



**acontis technologies GmbH**

**SOFTWARE**

# **Hypervisor-Graphics-Passthrough-Guide**

**acontis Real-time Hypervisor Graphics Passthrough Setup**

**Version 9.x**

**Edition: July 29, 2025**

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

© Copyright acontis technologies GmbH

Neither this document nor excerpts therefrom may be reproduced, transmitted, or conveyed to third parties by any means whatever without the express permission of the publisher. At the time of publication, the functions described in this document and those implemented in the corresponding hardware and/or software were carefully verified; nonetheless, for technical reasons, it cannot be guaranteed that no discrepancies exist. This document will be regularly examined so that corrections can be made in subsequent editions. Note: Although a product may include undocumented features, such features are not considered to be part of the product, and their functionality is therefore not subject to any form of support or guarantee.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Prerequisites</b>	<b>5</b>
2.1	Additional information	5
<b>3</b>	<b>Guest configuration</b>	<b>6</b>
<b>4</b>	<b>Windows guest remote desktop access</b>	<b>7</b>
4.1	Enable Remote Desktop	7
4.2	Remote Desktop Settings	9
<b>5</b>	<b>Activate <code>vflio</code> driver</b>	<b>11</b>
<b>6</b>	<b>Additional guest configuration</b>	<b>12</b>
<b>7</b>	<b>Boot Hypervisor in passthrough graphics mode</b>	<b>14</b>
<b>8</b>	<b>Connect to Hypervisor in iGVT-d Mode</b>	<b>15</b>
<b>9</b>	<b>Remote guest start</b>	<b>16</b>
<b>10</b>	<b>Update display driver</b>	<b>17</b>
<b>11</b>	<b>Troubleshooting</b>	<b>18</b>
11.1	Windows guest	18
11.2	Linux guest	19
11.3	General issues	20

## 1 Introduction

This guide describes how to set up a graphics passthrough to the Windows or Ubuntu guest in the acontis Hypervisor. It is assumed the steps listed in the Hypervisor Quick Start guide have been successfully executed.

## 2 Prerequisites

- Intel i915 graphics adapter
- additional extern graphics adapter (*optional*)
- **activated VT-D/IOMMU (UEFI/BIOS)**
- At least **6 GB of RAM** is recommended for the Windows guest (`ramsize=6144` in `guest_config.sh`).

### 2.1 Additional information

For more information on the topic checkout the following links:

- Intel (filtered) list of possible CPUs:

[https://ark.intel.com/content/www/us/en/ark/search/featurefilter.html?productType=873&1\\_Filter-ProcessorGraphics=19001&0\\_VTD=True](https://ark.intel.com/content/www/us/en/ark/search/featurefilter.html?productType=873&1_Filter-ProcessorGraphics=19001&0_VTD=True)

### 3 Guest configuration

Some custom steps are required to create a Windows or Ubuntu VM for QEMU.

**Caution:** It is important to create this machine with OVMF UEFI, because graphics passthrough needs it.

---

**Important:** Use the Windows or Ubuntu guest guide to setup a guest.

---

## 4 Windows guest remote desktop access

It's required to have Remote Desktop access of the Windows guest, as the standard vga graphics will be **deactivated** and therefore **only** remote access is possible with the Hypervisor Host and Windows guest.

### 4.1 Enable Remote Desktop

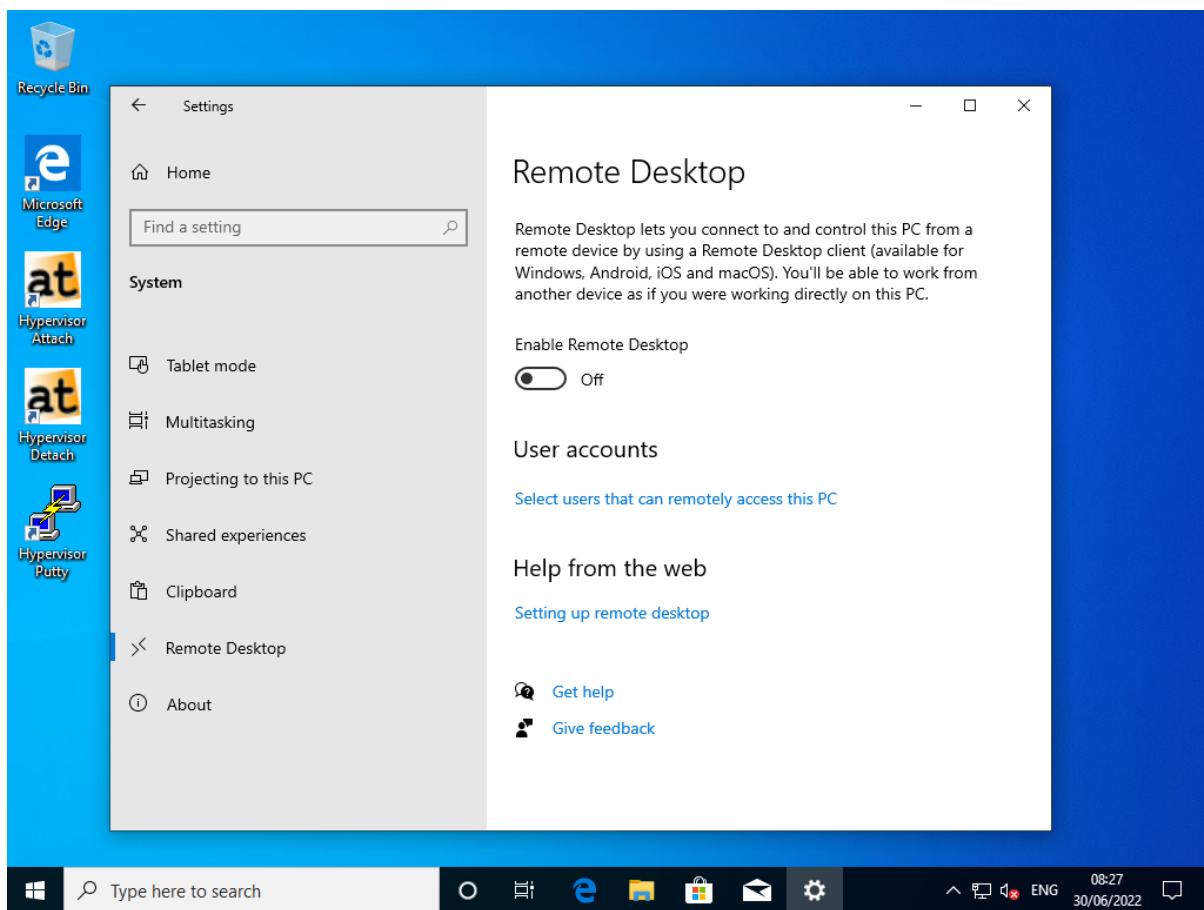


Fig. 4.1: Enable Windows 10 RDP

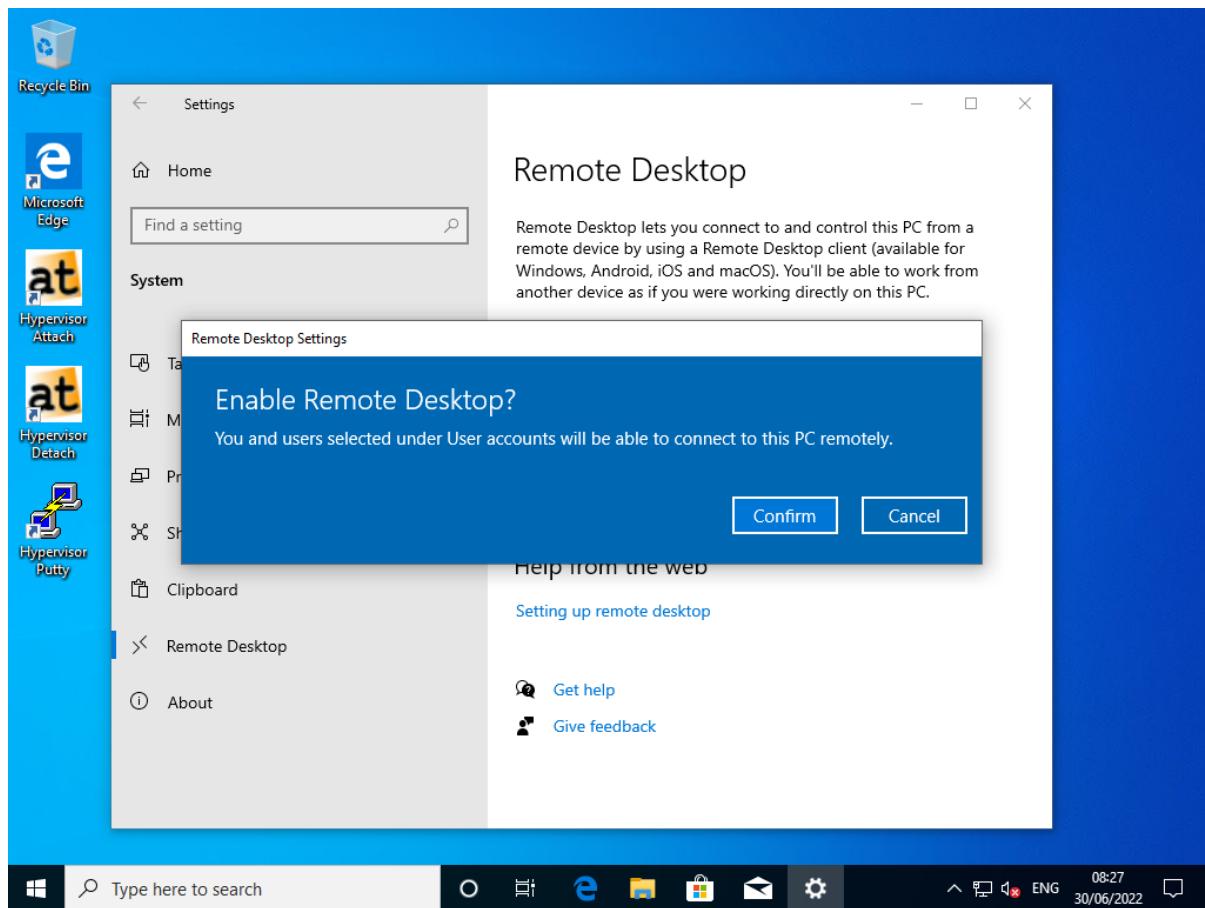


Fig. 4.2: Confirm Windows 10 RDP Enable.

## 4.2 Remote Desktop Settings

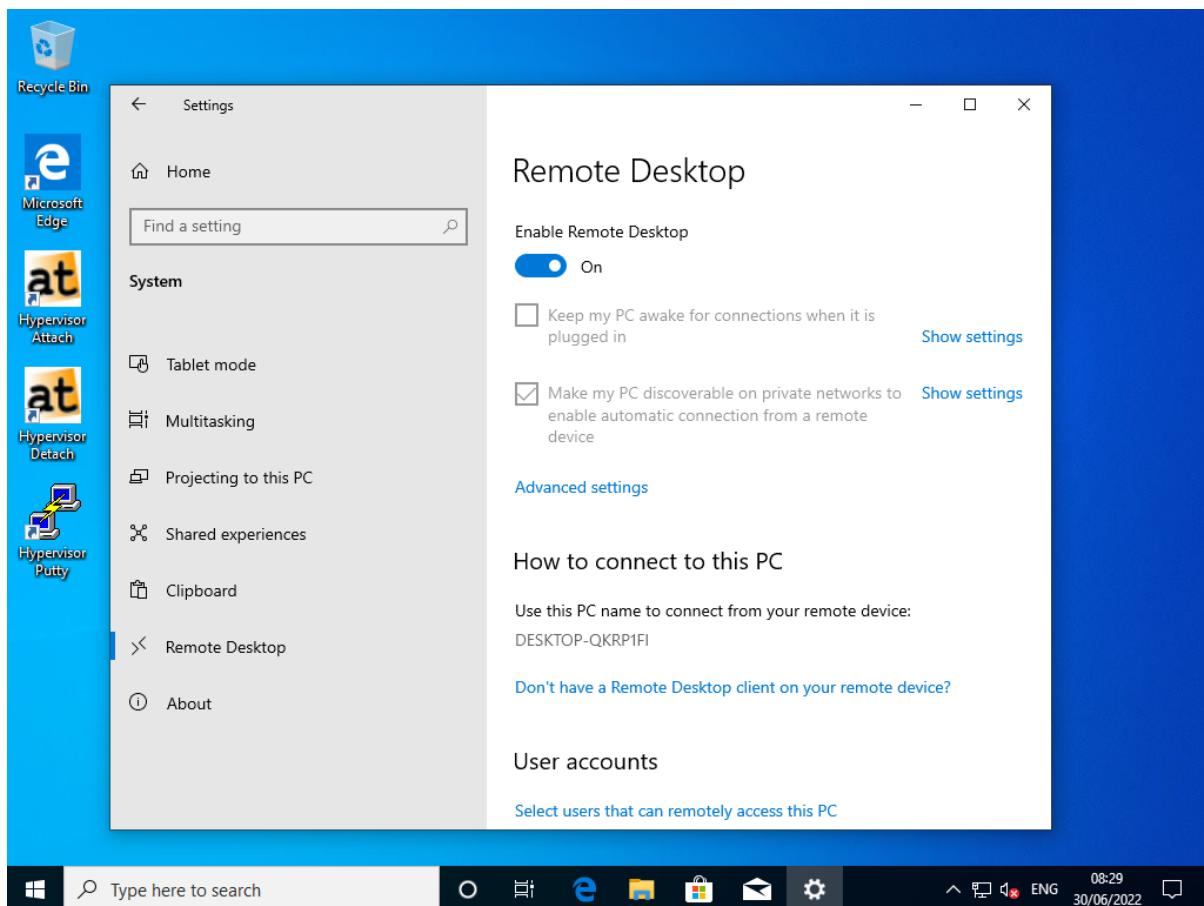


Fig. 4.3: Windows 10 RDP Settings.

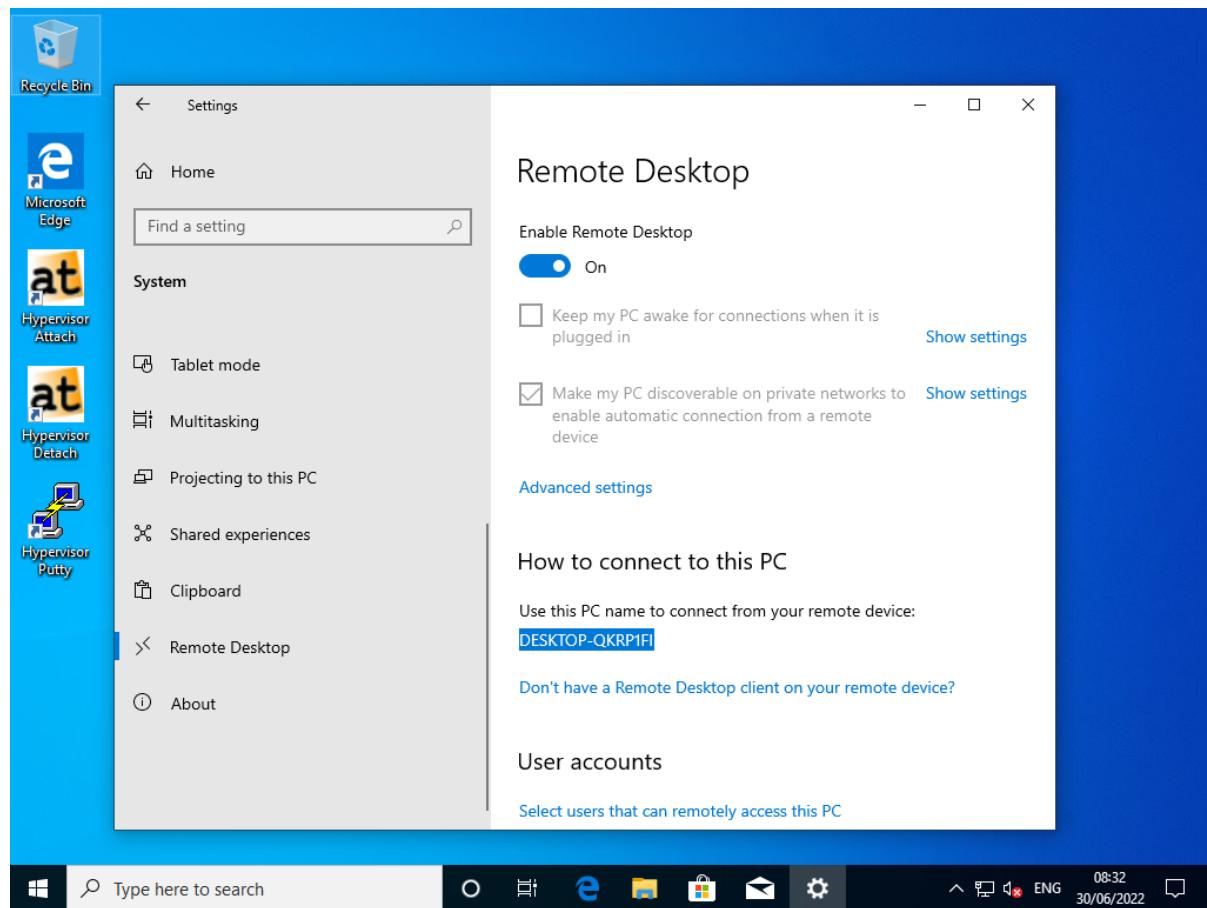


Fig. 4.4: Windows 10 RDP Settings PC-Name (Windows 10 guest).

## 5 Activate vfio driver

```
$ sudo mousepad /etc/modules
```

Add the following lines to the file:

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

```
$ sudo update-initramfs -u
```

```
$ sudo reboot now
```

## 6 Additional guest configuration

- Modify the file `usr_guest_config.sh` in your Windows or Ubuntu guest configuration

```
$ sudo nano /hv/guests/guestxxxx/usr_guest_config.sh
```

```
# UEFI support
export uefi_bios=1

# graphics passthrough
export enable_vga_gpt=1          # set to 1 to enable graphics
→passthrough
export vga_gpt_bdf=00:02.0        # PCI bus, device function values for
→the graphics card, use the lspci command to determine

# determine using ls -la /dev/input/by-id | grep -event- (search for
→the appropriate keyboard device name)
# e.g. vga_gpt_kbd_event_device="usb-LITEON_Technology_Corp._HP_125_
→Wired_Keyboard-event-kbd"
export vga_gpt_kbd_event_device="" # keyboard device name
export vga_gpt_mouse_event_device="" # mouse device name
```

For more information about graphics passthrough, see the [Hypervisor Manual](#).

**Important:** Use `lspci` to determine/validate the `vga_gpt_bdf` value!

```
rte@RTV-TP104:~$ lspci
00:00.0 Host bridge: Intel Corporation Xeon E3-1200 v2/3rd Gen Core
→processor DRAM Controller (rev 09)
00:02.0 VGA compatible controller: Intel Corporation Xeon E3-1200 v2/3rd
→Gen Core processor Graphics Controller (rev 09)
00:14.0 USB controller: Intel Corporation 7 Series/C210 Series Chipset
→Family USB xHCI Host Controller (rev 04)
00:16.0 Communication controller: Intel Corporation 7 Series/C216 Chipset
→Family MEI Controller #1 (rev 04)
...
...
```

**Important:** Use `ls -la` to determine/validate the `vga_gpt_kbd_event_device` and `vga_gpt_mouse_event_device` values! These values are only valid if bootet with GRUB entry Hypervisor + iGVT-d.

```
rte@RTV-TP104:~$ ls -la /dev/input/by-id | grep -event-
lrwxrwxrwx 1 root root 9 Feb 7 18:35 usb-Telink_Wireless_Receiver-event-
→if00 -> ../event5
lrwxrwxrwx 1 root root 9 Feb 7 18:35 usb-Telink_Wireless_Receiver-event-
→mouse -> ../event4
lrwxrwxrwx 1 root root 9 Feb 7 18:35 usb-Telink_Wireless_Receiver-if01-
→event-kbd -> ../event7
```

```
export vga_gpt_kbd_event_device="usb-Telink_Wireless_Receiver-if01-event-  
↪kbd"      # keyboard device name  
export vga_gpt_mouse_event_device=" usb-Telink_Wireless_Receiver-event-  
↪mouse"    # mouse device name
```

```
$ sudo reboot now
```

## 7 Boot Hypervisor in passthrough graphics mode

Please select the following boot line at GRUB:

Hypervisor + iGVT-d

---

**Hint:** When installing the hypervisor, a **separate** GRUB entry *Hypervisor + iGVT-d* is created.

---

---

**Important:** A missing GRUB entry *Hypervisor + iGVT-d* shows, that when installing the hypervisor, the installation script **didn't** find any *compatible integrated* graphics card!

---

## 8 Connect to Hypervisor in iGVT-d Mode

In this guide the Putty is used as remote SSH shell:

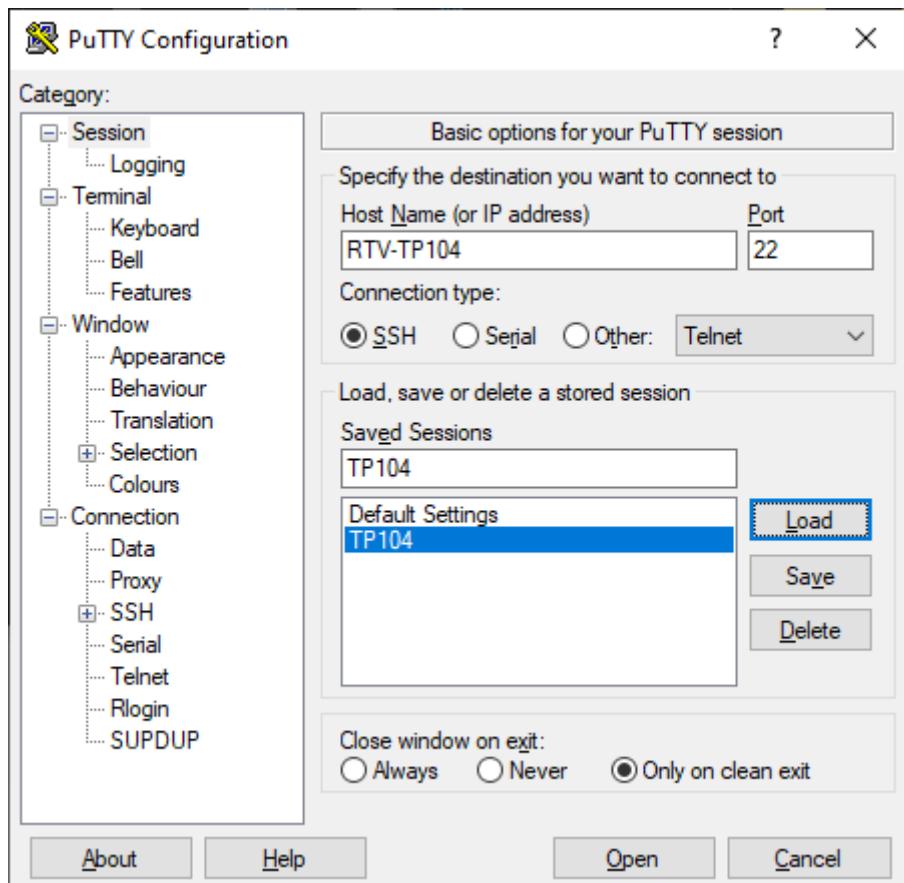


Fig. 8.1: Putty Configuration.

---

**Hint:** The screenshots uses the PC-Name RTV-TP104. Replace it by the name you used installing the Hypervisor!

---

## 9 Remote guest start

- Change to your guest directory

```
$ cd /hv/guests/guestxxxx
```

- Start the guest

```
$ hv_guest_start -view
```

- Wait 30..60 sec.

You should now see the guest at the display. If yes, you are **done!**

---

**Important:** If the screen **remains black** after 30..60 sec. please continue with the *next* chapter!

---

## 10 Update display driver

As the passthroughed display needs an appropriate display driver, please connect to the running Windows guest through **RDP** (In this guide the Windows guest PC-Name is DESKTOP-QKRP1FI) and run the Windows Update. Windows Update will find an appropriate display driver if available.

---

**Important:** If no appropriate display driver is found, please continue with the *Troubleshooting* section.

---

## 11 Troubleshooting

There are several points that can cause problems with Graphics Passthrough. Here are a few hints that can help solve these issues.

### 11.1 Windows guest

#### 11.1.1 Windows graphics driver

If the Windows guest screen remains black, it's usually due to a missing graphics driver in Windows guest. To install this, connect to the guest via RDP. In this guide the Windows guest PC-Name is DESKTOP-QKRP1FI. Replace it by the PC-Name of your Windows guest and connect it through the RDP client on your development PC. As *user/password* use the defined credentials at Windows guest install stage.

**Open your internet browser in the Windows guest and get the latest Intel graphic drivers:**

<https://downloadcenter.intel.com/product/80939/Graphics>

**OR**

<https://www.intel.com/content/www/us/en/download/19344/intel-graphics-windows-dch-drivers.html>

After installing the driver you should reboot the Windows guest. It could take some minutes until the screen will show up or RDP Connect will work.

#### 11.1.2 RAM size

Increase the RAM size of your Windows guest by changing the value of `ramsize` in the `guest_config.sh` file. At least 6 GB of RAM is recommended for the Windows guest (`ramsize=6144`).

```
# Major VM settings
export num_cpus=2          # Number of CPUs for the VM.
                            # Should not be higher than the number of_
                            # available physical CPUs
                            # At installation time, the number should be set_
                            # to one below the number of physical available CPUs!
export ramsize=6144        # RAM size of the physical machine.
export hdszie=50G          # Hard disk size.
                            # This value is only when creating a new hard disk_
                            # image.
                            # After creating the disk image, this value is_
                            # ignored.
```

### 11.1.3 vendor id

Search for `hv_vendor_id` in `/hv/bin/kvmguest_start.sh` file.

```
runvm_cmd="$guest_dir/$vm_procname -enable-kvm \
-machine pc-i440fx-2.0 -smp cpus=$num_cpus,cores=$num_cpus,threads=1,
→sockets=1 -m $ramsize -device virtio-balloon,addr=$virtio_balloon_pci_
→addr \
-monitor unix:$qemu_mon_sock,server,nowait \
-cpu host,hv_vendor_id=aconitis,hv_relaxed,hv_spinlocks=0x1fff,hv_vapic,
→hv_time,pmu=$PMU \
-device virtio-serial,addr=$virtio_serial_pci_addr \
```

Remove the `hv_vendor_id` entry.

```
runvm_cmd="$guest_dir/$vm_procname -enable-kvm \
-machine pc-i440fx-2.0 -smp cpus=$num_cpus,cores=$num_cpus,threads=1,
→sockets=1 -m $ramsize -device virtio-balloon,addr=$virtio_balloon_pci_
→addr \
-monitor unix:$qemu_mon_sock,server,nowait \
-cpu host,hv_relaxed,hv_spinlocks=0x1fff,hv_vapic,hv_time,pmu=$PMU \
-device virtio-serial,addr=$virtio_serial_pci_addr \
```

## 11.2 Linux guest

### 11.2.1 HDMI not working

For Intel 8th-generation processors with integrated UHD 630 graphics and HDMI interface, there may be instances where the HDMI output is not recognized during graphics passthrough, resulting in no display on the screen. To verify if this issue is occurring, generate a `dmesg` output in an SSH shell on the Linux guest:

```
$ sudo dmesg | grep -i i915
i915 0000:00:02.0: [drm] VT-d active for gfx access
[drm] Initialized i915 1.6.0 20230929 for 0000:00:02.0 on minor 0
i915 0000:00:02.0: [drm] Cannot find any crtcs or sizes
```

In this case, the HDMI interface can be enabled in grub of the Linux guest.

First, start the host in Hypervisor mode to determine the correct interface using `xrandr`:

```
$ xrandr
Screen @: minimum 320 x 200, current 1280 x 720, maximum 16384 x 16384
DP-1 disconnected (normal left inverted right x axis y axis)
HDMI-1 disconnected (normal left inverted right x axis y axis)
DP-2 disconnected (normal left inverted right x axis y axis)
HDMI-2 connected 1280x720+0+@ (normal left inverted right x axis y axis) ↗
→7@08mm x 398mm
```

Start the Ubuntu guest in graphics passthrough and open a ssh terminal. Determine the appropriate HDMI entry needed later in the grub cmdline.

```
$ ls -l /sys/class/drm/card0*
lrwxrwxrwx 1 root root 0 Nov  8 10:12 /sys/class/drm/card0 -> ../../
  ↳devices/pci0000:00/0000:00:02.0/drm/card0
lrwxrwxrwx 1 root root 0 Nov  8 10:12 /sys/class/drm/card0-DP-1 -> ../../
  ↳devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-1
lrwxrwxrwx 1 root root 0 Nov  8 10:12 /sys/class/drm/card0-DP-2 -> ../../
  ↳devices/pci0000:00/0000:00:02.0/drm/card0/card0-DP-2
lrwxrwxrwx 1 root root 0 Nov  8 10:12 /sys/class/drm/card0-HDMI-A-1 -> ../../
  ↳devices/pci0000:00/0000:00:02.0/drm/card0/card0-HDMI-A-1
lrwxrwxrwx 1 root root 0 Nov  8 10:12 /sys/class/drm/card0-HDMI-A-2 -> ../../
  ↳devices/pci0000:00/0000:00:02.0/drm/card0/card0-HDMI-A-2
```

Edit the file `/etc/default/grub` and search the Linux kernel boot line `GRUB_CMDLINE_LINUX_DEFAULT`. Add the parameter `video=HDMI-A-2:e` to the end of this line.:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash video=HDMI-A-2:e"
```

Save the file, update grub, and reboot:

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

The RTOSVisor can now be booted in Graphics Passthrough mode, and the HDMI output should work correctly.

## 11.3 General issues

- **Unplug mouse or keyboard**

If you unplug and replug the mouse or keyboard during graphics passthrough, they will no longer be recognized and won't function properly. To avoid this, you can pass through the mouse and keyboard to the guest using USB passthrough. However, in this case, the device must always be plugged into the same USB port.

Please refer to the [Hypervisor Manual](#) section on Windows/Linux USB guest access for information on how to get the appropriate hostbus and hostport.

In the `usr_guest_config.sh` file:

- Add entries for `USB_HOST_ADAPTER1_PASSTHROUGH`.
- Place a comment symbol before the lines with `vga_gpt_kbd_event_device` and `vga_gpt_mouse_event_device`.

```
# 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!
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=1, hostport=5"
```

(continues on next page)

(continued from previous page)

```
# UEFI support
export uefi_bios=1

# allow dynamically adjust the desktop (from Windows to full screen
# mode)
export enable_vga_spice=1

# graphics passthrough (see hypervisor manual for more information)
export enable_vga_gpt=1      # set to 1 to enable graphics
#passthrough
export vga_gpt_bdf=00:02.0    # PCI bus, device function values for
#the graphics card, use the lspci command to determine
#export vga_gpt_kbd_event_device="" # keyboard event, determine
#using ls -la /dev/input/by-id | grep -event- (search for the
#appropriate keyboard event number)
#export vga_gpt_mouse_event_device="" # keyboard event, determine
#using ls -la /dev/input/by-id | grep -event- (search for the
#appropriate mouse event number)
```