207  bytes 16390 (16.3 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 207  bytes 16390 (16.3 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

In this example the name of the desired network adapters are `enp1s0` and `enp2s0`.

4.2 Device assignment

In the previous section the detected name `[DETECTED_NAME_1]` of the adapter is `enp1s0`. This name will be given as first parameter of the `./addeth.sh` shell script. The second parameter (**mandatory**) is the rtos device name. Third parameter is a sequential number for each assigned device, starting with `1`.

```
$ cd /hv/config
$ sudo ./addeth.sh enp1s0 rtos_eth1 1
```

generic:

```
$ sudo ./addeth.sh [DETECTED_NAME_1] rtos_eth 1
```

The same **must** be done for the second network card (device) also. The the detected name `[DETECTED_NAME_2]` of the adapter is `enp2s0`:

```
$ sudo ./addeth.sh enp2s0 rtos_eth2 2
```

generic:

```
$ sudo ./addeth.sh [DETECTED_NAME_2] rtos_eth2 2
```

Important: The script **creates** `2` files located in `/hv/config`: `rtos_eth1.sh`, `rtos_eth1.config` respective `rtos_eth2.sh`, `rtos_eth2.config`.

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`, respective `rtos_eth2`.

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!

4.3 Adjust generated `rtos_eth2.config`

```
$ sudo gedit /hv/config/rtos_eth2.config
```

Replace **all** occurences of `Rtos` in the section labels `[..\Rtos\..]` with `Rtos2`. Finally it should looks like that: `[..\Rtos2\..]`

Example:

- **Old:**

```
[Devices\Rtos\Intel Corporation 82574L Gigabit Network Connection\
↪IoPort\1]
  "Address"=hex:00,d0,00,00,00,00,00,00
  "Length"=dword:20
```

- **New:**

```
[Devices\Rtos2\Intel Corporation 82574L Gigabit Network␣
↪Connection\IoPort\1]
  "Address"=hex:00,d0,00,00,00,00,00,00
  "Length"=dword:20
```

4.4 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`, respectivly `rtos_eth2` as the unique `<RTOS device name>`, as mentioned earlier.

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

The `hvpart.sh` file should contain at least the following strings **after** editing: `source /hv/config/rtos_eth1.sh add` and `source /hv/config/rtos_eth2.sh add`

The example below shows, how the devices with the *unique* names `rtos_eth1`, respective `rtos_eth2` are assigned.

```
#!/bin/bash
source /hv/config/rtos_eth1.sh add
source /hv/config/rtos_eth2.sh add
```

4.5 Reboot system

Please reboot the system to make the change effective.

```
$ sudo reboot now
```

5 Running the EC-Master Demo Application

Start Rtos:

The `multiosdemo-ecmaster.sh` will start *both* rtos (`RTOS` + `RTOS2`) instances and show `2` output consoles.

```
$ cd /hv/rtos-32
$ sudo ./multiosdemo-ecmaster.sh
```

Stop output and return to console: `CTRL` + `C`

Stop Rtos:

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

> **Caution:** **Always** call stop before start again!