Virtualization Guide for Intel® Data Center GPU Flex Series

Setup Reference Guide

January 2023

Revision 1.5

Intel Confidential

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document. The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Warning: Altering PC clock or memory frequency and/or voltage may (i) reduce system stability and use life of the system, memory and processor; (ii) cause the processor and other system components to fail; (iii) cause reductions in system performance; (iv) cause additional heat or other damage; and (v) affect system data integrity. Intel assumes no responsibility that the memory, included if used with altered clock frequencies and/or voltages, will be fit for any particular purpose. Check with memory manufacturer for warranty and additional details.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

Revision History

Revision Number	Description	Date
1.5	• Updated the Windows VM driver installation fix	February 2023
1.0	• Updated the kernel parameter for Single Root I/O Virtualization (SR-IOV)	January 2023
0.5	• Initial release of the document.	July 2022

1 Introduction

The purpose of this document is to provide the user with fundamentals of virtualization at the platform, system level and to cover the various alternatives for graphics device virtualization setting-up on the Host server with the Intel® Data Center GPU Flex Series Graphics Accelerator Card.

This document covers the implementation of graphics device virtualization approaches like Passthrough and Single Root I/O Virtualization (SR-IOV).

Intel Passthrough is one flavor of graphics virtualization approaches based on Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d) and some special graphics related configuration. This flavor allows direct assignment of an entire GPU prowess to a single user, passing the native driver capabilities through the hypervisor without any limitations.

The SR- IOV interface is an extension to the PCI Express* (PCIe*) Specification. SR-IOV allows a device, such as a GPU card or network adapter, to separate access to its resources among various PCIe* hardware functions.

2 System Setup

2.1 Intel® Data Center GPU Flex Series Card Setup

For Intel® Data Center GPU Flex Series card deployment, firmware, and driver installation on the system, see Intel® Data Center GPU Flex Series

2.2 Host BIOS Configuration

Notes: This setting is on CYP server, SMC or Wilson City may have different location of the menu options in BIOS setting

- Advanced -> Chipset Configuration -> NorthBridge -> IIO Configuration -> Intel VT for Directed I/O (VT-d) -> Enable
- Advanced -> Chipset Configuration -> NorthBridge -> IIO Configuration -> PCI-E ASPM Support (Global) -> No
- Advanced -> Chipset Configuration -> NorthBridge -> IIO Configuration -> IIO eDPC Support -> Disable
- Advanced -> PCIe/PCI/PnP Configuration -> Above 4G Decoding -> Enabled
- Advanced -> PCIe/PCI/PnP Configuration -> SR-IOV Support -> Enable
- Advanced -> PCIe/PCI/PnP Configuration -> ARI Support -> Enable
- Advanced -> PCIe/PCI/PnP Configuration -> MMCFG Base -> Auto
- Advanced -> PCIe/PCI/PnP Configuration -> MMIO High Base -> 40T
- Advanced -> PCIE/PCI/PnP Configuration -> MMIO High Granularity Size -> 1024G

3 Kernel-Based Virtual Machine (KVM) Host Setup

3.1 Library Dependency

1. In this document, **Ubuntu* 22.04** is used as host. You can use other operating systems such as Ubuntu* 20.04 or Red Hat Enterprise Linux* (RHEL*) 8.5 or 8.6. Install the following software packages to set up the host environment:

```
$ sudo apt-get install git vim socat autoconf xtightvncviewer
tightvncserver x11vnc uuid-runtime uuid qemu-kvm ovmf libvirt-daemon-
system libvirt-clients bridge-utils virtinst virt-manager
```

2. To fix the Windows VM driver issue, please remove below package as a workaround.

```
$ sudo apt-get purge -y swtpm swtpm-tools
```

3. With all the packages installed, enable, and start the Libvirt daemon:

```
$ sudo systemctl enable --now libvirtd

$ sudo systemctl start libvirtd

$ sudo usermod -aG kvm $USER

$ sudo usermod -aG libvirt $USER
```

3.2 Build Kernel

1. To install the `repositories.intel.com/graphics` package repository, add the following to your Ubuntu* installation:

```
$ sudo apt-get install -y gpg-agent wget

$ wget -qO - https://repositories.intel.com/graphics/intel-graphics.key
  | sudo gpg --dearmor --output /usr/share/keyrings/intel-graphics.gpg

$ echo 'deb [arch=amd64 signed-by=/usr/share/keyrings/intel-graphics.gpg]
https://repositories.intel.com/graphics/ubuntu jammy flex' | \
  sudo tee  /etc/apt/sources.list.d/intel.gpu.jammy.list
```

2. Install the `linux-image-5.15.0-48-generic` kernel:

```
$ sudo apt-get update && sudo apt-get install -y linux-image-5.15.0-48-
generic
```

3.3 Grub Update

1. Update the kernel parameter to the `/etc/default/grub` using the following command:

```
$ GRUB_CMDLINE_LINUX="i915.max_vfs=31 intel_iommu=on iommu=pt"
```

2. Run the following commands to update the grub configuration file with changes made:

```
$ sudo update-grub

$ sudo reboot
```

3. Make sure that you are using `5.15.0-48-generic`. Then, proceed with the following instructions:

```
$ uname -r

  5.15.0-48-generic
```

4. Install the dkms and kernel header files:

```
$ sudo apt-get update

$ sudo apt-get install gawk \
    dkms \
    linux-headers-$(uname -r) \
    libc-dev

$ sudo apt-get install -y intel-platform-vsec-dkms intel-platform-
cse-dkms
$ sudo apt-get install -y intel-i915-dkms intel-fw-gpu
```

5. Install the run-time packages:

```
$ sudo apt-get install -y \
  intel-opencl-icd intel-level-zero-gpu level-zero \
  intel-media-va-driver-non-free libmfx1 libmfxgen1 libvpl2 \
  libegl-mesa0 libegl1-mesa libegl1-mesa-dev libgbm1 \
  libgl1-mesa-dev libgl1-mesa-dri \
  libglapi-mesa libgles2-mesa-dev libglx-mesa0 \
  ibigdgmm12 libxatracker2 mesa-va-drivers \
  mesa-vdpau-drivers mesa-vulkan-drivers va-driver-all
```

6. Install the Developer packages (optional):

```
$ sudo apt-get install libigc-dev intel-igc-cm libigdfcl-dev
libigfxcmrt-dev \
   level-zero-dev
```

7. Reboot the system for these changes to take effect:

```
 $ sudo reboot
```

8. Configure the Permissions to access the GPU capabilities:

```
$ stat -c "%G" /dev/dri/render*

$ groups ${USER}

$ sudo gpasswd -a ${USER} render

$ newgrp render
```

9. Set CPU Frequency to **performance** mode:

```
$ for i in $(seq 0 $(($(nproc)-1))); do \
  echo performance | sudo tee \
  /sys/devices/system/cpu/cpu$i/cpufreq/scaling_governor; \
  done
```

3.4 KVM Host Check

1. GPU device status:

```
$ ls -l /dev/dri/
 total 0
drwxr-xr-x   2 root root         100 Dec 11 22:35 by-path
crw-rw----+  1 root video   226,    0 Dec 11 22:35 card0
crw-rw----+  1 root video   226,    1 Dec 11 22:35 card1
crw-rw----+  1 root render 226, 128 Dec 11 22:35 renderD128

$  lspci | grep Display

     b3:00.0 Display controller: Intel Corporation Device 56c0 (rev 08)
```

2. Check the virtualization status:

```
 $ lscpu | grep -o vmx

    Vmx

 $ kvm-ok

    INFO: /dev/kvm exists
    KVM acceleration can be used

 $ sudo dmesg | grep -E 'Passthrough|SR-IOV'

[    4.486823] iommu: Default domain type: Passthrough (set via kernel
command line)
      8.976310] i915 0000:b3:00.0: Running in SR-IOV PF mode

  $ virt-host-validate | grep QEMU

    QEMU: Checking for hardware virtualization : PASS
    QEMU: Checking if device /dev/kvm exists   : PASS
    QEMU: Checking if device /dev/kvm is accessible: PASS
    QEMU: Checking if device /dev/vhost-net exists : PASS
    QEMU: Checking if device /dev/net/tun exists : PASS
    QEMU: Checking for cgroup 'cpu' controller support: PASS
    QEMU: Checking for cgroup 'cpuacct' controller support: PASS
    QEMU: Checking for cgroup 'cpuset' controller support : PASS
    QEMU: Checking for cgroup 'memory' controller support : PASS
    QEMU: Checking for cgroup 'devices' controller support : PASS
    QEMU: Checking for cgroup 'blkio' controller support : PASS
    QEMU: Checking for device assignment IOMMU support : PASS
    QEMU: Checking if IOMMU is enabled by kernel       : PASS
    QEMU: Checking for secure guest support            : WARN
```

4 Guest VMs for Passthrough Setup

4.1 Installing Windows Server 2022 VM

Run the following steps as a root user:

1. Download Windows* 2022 ISO from the Microsoft* Evaluation Center.
2. To enable the Windows* VM to work with KVM or QEMU*, you need to install the **virtIO-win** driver.

```
$ wget -c https://fedorapeople.org/groups/virt/virtio-win/direct-
downloads/archive-virtio/virtio-win-0.1.215-2/virtio-win-0.1.215.iso
```

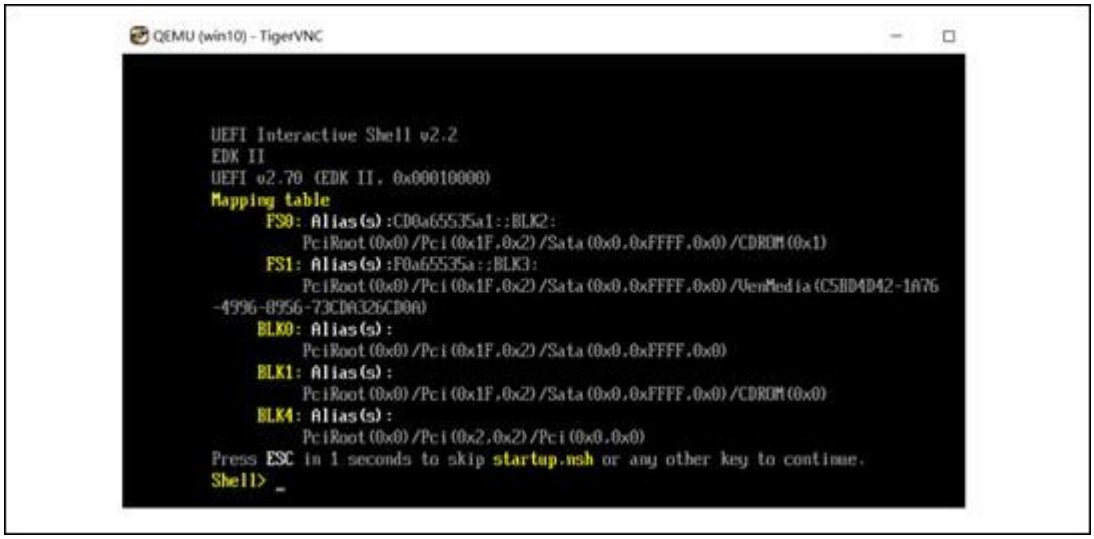
3. Run the **virt-install** tool to install the Windows* 2022 VM.

   VM configuration: 8 core/8 Gb RAM/80 Gb disk.

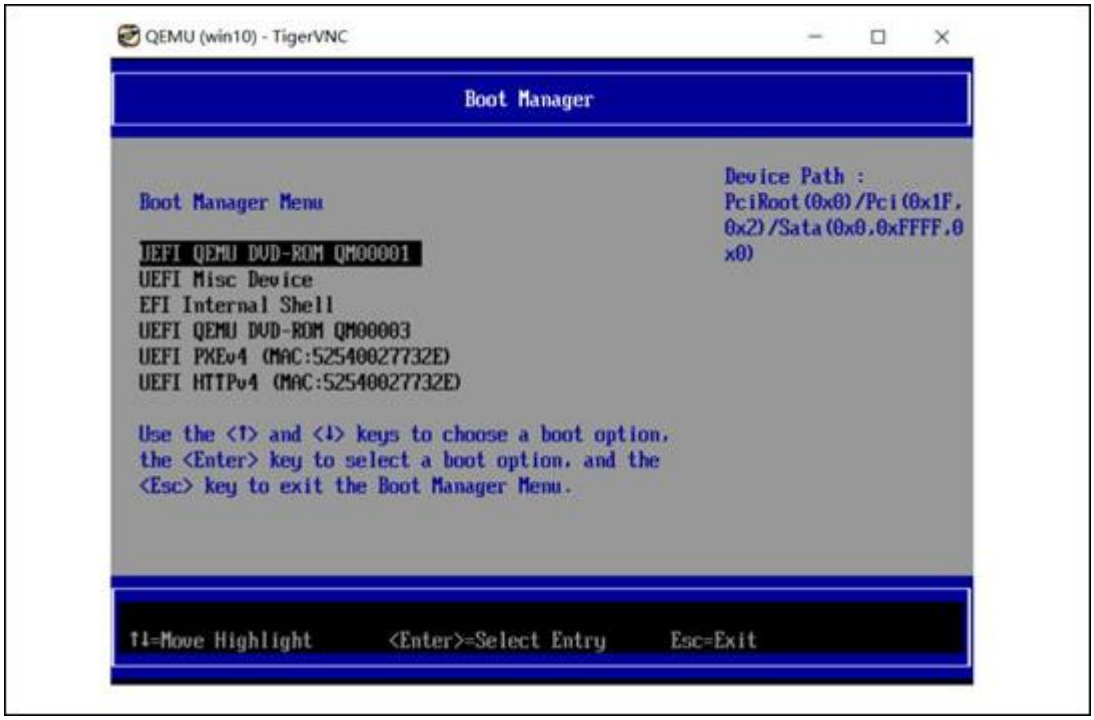
```
$ virsh destroy win2022
$ virsh undefine --nvram win2022
$ virsh vol-delete win2022.qcow2 --pool default
$ virt-install \
   --boot uefi \
   --name win10 \
   --vcpus 8 \
   --cpu host-passthrough \
   --ram 8192 \
   --memballoon none \
   --features
   hyperv.relaxed.state=off,hyperv.reset.state=off,hyperv.spinlocks.stat
   e=off,hyperv.synic.state=off,hyperv.vapic.state=off \
   --clock offset='localtime',hypervclock_present=no \
   --clock offset='localtime' \
   --network network=default \
   --graphics vnc,listen=0.0.0.0,port=5901 \
   --video=qxl \
   --disk pool=default,size=80,format=qcow2,bus=virtio \
   --cdrom=/var/lib/libvirt/images/Windows2022_EVAL_x64FRE_en-us.iso \
   --disk /var/lib/libvirt/images/virtio-win-
   0.1.215.iso,device=cdrom \ --boot cdrom,hd \
   --os-variant win10 \
   --input tablet
```

Note: KVM supports Network Address Translation (NAT) by default. To connect VMs directly to the host server network, you need to create a bridge. See Appendix A to configure the bridge networking.

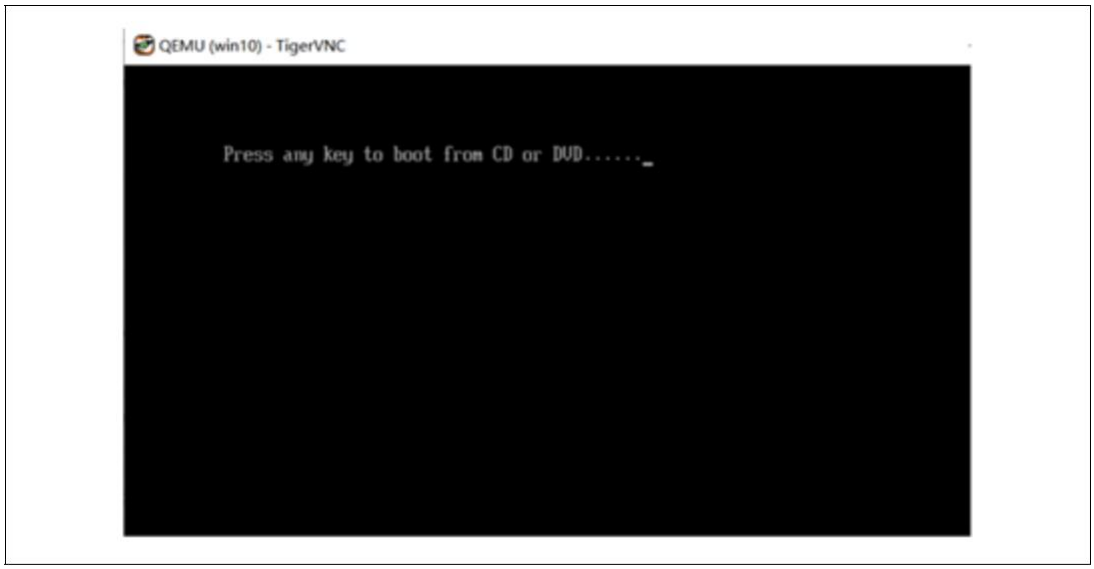
4. You can use VNC client to start the installation. Connect to port 5901 from the host machine.

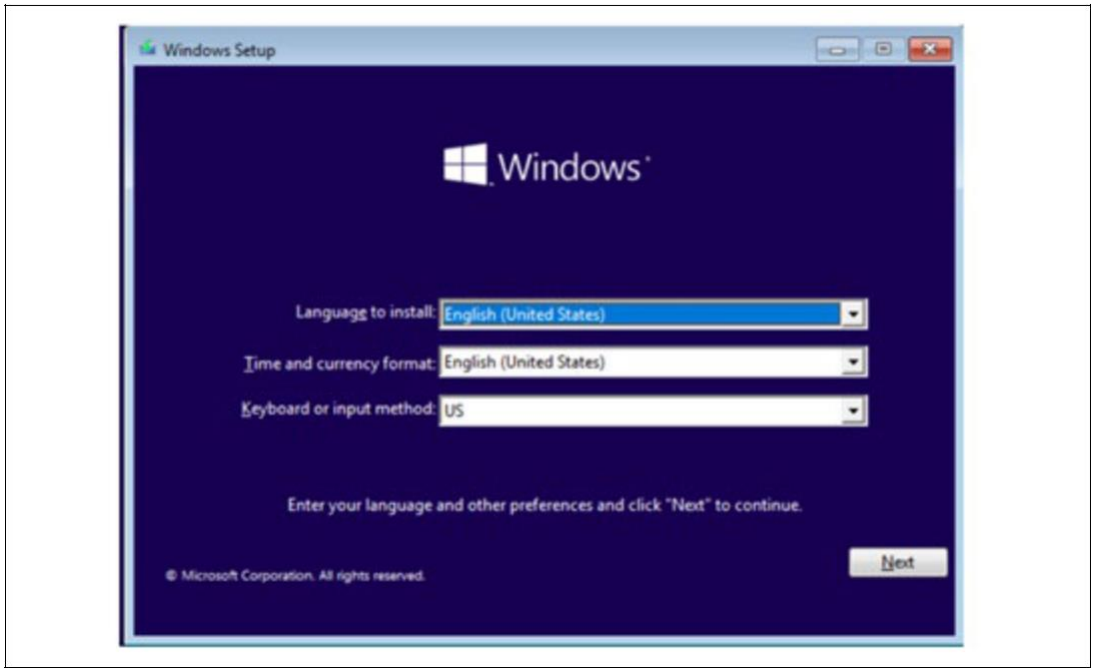


5. Type **Exit** and go to the **Boot Manager Menu**. Select the first option.

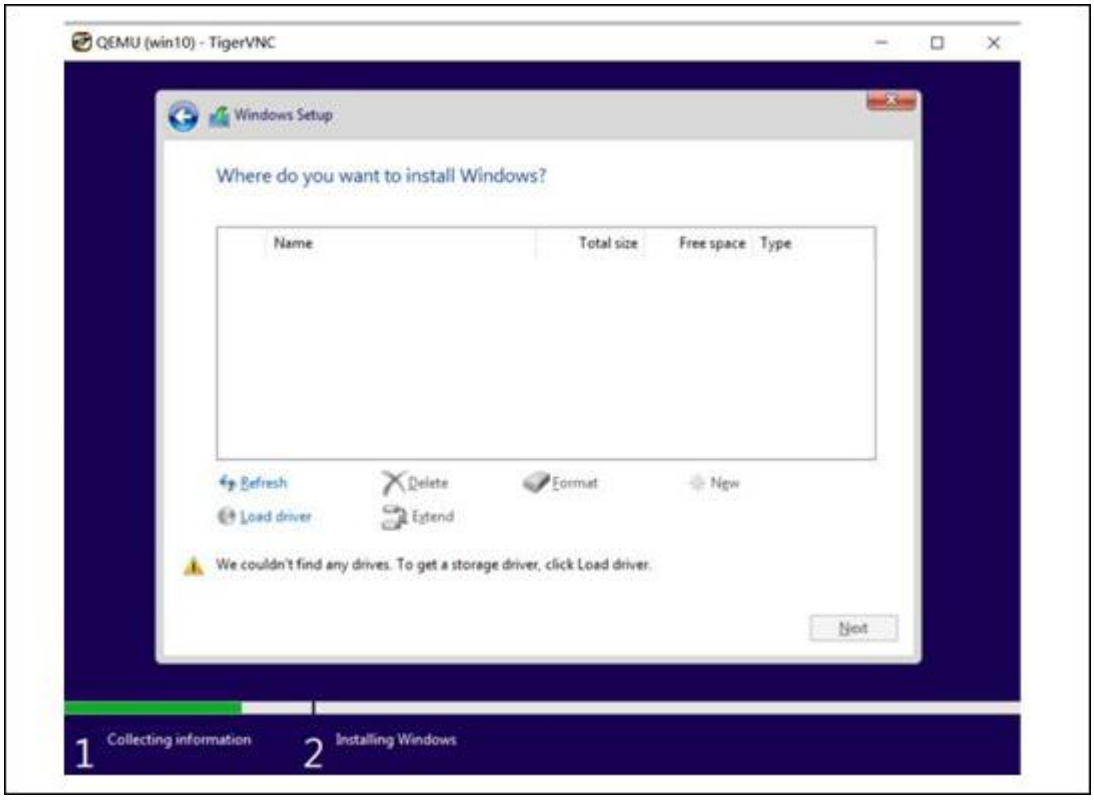


6. Press the spacebar.

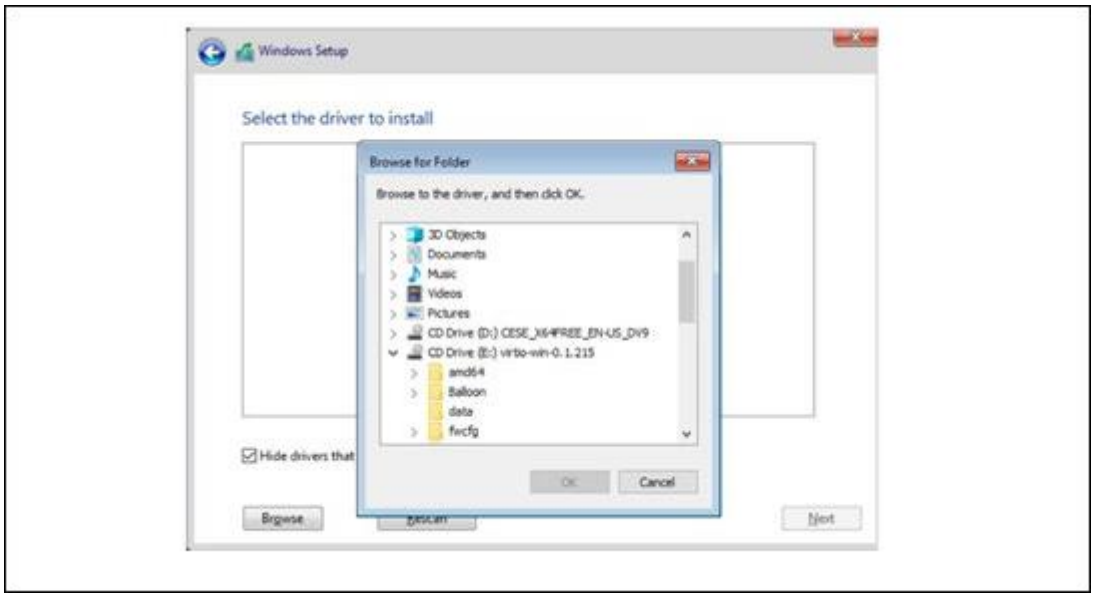


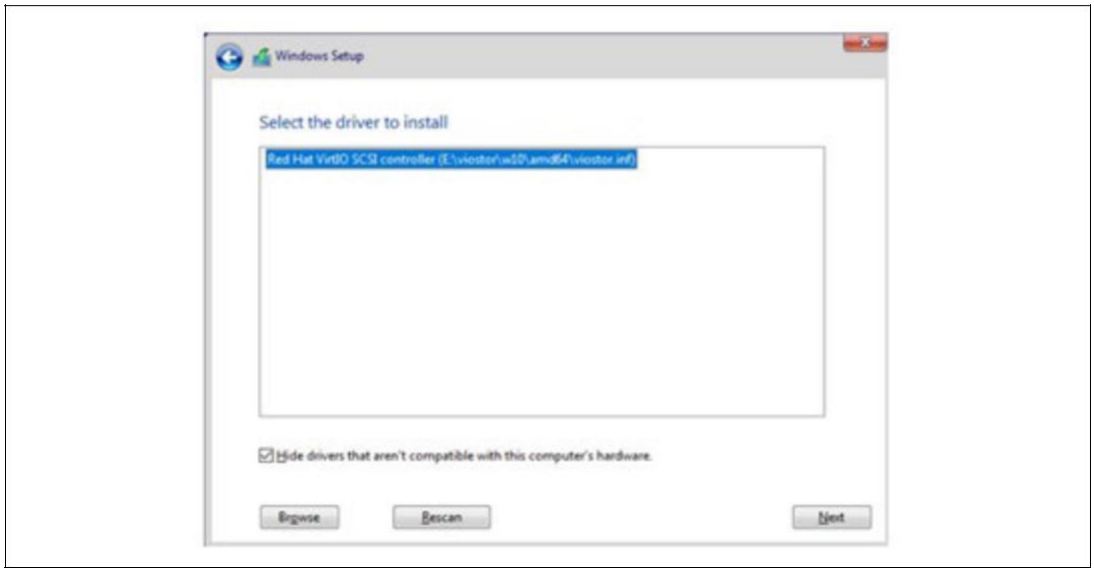


7. By default, Windows* 2022 does not support QEMU* Copy on Write (QCOW2).

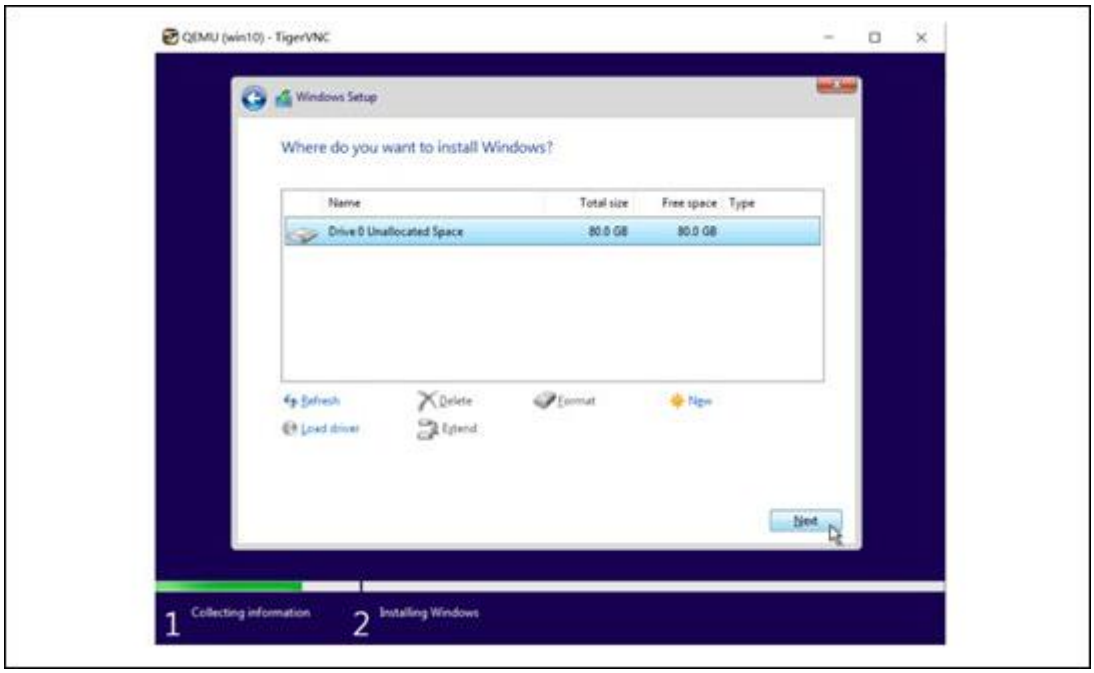


8. Install the **virtIO** driver manually.





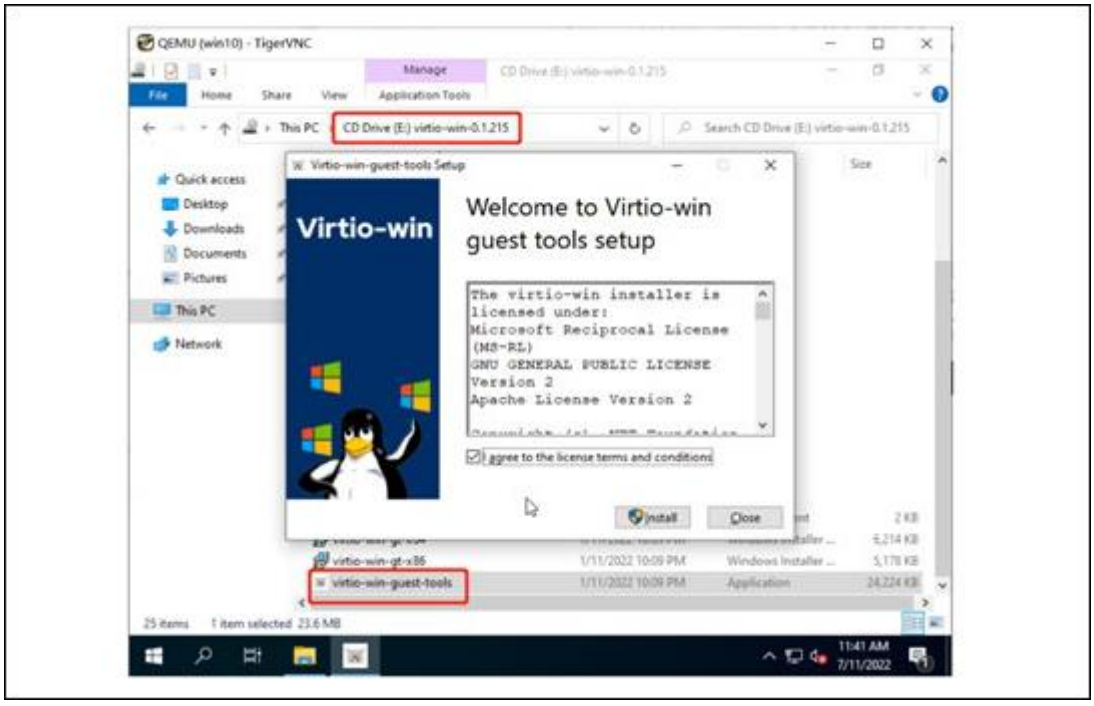
9. Now you can see the disk.



10. Continue with the rest of steps to complete the installation.

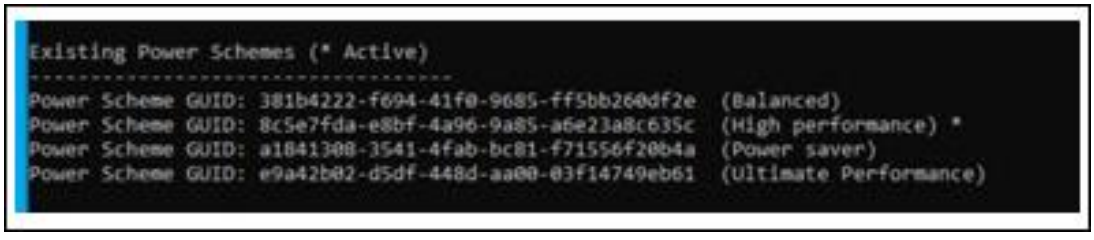
4.2 Tuning the Windows* 2022 VM

1. Install the **virtio-win-guest-tool** driver to get the best mouse and keyboard interaction with the VM:



2. Set the OS Power Scheme to **Performance**. Open the command line as an Administrator and execute the following command:

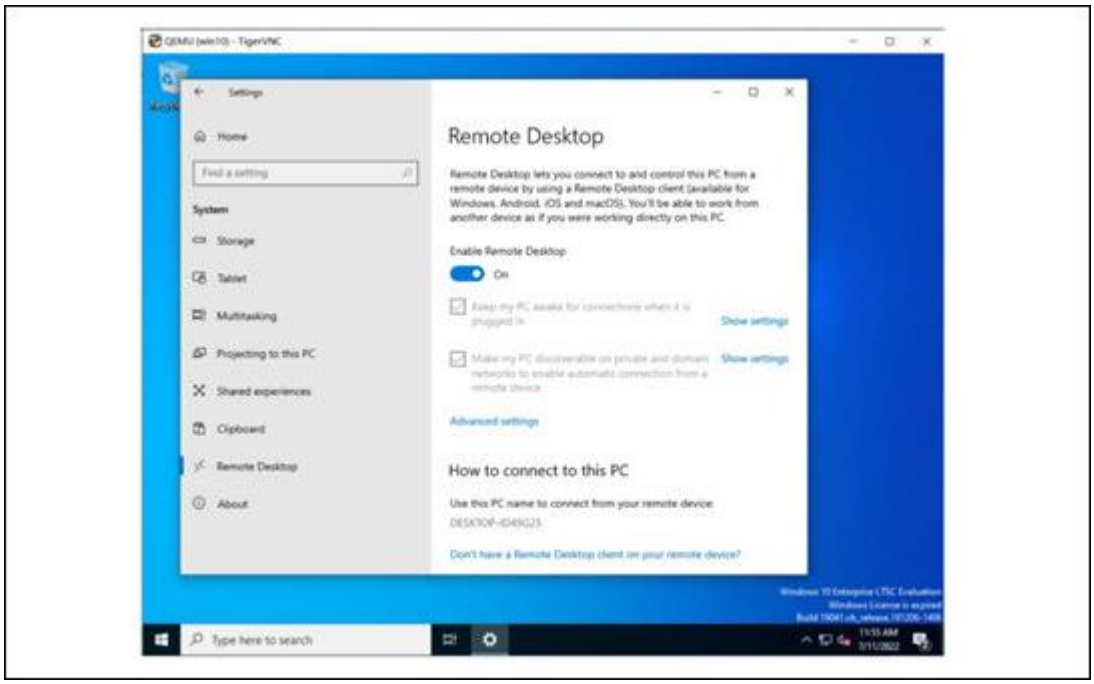
```
powercfg -setactive 8c5e7fda-e8bf-4a96-9a85-a6e23a8c635c

powercfg list
```



3. Stop the Firewall Services.

```
netsh advfirewall set allprofiles state

off netsh advfirewall show allprofiles
```

4. Enable Remote Desktop (RDP) to connect.



4.3 Enable GPU Passthrough on Windows Server 2022 VM

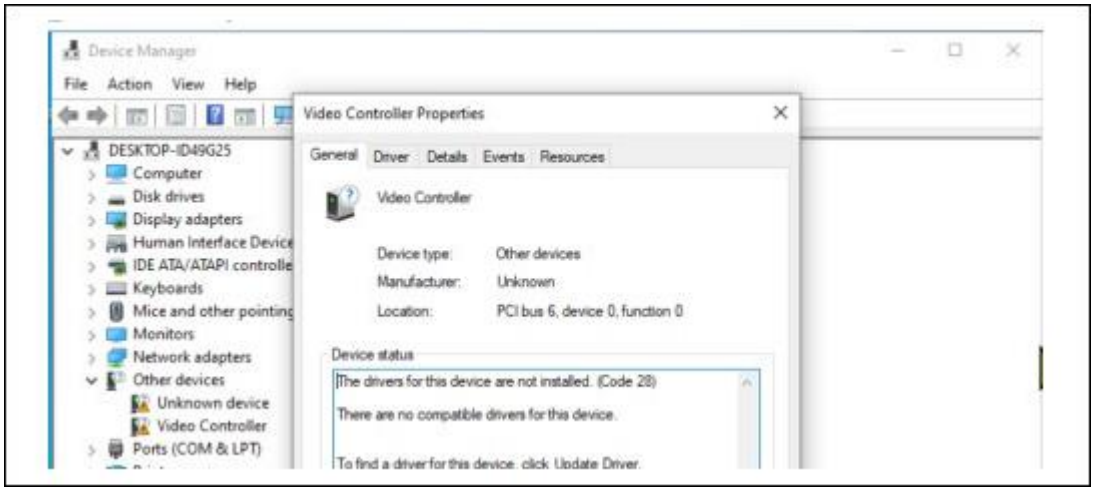
1. Query the Intel® Data Center GPU on host.

```
$ lspci | grep Dis

b3:00.0 Display controller: Intel Corporation Device 56c0 (rev 08)
```

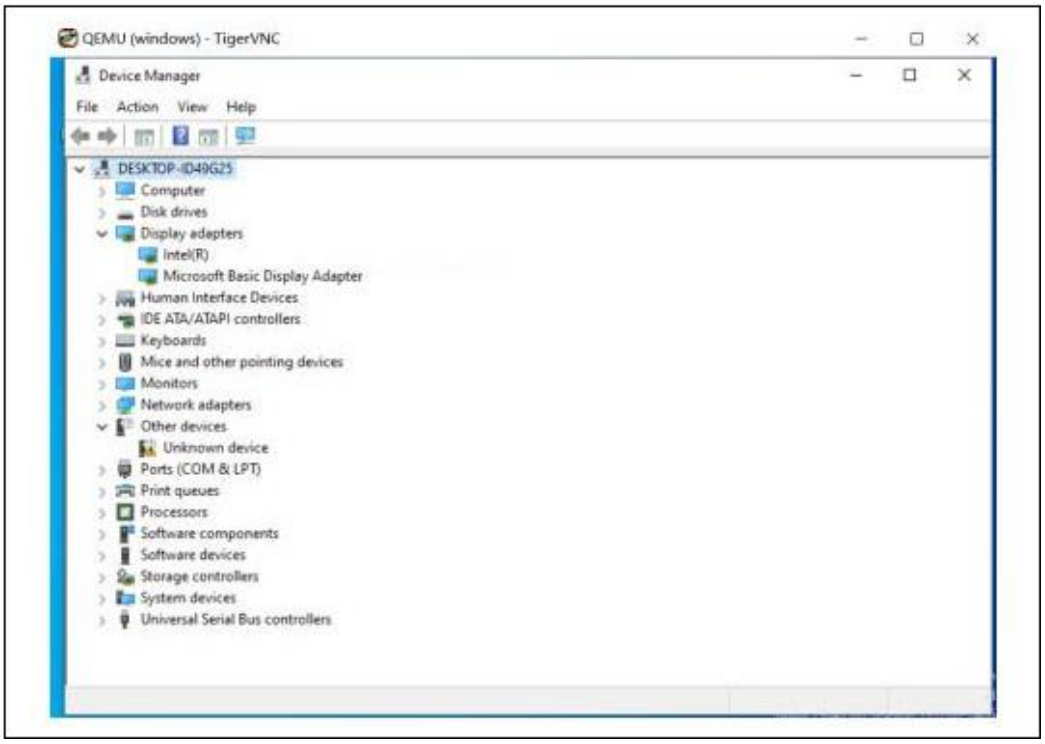
2. Create an XML file to enter the GPU card details and attach to the VM.

```
# virsh destroy win2022

# cat > vGPU.xml << EOF
<hostdev mode='subsystem' type='pci' managed='yes'>
<driver name='vfio'/>
<source>
<address domain='0x0' bus='0xb3' slot='0x0' function='0x0'/>
</source>
</hostdev>
EOF

# virsh attach-device win2022 vGPU.xml –config

# virsh start win2022
```

3. You can see the **Video Controller** on the **Device Manager**.



4. See Section 2.1 to download the drivers for the GPU card. After driver installation, the GPU card will be listing out under **Display adapters**.



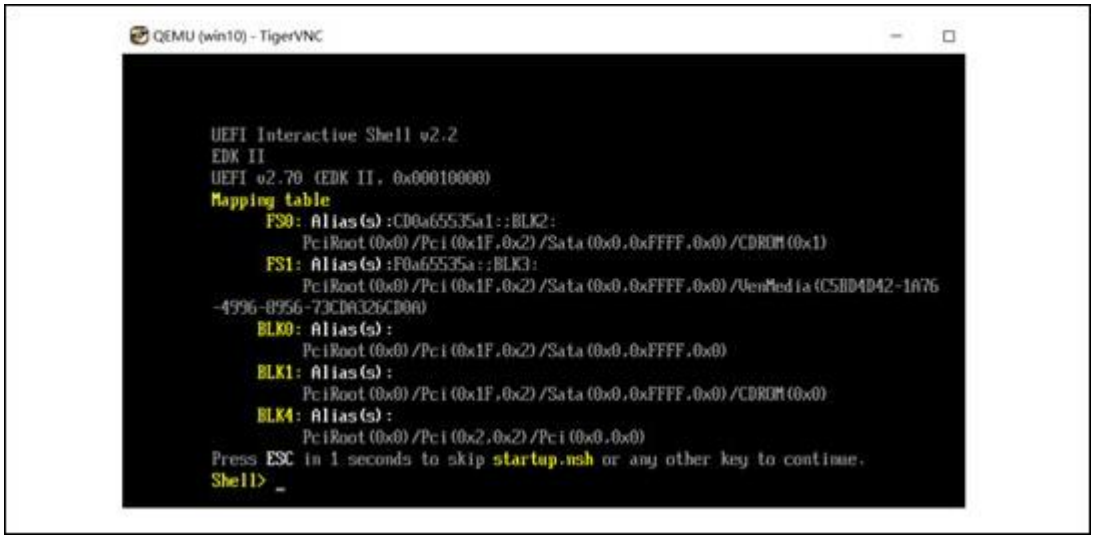
4.4 Installing Ubuntu* 22.04 VM

1. Download Ubuntu* 22.04 ISO from the Download Ubuntu* Desktop.
2. Run the **virt-install** tool to install Ubuntu* VM.

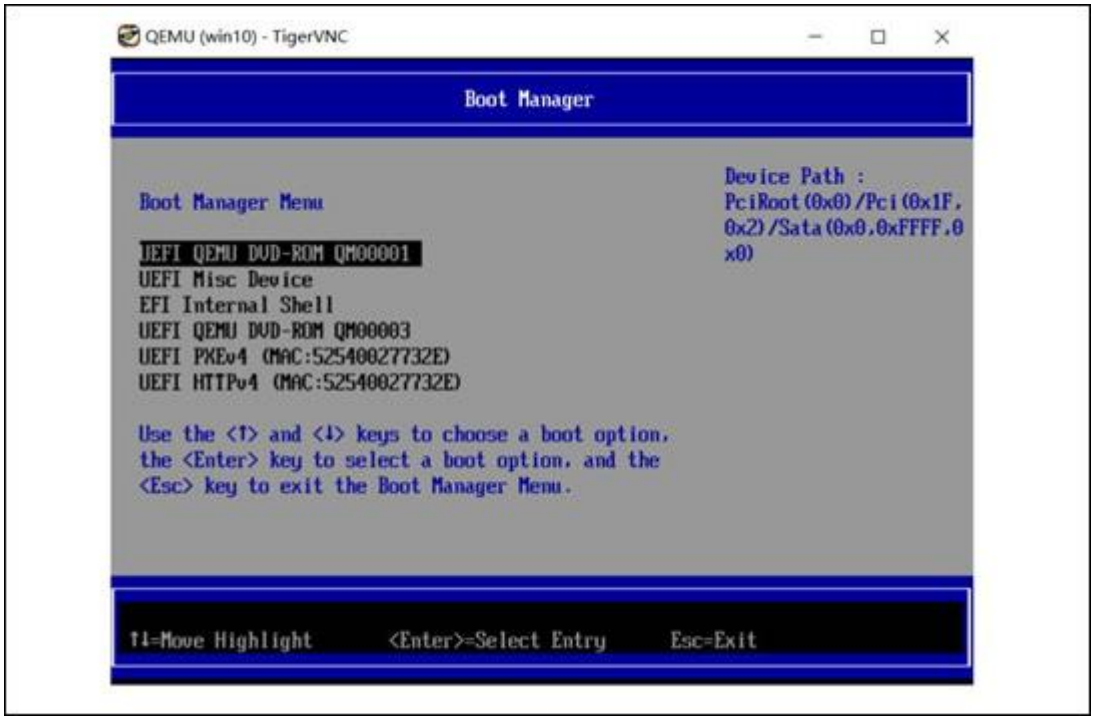
   VM configuration: 8 core/8 Gb RAM/80 Gb disk.

```
virt-install \
--boot uefi \
--name ubuntu \
--vcpus 8 \
--cpu host-passthrough \
--ram 8192 \
--memballoon none \
--clock offset='localtime' \
--network network=default \
--graphics vnc,listen=0.0.0.0,port=5901 \
--video=qxl \
--disk pool=default,size=80,format=qcow2,bus=virtio \
--cdrom=/var/lib/libvirt/images/ubuntu22.04.iso \
--boot cdrom,hd \
--input tablet
```

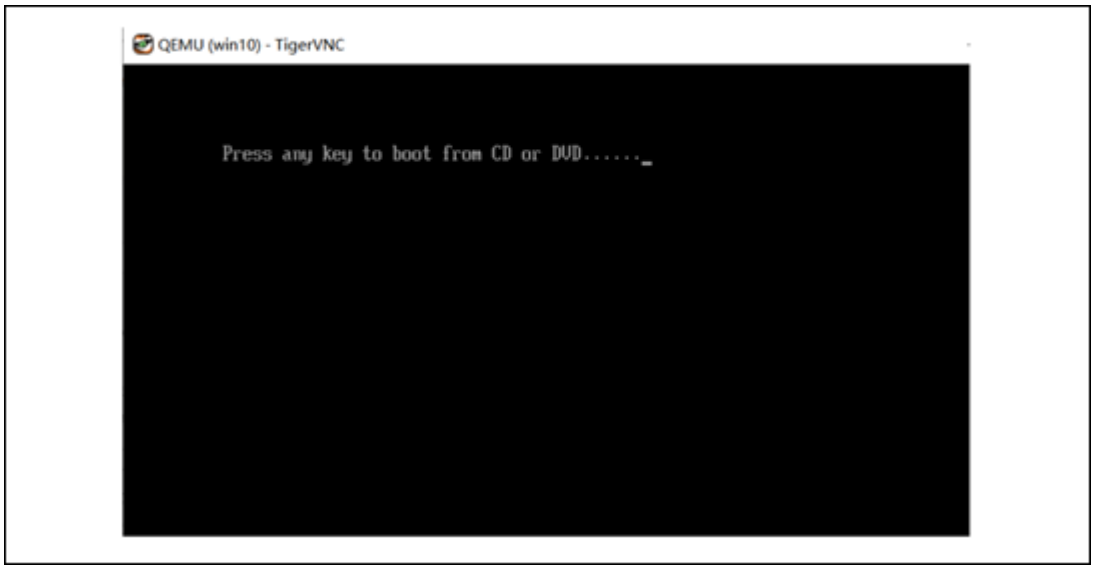
3. You can use VNC* client to start the installation. Connect to **port 5901** from the host machine.



4. Type **Exit** and go to the **Boot Manager Menu**. Select the first option.

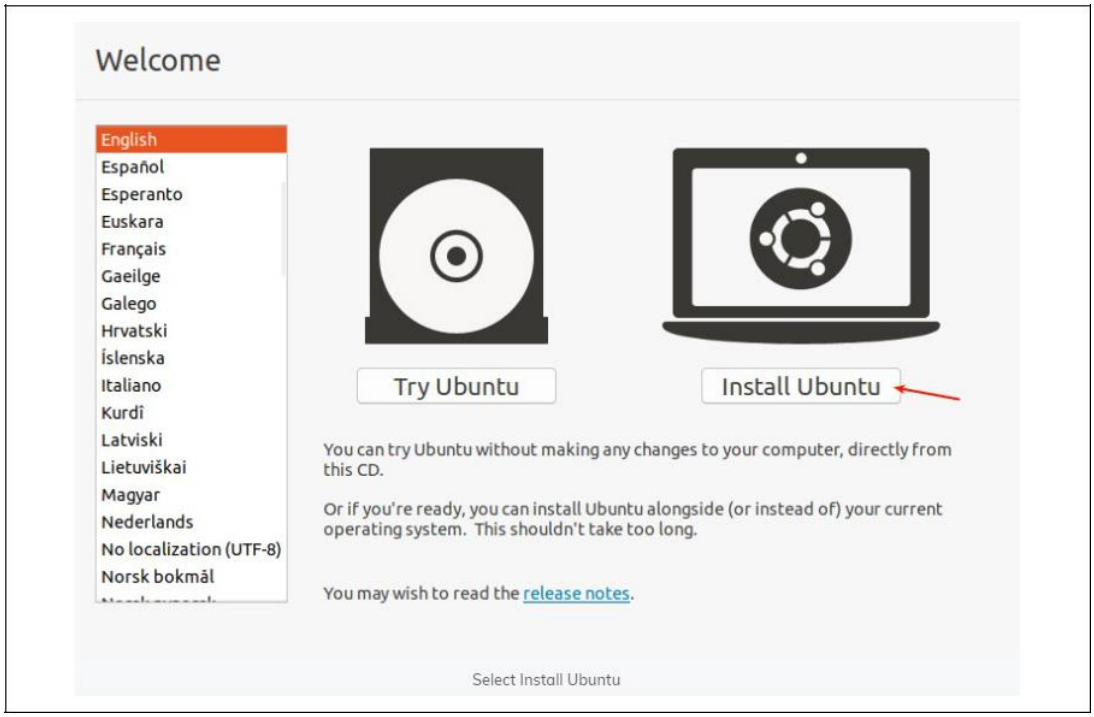


5. Press the spacebar.



6. Once the system boots up with bootable media, the following screen will appear:

7. Click the **Install Ubuntu** option to start the installation process:



8. Continue with the rest of steps to complete the installation.

9. Set CPU Frequency to **performance** mode:

```
$ for i in $(seq 0 $(($(nproc)-1))); do \
  echo performance | sudo tee \
  /sys/devices/system/cpu/cpu$i/cpufreq/scaling_governor; \
  done
```

4.5 Enable GPU Passthrough on Ubuntu* 22.04 VM

1. Query the Intel® Data Center GPU on host.

```
$ lspci | grep Dis

b3:00.0 Display controller: Intel Corporation Device 56c0 (rev 08)
```

2. Create an XML file to enter the GPU card details and attach to the VM.

```
# virsh destroy ubuntu

# cat << EOF > vGPU.xml
 <hostdev mode='subsystem' type='pci' managed='yes'>
<driver name='vfio'/>
<source>
<address domain='0x0' bus='0xb3' slot='0x0' function='0x0'/>
</source>
</hostdev>
EOF

# virsh attach-device ubuntu vGPU.xml --config

# virsh start ubuntu
```

3. You can see the **Video Controller** on the Ubuntu* VM.

```
# lspci | grep Dis

00:06.0 Display controller: Intel Corporation Device 56c0 (rev 08)
```

4. See Section 2.1 to download the drivers for the GPU card.

5 Guest VMs for SR-IOV Setup

5.1 Create vGPU

The SR-IOV and Passthrough are already enabled in the host, so you need to create the vGPU function on the physical GPU card.

Note: In this example, two VF (Virtual Function) functions are created. For more VF, you need to change the -vp parameter in the following script:

See Appendix B to create the vGPU files.

```
$ sudo ./vgpu_profiles.sh -cp card1 -vp V2

Provision From Config Files Script
Version:1.1 vgpu_profile script Version:2.3
No. of cards detected are: 1
devId=0x56C0
Identified ATSM_150 device
InCardPath=card1
InVgpuProfile=V2
No. of cards detected are: 1
Applying provisioning from vgpu_profile data sheet
inputs CardPath = card1
No. of cards detected are: 1
############################################################
Applying Config file name = config_files/ATSM/ATSM150_vfs.csv
############################################################
profileType=V
numOfVfsFromInVgpuProfileId=2
/sys/class/drm/card1/iov/pf/gt/exec_quantum_ms            =   1
/sys/class/drm/card1/iov/pf/gt/preempt_timeout_us            =  2000
/sys/class/drm/card1/iov/pf/gt/policies/sched_if_idle           =    0
/sys/class/drm/card1/iov/vf1/gt/exec_quantum_ms             =   16
/sys/class/drm/card1/iov/vf1/gt/preempt_timeout_us           =   32000
/sys/class/drm/card1/iov/vf2/gt/exec_quantum_ms             =   16
/sys/class/drm/card1/iov/vf2/gt/preempt_timeout_us           =   32000
```

```
############################################################
Applying Config file name = config_files/ATSM/ATSM150_int.csv
############################################################
profileType=V
numOfVfsFromInVgpuProfileId=2
/sys/class/drm/card1/iov/pf/gt/policies/engine_reset             =    0
/sys/class/drm/card1/iov/pf/gt/doorbells_quota            =    16
/sys/class/drm/card1/iov/pf/gt/policies/sample_period_ms             =   0
************************************************
*         Resources alloted to VF1         *
************************************************
/sys/class/drm/card1/iov/vf1/gt/doorbells_quota            =    120
/sys/class/drm/card1/iov/vf1/gt/contexts_quota            =    1024
/sys/class/drm/card1/iov/vf1/gt/lmem_quota             =   8053063680
/sys/class/drm/card1/iov/vf1/gt/ggtt_quota             =   2013265920
/sys/class/drm/card1/iov/vf1/gt/threshold/cat_error_count              =    0
/sys/class/drm/card1/iov/vf1/gt/threshold/doorbell_time_us              =    0
/sys/class/drm/card1/iov/vf1/gt/threshold/engine_reset_count              =    0
/sys/class/drm/card1/iov/vf1/gt/threshold/h2g_time_us             =    0
/sys/class/drm/card1/iov/vf1/gt/threshold/irq_time_us             =    0
/sys/class/drm/card1/iov/vf1/gt/threshold/page_fault_count              =    0
************************************************
*         Resources alloted to VF2         *
************************************************
/sys/class/drm/card1/iov/vf2/gt/doorbells_quota            =    120
/sys/class/drm/card1/iov/vf2/gt/contexts_quota            =    1024
/sys/class/drm/card1/iov/vf2/gt/lmem_quota             =   8053063680
/sys/class/drm/card1/iov/vf2/gt/ggtt_quota             =   2013265920
/sys/class/drm/card1/iov/vf2/gt/threshold/cat_error_count              =    0
/sys/class/drm/card1/iov/vf2/gt/threshold/doorbell_time_us              =    0
/sys/class/drm/card1/iov/vf2/gt/threshold/engine_reset_count              =    0
/sys/class/drm/card1/iov/vf2/gt/threshold/h2g_time_us             =    0
/sys/class/drm/card1/iov/vf2/gt/threshold/irq_time_us             =    0
/sys/class/drm/card1/iov/vf2/gt/threshold/page_fault_count              =    0
```

The new vGPU created with PCI ID b3:00.1 is the following:

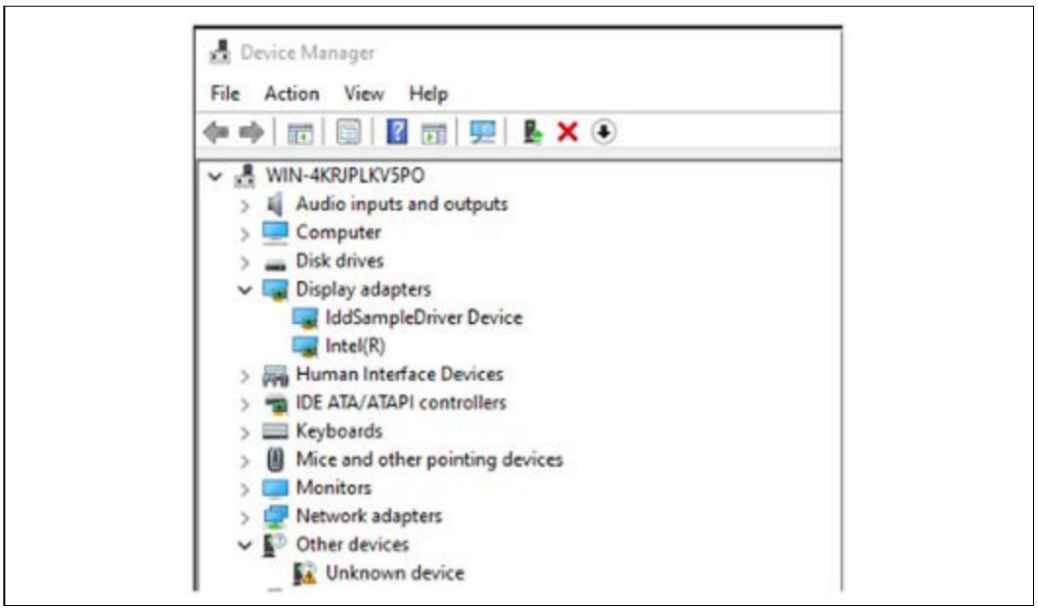
```
#  lspci | grep Display

b3:00.0 Display controller: Intel Corporation Device 56c0 (rev 08)
b3:00.1 Display controller: Intel Corporation Device 56c0 (rev 08)
b3:00.2 Display controller: Intel Corporation Device 56c0 (rev 08)
```

5.2 Assign vGPU to Windows Server 2022 VM

Note: VM creation/Tuning steps will be same as mentioned in the Passthrough mode.

Shut down the running Windows Server 2022 VM, remove the exiting Passthrough GPU, and assign the newly created vGPU.

```
$ virsh destroy win2022

$ virt-xml win2022 --remove-device --hostdev b3:00.0

$ virt-xml win2022 --add-device --hostdev b3:00.1,driver.name=vfio


$ virsh start win2022
```



5.3 Assign vGPU to Ubuntu* 22.04 VM

Shut down the running Ubuntu* 22.04 VM, remove the exiting Passthrough GPU, and assign the newly created vGPU.

```
$ virsh destroy ubuntu

$ virt-xml ubuntu --remove-device --hostdev b3:00.0

$ virt-xml ubuntu --add-device --hostdev b3:00.2,driver.name=vfio


$ virsh start ubuntu
```

```
root@analyticnode:~# lshw -c Video
  *-display
       description: VGA compatible controller
       product: ASPEED Graphics Family
       vendor: ASPEED Technology, Inc.
       physical id: 0
       bus info: pci@0000:02:00.0
       version: 41
       width: 32 bits
       clock: 33MHz
       capabilities: pm msi vga_controller cap_list
       configuration: driver=ast latency=0
       resources: irq:19 memory:92000000-92ffffff memory:93000000-9301ffff ioport:2000(size=128)
  *-display
       description: Display controller
       product: Intel Corporation
       vendor: Intel Corporation
       physical id: 0
       bus info: pci@0000:b3:00.0
       version: 08
       width: 64 bits
       clock: 33MHz
       capabilities: pciexpress msi pm bus_master cap_list rom
       configuration: driver=vfio-pci latency=0
       resources: iomemory:3ffe0-3ffdf iomemory:3fbc0-3fbbf irq:0 memory:3ffe00000000-3ffe00ffffff
```

Note: See Section 2.1 to download the drivers for the GPU card.

A Create a Network Bridge

The goal of this section is to cover the steps involved in creating a network bridge on the Ubuntu* enabling guest systems to share one or more of the host system's physical network connections while still allowing the guest and the host systems to communicate with each other.

A.1 Creating a Netplan Network Bridge

The creation of a network bridge on an Ubuntu* system using Netplan involves the addition of an entry to the `/etc/netplan/01-netcfg.yaml` or `/etc/netplan/00-installer-config.yaml` file. Using your preferred editor, open the file and add a bridges entry beneath the current content as follows (replacing eno1 with the connection name on your system):

```
network:
  ethernets:
    eno1:
      dhcp4: true
  version: 2

  bridges:
    br0:
      interfaces: [eno1]
      dhcp4: yes
```

Once the changes have been made, apply them using the following command:

```
# netplan apply
```

A.2 Creating a Network Manager Bridge

1. The first step in creating the network bridge is to add a new connection to the network configuration. This can be achieved using the **nmcli** tool.

```
# nmcli con add ifname br0 type bridge con-name br0
```

2. Once the connection has been added, a bridge secondary interface needs to be established between the physical device eno1 (secondary) and the bridge connection br0 (primary).

```
# nmcli con add type bridge-slave ifname eno1 master br0
```

3. Start up the bridge interface. When the following command executes, the connection will be lost when the eno1 connection is brought down. After waiting a few seconds, however, it should be possible to reconnect to the host once the br0 connection has been activated. It is better to run this command locally on the host.

```
# nmcli con up br0
```

4. Once the bridge is up and running, the connection list should now include both the bridge and the bridge-secondary connections.

```
# nmcli con show

NAME                UUID                                  TYPE      DEVICE
br0                 8416607e-c6c1-4abb-8583-1661689b95a9  bridge    br0
bridge-slave-eno1   43383092-6434-448f-b735-0cbea39eb38f  ethernet  eno1
virbr0              dffab88d-1588-4e69-8d1c-2148090aa5ee  bridge    virbr0
```

A.3 Declaring the KVM Bridged Network

1. Create a definition file for the bridge network named bridge.xml that reads as follows:

```
<network>
  <name>br0</name>
  <forward mode="bridge"/>
  <bridge name="br0" />
</network>
```

2. Use the file to define the new network:

```
# virsh net-define ./bridge.xml
```

3. Once the network has been defined, start it and, if required, configure it to autostart each time the system reboots:

```
# virsh net-start br0

# virsh net-autostart br0
```

4. List the networks to verify that the bridge network is now accessible within the KVM environment:

```
# virsh net-list –all

 Name                 State      Autostart     Persistent
----------------------------------------------------------
 br0                  active     yes           yes
 default              active     yes           yes
```

A.4 Using a Bridge Network in a VM

1. To create a VM that makes use of the bridge network, use the `virt-install --network` option and specify the br0 bridge name:

```
# virt-install --name MyFedora --memory 1024 --disk
path=/tmp/myFedora.img,size=10 --network network=br0 --os-variant
fedora28 --cdrom /home/demo/Downloads/Fedora-Server-dvd-x86_64-29-
1.2.iso
```

2. To modify an existing VM so that it uses the bridge, use the `virsh edit` command:

```
# virsh edit GuestName
```

3. To change from the **default** virtual network to **Br0**, locate the `<interface>` section of the file, which will read as follows for a NAT-based configuration:

```
<interface type='network'>
      <mac address='<your mac address here>'/>
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x01' slot='0x00'
function='0x0'/>
</interface>
```

4. If the VM is already running, the change will not take effect until it is restarted.

B vGPU Script

```
$ mkdir vgpu_profiles && cd vgpu_profiles

$ mkdir config_files && cd config_files

$ mkdir ATSM && cd ATSM

$ cat >> ATSM150_int.csv << EOF
vGPUProfileInfo ProfileID,vGPUScheduler ResetAfterVfSwitch,General
TileProvisioningMode,PFResources Lmem(B/tile),PFResources
Contexts(perTile),PFResources Doorbells(perTile),PFResources
GGTTSize(B/tile),VFResources Lmem(B/tile),VFResources
Contexts(perTile),VFResources Doorbells(perTile),VFResources
GGTTSize(B/tile),AdverseEvents GuCSamplingPeriod(msec),AdverseEvents
GuCThresholdCATError,AdverseEvents G2PFNotificationCountCATError,AdverseEvents
PFNotificationFreqCATError(msec),AdverseEvents GuCThresholdPageFault,AdverseEvents
G2PFNotificationCountPageFault,AdverseEvents
PFNotificationFreqPageFault(msec),AdverseEvents GuCThresholdH2GStorm,AdverseEvents
G2PFNotificationCountH2GStorm,AdverseEvents
PFNotificationFreqH2GStorm(msec),AdverseEvents GuCThresholdDbStorm,AdverseEvents
G2PFNotificationCountDbStorm,AdverseEvents
PFNotificationFreqDbStorm(msec),AdverseEvents GuCThresholdGTIrqStorm,AdverseEvents
G2PFNotificationCountGTIrqStorm,AdverseEvents
PFNotificationFreqGTIrqStorm(msec),AdverseEvents
GuCThresholdEngineReset,AdverseEvents
G2PFNotificationCountEngineReset,AdverseEvents PFNotificationFreqEngineReset(msec)
ATSM150_R1,F,1,1073741824,1024,16,268435456,16106127360,1024,240,4026531840,0,0,3,
10000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_V1,F,1,1073741824,1024,16,268435456,16106127360,1024,240,4026531840,0,0,3,
10000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_V2,F,3,1073741824,1024,16,268435456,8053063680,1024,120,2013265920,0,0,3,1
0000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_V4,F,3,1073741824,1024,16,268435456,4026531840,1024,60,1006632960,0,0,3,10
000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_V5,F,3,1073741824,1024,16,268435456,3221225472,1024,48,805306368,0,0,3,100
00,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_V16,F,3,1073741824,1024,16,268435456,1006632960,1024,15,251658240,0,0,3,10
000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_M1,F,1,1073741824,1024,16,268435456,16106127360,1024,240,4026531840,0,0,3,
10000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_M2,F,3,1073741824,1024,16,268435456,8053063680,1024,120,2013265920,0,0,3,1
0000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_M4,F,3,1073741824,1024,16,268435456,4026531840,1024,60,1006632960,0,0,3,10
000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_M5,F,3,1073741824,1024,16,268435456,3221225472,1024,48,805306368,0,0,3,100
00,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_M8,F,3,1073741824,1024,16,268435456,2013265920,1024,30,503316480,0,0,3,100
00,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM150_M16,F,3,1073741824,1024,16,268435456,1006632960,1024,15,251658240,0,0,3,10
000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100

EOF
```

```
$ cat >> ATSM150_vfs.csv<< EOF
vGPUProfileInfo ProfileID,vGPUProfileInfo Description,vGPUScheduler
vGPUSchedulerMode,vGPUScheduler PFExecutionQuanta(msec),vGPUScheduler
PFPreemptionTimeout(usec),vGPUScheduler VFExecutionQuanta(msec),vGPUScheduler
VFPreemptionTimeout(usec),vGPUScheduler ScheduleIfIdle
ATSM150_R1,RDSH| 1VF per pGPU | #VFs=1 | 60 fps upto [1x5K 2x4K 4xQHD 8xHD]
at H.264,TS-GPUTile,1,2000,32,64000,F
ATSM150_V1,VDI | 1VF per pGPU | #VFs=1 | 60 fps upto [1x5K 2x4K 4xQHD 8xHD]
at H.264,TS-GPUTile,1,2000,32,64000,F
ATSM150_V2,VDI | NVF per pGPU | #VFs=2 | 30 fps upto [1x5K 2x4K 4xQHD 8xHD]
at H.264,TS-GPUTile,1,2000,16,32000,T
ATSM150_V4,VDI | NVF per pGPU | #VFs=4 | 30 fps upto [1x4K 2xQHD 4xHD]
at H.264,TS-GPUTile,1,2000,8,16000,T
ATSM150_V5,VDI | NVF per pGPU | #VFs=5 | 30 fps upto [2xQHD 4xHD] at
H.264,TS-GPUTile,1,2000,6,12000,T
ATSM150_V16,VDI | NVF per pGPU | #VFs=16 | 30 fps upto [1xHD] at
H.264,TS-GPUTile,1,2000,2,4000,T
ATSM150_M1,MULTI | 1VF per pGPU | #VFs=1 | Best Effort Virtual
Display,TS-GPUTile,1,2000,64,128000,F
ATSM150_M2,MULTI | NVF per pGPU | #VFs=2 | Best Effort Virtual
Display,TS-GPUTile,1,2000,32,64000,F
ATSM150_M4,MULTI | NVF per pGPU | #VFs=4 | Best Effort Virtual
Display,TS-GPUTile,1,2000,16,32000,F
ATSM150_M5,MULTI | NVF per pGPU | #VFs=5 | Best Effort Virtual
Display,TS-GPUTile,1,2000,12,24000,F
ATSM150_M8,MULTI | NVF per pGPU | #VFs=8 | Best Effort Virtual
Display,TS-GPUTile,1,2000,8,16000,F
ATSM150_M16,MULTI | NVF per pGPU | #VFs=16 | Best Effort Virtual
Display,TS-GPUTile,1,2000,4,8000,F
EOF
```

```
$ cat >> ATSM75_int.csv<< EOF

vGPUProfileInfo ProfileID,vGPUScheduler ResetAfterVfSwitch,General
TileProvisioningMode,PFResources Lmem(B/tile),PFResources
Contexts(perTile),PFResources Doorbells(perTile),PFResources
GGTTSize(B/tile),VFResources Lmem(B/tile),VFResources
Contexts(perTile),VFResources Doorbells(perTile),VFResources
GGTTSize(B/tile),AdverseEvents GuCSamplingPeriod(msec),AdverseEvents
GuCThresholdCATError,AdverseEvents G2PFNotificationCountCATError,AdverseEvents
PFNotificationFreqCATError(msec),AdverseEvents GuCThresholdPageFault,AdverseEvents
G2PFNotificationCountPageFault,AdverseEvents
PFNotificationFreqPageFault(msec),AdverseEvents GuCThresholdH2GStorm,AdverseEvents
G2PFNotificationCountH2GStorm,AdverseEvents
PFNotificationFreqH2GStorm(msec),AdverseEvents GuCThresholdDbStorm,AdverseEvents
G2PFNotificationCountDbStorm,AdverseEvents
PFNotificationFreqDbStorm(msec),AdverseEvents GuCThresholdGTIrqStorm,AdverseEvents
G2PFNotificationCountGTIrqStorm,AdverseEvents
PFNotificationFreqGTIrqStorm(msec),AdverseEvents
GuCThresholdEngineReset,AdverseEvents
G2PFNotificationCountEngineReset,AdverseEvents PFNotificationFreqEngineReset(msec)

ATSM75_R1,F,1,1073741824,1024,16,268435456,5368709120,1024,240,4026531840,0,0,3,10
000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM75_V1,F,1,1073741824,1024,16,268435456,5368709120,1024,240,4026531840,0,0,3,10
000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM75_V3,F,3,1073741824,1024,16,268435456,1789526016,1024,80,1342177280,0,0,3,100
00,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM75_V6,F,3,1073741824,1024,16,268435456,894763008,1024,40,671088640,0,0,3,10000 ,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100


ATSM75_M1,F,1,1073741824,1024,16,268435456,5368709120,1024,240,4026531840,0,0,3,10
000,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM75_M3,F,3,1073741824,1024,16,268435456,1789526016,1024,80,1342177280,0,0,3,100
00,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
ATSM75_M6,F,3,1073741824,1024,16,268435456,894763008,1024,40,671088640,0,0,3,10000 ,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100


ATSM75_M12,F,3,1073741824,1024,16,268435456,447348736,1024,20,335544320,0,0,3,1000
0,0,3,10000,0,3,100,0,3,100,0,3,100,0,3,100
EOF
```

```
$ cat >> ATSM75_vfs.csv<< EOF
vGPUProfileInfo ProfileID,vGPUProfileInfo Description,vGPUScheduler
vGPUSchedulerMode,vGPUScheduler PFExecutionQuanta(msec),vGPUScheduler
PFPreemptionTimeout(usec),vGPUScheduler VFExecutionQuanta(msec),vGPUScheduler
VFPreemptionTimeout(usec),vGPUScheduler ScheduleIfIdle
ATSM75_R1,RDSH | 1VF per pGPU | #VFs=1 | 30fps upto [1x5K 2x4K 4xQHD 8xHD]
@ H.264,TS-GPUTile,1,2000,32,64000,F
ATSM75_V1,VDI | 1VF per pGPU | #VFs=1 | 30fps upto [1x5K 2x4K 4xQHD 8xHD]
@ H.264,TS-GPUTile,1,2000,32,64000,F
ATSM75_V3,VDI | NVF per pGPU | #VFs=3 | 30fps upto [1x4K 2xQHD 4xHD] @
H.264,TS-GPUTile,1,2000,11,22000,T
ATSM75_V6,VDI | NVF per pGPU | #VFs=6 | 30fps upto [1xQHD2xHD] @
H.264,TS-GPUTile,1,2000,5,16000,T
ATSM75_M1,MULTI | 1VF per pGPU | #VFs=1 | Best Effort Virtual
Display,TS-GPUTile,1,2000,64,128000,F
ATSM75_M3,MULTI | NVF per pGPU | #VFs=3 | Best Effort Virtual
Display,TS-GPUTile,1,2000,22,44000,F
ATSM75_M6,MULTI | NVF per pGPU | #VFs=6 | Best Effort Virtual
Display,TS-GPUTile,1,2000,16,32000,F
ATSM75_M12,MULTI | NVF per pGPU | #VFs=12 | Best Effort Virtual
Display,TS-GPUTile,1,2000,8,16000,F
EOF
```

```
$ chmod +x *.csv && cd ../..
```

- Create a file **vgpu_profiles.sh** and add the following lines of scripts.

```
#!/bin/bash

source ./provisioning.sh
source ./provision_from_config_files.sh

VERSION="2.3"
printf "${RED}vgpu_profile script Version:${GREEN}$VERSION ${NC}\n"

export MODE_1=1
export MODE_2=2
export MODE_3=3

show_progress=0
function print() {

    echo "step :" $show_progress
    for ((i = 0; i <= $show_progress; i++)); do
        echo -ne "*"
```

```
    done

    echo "*"
    echo $1
    (( show_progress += 1))
}

#Enable this for debugging
#set -x



function func_printHelp_generic() {
    printf "**************************************\n"
    printf "No arguments supplied\n"
    printf "**************************************\n"
    printf "Usage. $0 --card <card-instance> --vgpu-profile <vgpu-profile> [-
- disable-clear-provisioning] [--auto-provisioning] [--

ecc] [--fixed-scheduling] --vf-attributes <vfNum tile dbs ctxs ggtt lmem
exec_quanta preempt_quanta> [--debug <info | high]\n"


    printf "\n\n--card-instance \t\t\t - \tcard0/card1 etc. Check from /dev/dri/.
\n"
    printf "\t\t\t - \tIf not provided, will auto select the ${RED}first
gfx card${NC}\n"

    printf "\n--fixed-scheduling \t\t - \t Enable fixed scheduling mode
(Experimental pupose only. Works only with --auto-provisioning mode) (default:
no)\n"
    printf "\n--disable-clear-provisioning \t\t - \t Do not clear previous
Provisioning (default: no)\n"
    printf "\n${RED}--auto-provisioning \t - \t Enable auto provisioning
(Experimental purpose only.) (default: no. If enabled, it will be deviated for
vgpu_profile definition). ${NC}\n"
    printf "\n--ecc \t\t\t - \t Enable error correction overhead Memory into
consideration while provisioning VFs\n"
    printf "\n--debug \t\t\t - \t info level or high level\n"

    printf "\n${RED} ----------------> For Manual Provisioning <-----------------
${NC}\n"
    printf "\n--vfattr\t\t - \t\t\n"
    printf "\t\t - \t\tvfNum --> VF Number to be configured for\n"
    printf "\t\t - \t\tdbs   --> Number of doorbells to be configured for\n"
    printf "\t\t - \t\tctxs   --> Number of context to be configured for\n"
    printf "\t\t - \t\tggtt   --> Size of GGTT (in Bytes) to be configured for\n"
    printf "\t\t - \t\tlmem   --> Size of lmem (in Bytes) be configured for\n"
    printf "\t\t - \t\texec_quanta   --> Execution Quanta (in msec) to be
 configured for\n"
    printf "\t\t - \t\tpreempt_quanta   --> Preemption Qunata (in usec) to be
 configured for\n"

}
function func_printHelp_ATSM_75() {
```

```
    printf "\n${GREEN} ---------------            > For Auto Provisioning <------------------
${NC}\n"
    printf "\n--vgpu-profile \t\t - \t\tvgpu-profile (For auto provisioning)\n"
    printf "\t\t - \t\tR1   : RDSH    - 1VF\n"
    printf "\t\t - \t\tV1   : VDI     - 1VF\n"
    printf "\t\t - \t\tV3   : VDI     - 2VFs \n"
    printf "\t\t - \t\tV6   : VDI     - 6VFs \n"
    printf "\t\t - \t\tM1   : MULTI -  1VF \n"
    printf "\t\t - \t\tM3   : MULTI -  3VFs \n"
    printf "\t\t - \t\tM6   : MULTI -  6VFs \n"
    printf "\t\t - \t\tM12  : MULTI -  12VFs\n"

}

function func_printHelp_ATSM_150() {
    printf "\n${GREEN} ---------------            > For Auto Provisioning <------------------
${NC}\n"
    printf "\n--vgpu-profile \t\t - \t\tvgpu-profile (For auto provisioning)\n"
    printf "\t\t - \t\tR1   : RDSH    - 1VF\n"
    printf "\t\t - \t\tV1   : VDI     - 1VF\n"
    printf "\t\t - \t\tV2   : VDI     - 2VFs \n"
    printf "\t\t - \t\tV4   : VDI     - 4VFs \n"
    printf "\t\t - \t\tV5   : VDI     - 5VFs \n"
    printf "\t\t - \t\tV16  : VDI     - 16VFs \n"
    printf "\t\t - \t\tM1   : MULTI -  1VF \n"
    printf "\t\t - \t\tM2   : MULTI -  2VFs \n"
    printf "\t\t - \t\tM4   : MULTI -  4VFs \n"
    printf "\t\t - \t\tM5   : MULTI -  5VFs \n"
    printf "\t\t - \t\tM8   : MULTI -  8VFs\n"
    printf "\t\t - \t\tM16  : MULTI -  16VFs\n"

}




function func_printHelp_ATSP_PVC() {

    printf "\n${GREEN} ----------------> For Auto Provisioning <------------------
${NC}\n"
    printf "\n--vgpu-profile \t\t - \t\tvgpu-profile (For auto provisioning)\n"
    printf "\t\t - \t\tC1   : 1VF\n"
    printf "\t\t - \t\tC2   : 2 VFs.  1-VF per tile (if exists)\n"
    printf "\t\t - \t\tC4   : 4 VFs.  2-VFs per tile (if exists)\n"
    printf "\t\t - \t\tC8   : 8 VFs.  4-VFs per tile (if exists)\n"
    printf "\t\t - \t\tC16  : 16 VFs.  8-VFs per tile (if exists)\n"
    printf "\t\t - \t\tC32  : 32 VFs.  32-VFs per tile (if exists)\n"
    printf "\t\t - \t\tC62  : 62 VFs. 62-VFs per tile (Only for 2T
configuration)\n"
    printf "\t\t - \t\tC63  : 63 VFs.  Only for 1T configuration\n"

}
```

```
#Setup Temp directory
enableDebugLogs=0
function_setUpTempData
function_getNumOfCards

function_GetDeviceId

if [ $# -eq 0 ]
then
    func_printHelp_generic

    if [[ $global_productName = "PVC" ]] || [[ $global_productName = "ATSP" ]]
    then
        func_printHelp_ATSP_PVC

    elif [[ $global_productName = "ATSM_75" ]]
    then
        func_printHelp_ATSM_75

    elif [[ $global_productName = "ATSM_150" ]]
    then
        func_printHelp_ATSM_150

    else
        printf "${RED} No valid device found. Supported for ATS-M1
(ATSM_150), ATS-M3 (ATSM_75), ATS-P & PVC\n${NC}"
    fi

    #Cleanup Temporary data
    function_cleanUpTempData

    exit 0
fi

#Cleanup Temporary data. Need to show help menu according to platform.
Now cleaning it.
function_cleanUpTempData

InCardPath=0
InVgpuProfile=0
InMode="auto"
InTileNum=0
InVfNum=0
InDbs=0
InCtxs=0
InGgtt=0
InLmem=0
InExecQuanta=0
InPreemptQuanta=0
DonotClearProvisioning=0
```

```
InPerfMode=0
autoProvisioning=0
InMemEcc=0
#Go over CommandLine Params and identify them.
arg=1
while [[ $arg -le $# ]];
do
    caseParam=${!arg}
    #printf "arg=$arg, param=$caseParam\n"
    case $caseParam in
        -cp|--card|--cardpath)
            ActualParam=$(($arg+1))
            InCardPath=${!ActualParam}
            printf "InCardPath=$InCardPath\n"
            ;;
        -vp|--vgpu-profile|--vgpuprofile)
            ActualParam=$(($arg+1))
            InVgpuProfile=${!ActualParam}
            printf "InVgpuProfile=$InVgpuProfile\n"
            ;;
        -perf-mode|-fixed-scheduling|--perf-mode)
            printf "Enabling fixed-scheduling (sched_if_idle=0)\n"
            InPerfMode=0
            ;;
        -disable-cp|--disable-clear-provisioning)
            printf "DonotClearProvisioning=$DonotClearProvisioning\n"
            DonotClearProvisioning=1
            ;;
        -auto-provisioning|--auto-provisioning)
            printf "Enabling flexible
            provisioning\n" autoProvisioning=1
            ;;
        -ecc|--ecc)
            printf "Memory ECC enabled\n"
            InMemEcc=1
            ;;
        -debug|--debug)
            ActualParam=$(($arg+1))
            enableDebugLogs=${!ActualParam}
            ;;
        -vfattr|--vf-attributes)
            function_setUpTempDataCustom
            #Mark provisioning mode as Manual as -vfattr is provided by user.
            InMode="custom"
            #Mark autoProvisioning=1 to avoid going via scripts.
            autoProvisioning=1

            paramOffset=1
            #Store VF paramters in Temp file systsem.
            ActualParam=$(($arg+$paramOffset))
            InVfNum=${!ActualParam}
            echo $InVfNum >> $TEMP_ROOT_DIR_CUSTOM_DATA"/"$IN_VF_NUMS_PATH if
            [[ $global_productName = "PVC" ]] || [[ $global_productName =
"ATSP" ]]
            then
                (( paramOffset += 1 ))
```

```
                ActualParam=$(($arg+$paramOffset))
                InTileNum=${!ActualParam}
            else
                InTileNum=0
            fi
            echo $InTileNum > $TEMP_ROOT_DIR_CUSTOM_DATA"/"$InVfNum$IN_TILE_PATH

            (( paramOffset += 1 ))
            ActualParam=$(($arg+$paramOffset))
            InDbs=${!ActualParam}
            echo $InDbs > $TEMP_ROOT_DIR_CUSTOM_DATA"/"$InVfNum$IN_DBS_PATH

            (( paramOffset += 1 ))
            ActualParam=$(($arg+$paramOffset))
            InCtxs=${!ActualParam}
            echo $InCtxs> $TEMP_ROOT_DIR_CUSTOM_DATA"/"$InVfNum$IN_CTXS_PATH

            (( paramOffset += 1 ))
            ActualParam=$(($arg+$paramOffset))
            InGgtt=${!ActualParam}
            echo $InGgtt > $TEMP_ROOT_DIR_CUSTOM_DATA"/"$InVfNum$IN_GGTT_PATH

            (( paramOffset += 1 ))
            ActualParam=$(($arg+$paramOffset))
            InLmem=${!ActualParam}
            echo $InLmem > $TEMP_ROOT_DIR_CUSTOM_DATA"/"$InVfNum$IN_LMEM_PATH

            (( paramOffset += 1 ))
            ActualParam=$(($arg+$paramOffset))
            InExecQuanta=${!ActualParam}
            echo $InExecQuanta >
$TEMP_ROOT_DIR_CUSTOM_DATA"/"$InVfNum$IN_EXEC_QUOTA_PATH

            (( paramOffset += 1 ))
            ActualParam=$(($arg+$paramOffset))
            InPreemptQuanta=${!ActualParam}
            echo $InPreemptQuanta >
$TEMP_ROOT_DIR_CUSTOM_DATA"/"$InVfNum$IN_PREEMPT_QUOTA_PATH

            function_debugPrints "InVfNum=$InVfNum, tile=$InTileNum, InDbs=$InDbs,
InCtxs=$InCtxs, InGgtt=$InGgtt \n"
            function_debugPrints "InLmem=$InLmem, InExecQuanta=$InExecQuanta,
InPreemptQuanta=$InPreemptQuanta\n"
    esac

    (( arg+=1 ))
done

#Setup Temp directory
function_setUpTempData
function_getNumOfCards

if [ "$InCardPath" = "0" ]
 then
        #Select the first valid gfx card.
        InCardPath=$(sed '1!d' $TEMP_ROOT_DIR$GFX_CARDS_PATH)
        printf "Selecting card automatically: ${RED} $InCardPath ${NC}\n"
```

```
 fi



#Clear previous provisioning
if [ $DonotClearProvisioning -eq 0 ]
then
    function_clearProvisioning $InCardPath
    #Setup Temp directory as we have cleared in above function
    function_setUpTempData
    #Wait for some time so that lmem from all VFs is released.
    sleep 2
fi




if [ $autoProvisioning -eq 0 ]
then
    printf "Applying provisioning from vgpu_profile data sheet inputs\n"
    printf "CardPath = $InCardPath\n"
    #echo $CONFIG_FILE_PATH

    #Update ECC info
    global_mem_ecc_enable=$InMemEcc

    function_getNumOfCards
    function_getPfResources

    case $global_productName in
                "ATSP"|"PVC")
            #Get No. of tiles.
            #global_no_of_tiles=$(cat $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum)
            echo $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum
            echo "global_no_of_tiles=$global_no_of_tiles"

            vfs_filename=0
            int_filename=0

            if [ $global_no_of_tiles -eq 1 ]
            then
                vfs_filename=$CONFIG_FILE_PATH$PVC_CONFIG_1T_FILE_VFS_NAME
                int_filename=$CONFIG_FILE_PATH$PVC_CONFIG_1T_FILE_INT_NAME
            elif [ $global_no_of_tiles -eq 2 ]
            then
                vfs_filename=$CONFIG_FILE_PATH$PVC_CONFIG_2T_FILE_VFS_NAME
                int_filename=$CONFIG_FILE_PATH$PVC_CONFIG_2T_FILE_INT_NAME
            else
                echo "The Tiles=$global_no_of_tiles is not supported yet"
                exit -1
            fi
```

```
            printf
"#############################################################\n"
            printf "Applying Config file name = $vfs_filename\n"
            printf
"#############################################################\n"

            func_config_vfs_fileData $InCardPath $global_productName $vfs_filename
$InVgpuProfile
            printf
"#############################################################\n"
            printf "Applying Config file name = $int_filename\n"
            printf
"#############################################################\n"
            func_config_int_fileData $InCardPath $global_productName $int_filename
$InVgpuProfile
            ;;

        "ATSM_75")
            printf
"#############################################################\n"
            printf "Applying Config file name =
$CONFIG_FILE_PATH$ATSM75_CONFIG_FILE_VFS_NAME\n"
            printf
"#############################################################\n"
            func_config_vfs_fileData $InCardPath $global_productName
$CONFIG_FILE_PATH$ATSM75_CONFIG_FILE_VFS_NAME $InVgpuProfile

            printf
"#############################################################\n"
            printf "Applying Config file name =
$CONFIG_FILE_PATH$ATSM75_CONFIG_FILE_VFS_NAME\n"
            printf
"#############################################################\n"
            func_config_int_fileData $InCardPath $global_productName
$CONFIG_FILE_PATH$ATSM75_CONFIG_FILE_INT_NAME $InVgpuProfile
            ;;


        "ATSM_150")
            printf
"#############################################################\n"
            printf "Applying Config file name =
$CONFIG_FILE_PATH$ATSM150_CONFIG_FILE_VFS_NAME\n"
            printf
"#############################################################\n"
            func_config_vfs_fileData $InCardPath $global_productName
$CONFIG_FILE_PATH$ATSM150_CONFIG_FILE_VFS_NAME $InVgpuProfile

            printf
"#############################################################\n"
            printf "Applying Config file name =
$CONFIG_FILE_PATH$ATSM150_CONFIG_FILE_INT_NAME\n"
            printf
"#############################################################\n"
            func_config_int_fileData $InCardPath $global_productName
$CONFIG_FILE_PATH$ATSM150_CONFIG_FILE_INT_NAME $InVgpuProfile
            ;;
```

```
        *)
            printf "This script does not support this product yet\n"
            exit -1
        esac

else

#If provisioning mode is auto, perform auto provisioning.
printf "InMode=$InMode\n"
if [ $InMode = "auto" ]
then
    printf "InVgpuProfile=$InVgpuProfile\n"

    case $global_productName in
        "ATSP"|"PVC")
            global_pfExecQuantInMS=1
            global_pfPreemptQuanInUS=2000
            global_cat_error_count=0
            global_doorbell_time_us=0
            global_engine_reset_count=0
            global_h2g_time_us=0
            global_irq_time_us=0
            global_page_fault_count=0
            global_pf_policies_sched_if_idle=$InPerfMode
            global_pf_policies_sample_period_ms=2
            global_pf_policies_engine_reset=0
            case $InVgpuProfile in
                "C1")
                    printf "**************************************\n"
                    printf "Provisioning in mode-1\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_1
                    NoOfVfs=1
                    global_vfExecQuantInMS=128
                    global_vfPreemptQuanInUS=0

                    printf "provisioningMode = $provisioningMode\n"
                    printf "NoOfVfs = $NoOfVfs\n"

                    ;;

                "C2")
                    printf "**************************************\n"
                    printf "Provisioning in mode-2\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_2
                    global_vfExecQuantInMS=64
                    global_vfPreemptQuanInUS=128000
                    NoOfVfs=2

                    printf "provisioningMode = $provisioningMode\n"
                    printf "NoOfVfs = $NoOfVfs\n"



                    ;;
```

```
                "C4")
                    printf "**************************************\n"
                    printf "Provisioning in mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=32
                    global_vfPreemptQuanInUS=64000

                    NoOfVfs=4
                    ;;
                "C8")
                    printf "**************************************\n"
                    printf "Provisioning in mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=16
                    global_vfPreemptQuanInUS=32000

                    NoOfVfs=8
                    ;;
                "C16")
                    printf "**************************************\n"
                    printf "Provisioning in mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=8
                    global_vfPreemptQuanInUS=16000

                    NoOfVfs=16
                    ;;
                "C32")
                    printf "**************************************\n"
                    printf "Provisioning in mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=4
                    global_vfPreemptQuanInUS=8000

                    NoOfVfs=32
                    ;;
                "C62")
                    printf "**************************************\n"
                    printf "Provisioning in mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=2
                    global_vfPreemptQuanInUS=4000

                    NoOfVfs=62
                    ;;
                "C63")
                    printf "**************************************\n"
                    printf "Provisioning in mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=2
                    global_vfPreemptQuanInUS=4000
```

```
                NoOfVfs=63
                ;;



            *)
                printf "Error in received vgpu profile\n"
                exit 0
        esac
        ;;

    "ATSM_150")
        global_pfExecQuantInMS=1
        global_pfPreemptQuanInUS=2000
        global_cat_error_count=0
        global_doorbell_time_us=0
        global_engine_reset_count=0
        global_h2g_time_us=0
        global_irq_time_us=0
        global_page_fault_count=0
        global_pf_policies_sched_if_idle=$InPerfMode
        global_pf_policies_sample_period_ms=2
        global_pf_policies_engine_reset=0
        case $InVgpuProfile in
            "R1")
                printf "***************************************\n"
                printf "Provisioning in RDSH mode-1\n"
                printf "***************************************\n"
                provisioningMode=$MODE_1
                NoOfVfs=1
                global_vfExecQuantInMS=32
                global_vfPreemptQuanInUS=64000

                ;;
            "V1")
                printf "***************************************\n"
                printf "Provisioning in VDI mode-1\n"
                printf "***************************************\n"
                provisioningMode=$MODE_1
                NoOfVfs=1
                global_vfExecQuantInMS=32
                global_vfPreemptQuanInUS=64000

                ;;

            "V2")
                printf "***************************************\n"
                printf "Provisioning in VDI mode-3\n"
                printf "***************************************\n"
                provisioningMode=$MODE_3
                global_vfExecQuantInMS=16
                global_vfPreemptQuanInUS=32000
                NoOfVfs=2

                ;;
```

```
"V4")
    printf "*************************************\n"
    printf "Provisioning in VDI mode-3\n"
    printf "*************************************\n"
    provisioningMode=$MODE_3
    global_vfExecQuantInMS=8
    global_vfPreemptQuanInUS=16000

    NoOfVfs=4
    ;;

"V5")
    printf "*************************************\n"
    printf "Provisioning in mode-3\n"
    printf "*************************************\n"
    provisioningMode=$MODE_3
    global_vfExecQuantInMS=6
    global_vfPreemptQuanInUS=13000

    NoOfVfs=5
    ;;
"V8")
    printf "*************************************\n"
    printf "Provisioning in mode-3\n"
    printf "*************************************\n"
    provisioningMode=$MODE_3
    global_vfExecQuantInMS=4
    global_vfPreemptQuanInUS=8000

    NoOfVfs=8
    ;;
"V16")
    printf "*************************************\n"
    printf "Provisioning in mode-3\n"
    printf "*************************************\n"
    provisioningMode=$MODE_3
    global_vfExecQuantInMS=2
    global_vfPreemptQuanInUS=4000

    NoOfVfs=16
    ;;
"M1")
    printf "*************************************\n"
    printf "Provisioning in MULTI mode-1\n"
    printf "*************************************\n"
    provisioningMode=$MODE_1
    NoOfVfs=1
    global_vfExecQuantInMS=64
    global_vfPreemptQuanInUS=128000

    ;;

"M2")
    printf "*************************************\n"
    printf "Provisioning in MULTI mode-3\n"
    printf "*************************************\n"
```

```
        provisioningMode=$MODE_3
        global_vfExecQuantInMS=32
        global_vfPreemptQuanInUS=64000
        NoOfVfs=2

        ;;

    "M4")
        printf "****************************************\n"
        printf "Provisioning in MULTI mode-3\n"
        printf "****************************************\n"
        provisioningMode=$MODE_3
        global_vfExecQuantInMS=16
        global_vfPreemptQuanInUS=32000

        NoOfVfs=4
        ;;

    "M5")
        printf "****************************************\n"
        printf "Provisioning in MULTI mode-3\n"
        printf "****************************************\n"
        provisioningMode=$MODE_3
        global_vfExecQuantInMS=13
        global_vfPreemptQuanInUS=26000

        NoOfVfs=5
        ;;
    "M8")
        printf "****************************************\n"
        printf "Provisioning in mode-3\n"
        printf "****************************************\n"
        provisioningMode=$MODE_3
        global_vfExecQuantInMS=8
        global_vfPreemptQuanInUS=16000

        NoOfVfs=8
        ;;
    "M16")
        printf "****************************************\n"
        printf "Provisioning in mode-3\n"
        printf "****************************************\n"
        provisioningMode=$MODE_3
        global_vfExecQuantInMS=4
        global_vfPreemptQuanInUS=8000

        NoOfVfs=16
        ;;
    "M31")
        printf "****************************************\n"
        printf "Provisioning in mode-3\n"
        printf "****************************************\n"
        provisioningMode=$MODE_3
        global_vfExecQuantInMS=2
        global_vfPreemptQuanInUS=4000

        NoOfVfs=31
```

```
                    ;;

                *)
                    printf "Error in received vgpu profile\n"
                    exit -1
            esac
            ;;
        "ATSM_75")
            global_pfExecQuantInMS=1
            global_pfPreemptQuanInUS=2000
            global_cat_error_count=0
            global_doorbell_time_us=0
            global_engine_reset_count=0
            global_h2g_time_us=0
            global_irq_time_us=0
            global_page_fault_count=0
            global_pf_policies_sched_if_idle=$InPerfMode
            global_pf_policies_sample_period_ms=2
            global_pf_policies_engine_reset=0 case
            $InVgpuProfile in
                "R1")
                    printf "***************************************\n"
                    printf "Provisioning in RDSH mode-1\n"
                    printf "***************************************\n"
                    provisioningMode=$MODE_1
                    NoOfVfs=1
                    global_vfExecQuantInMS=32
                    global_vfPreemptQuanInUS=64000

                    ;;
                "V1")
                    printf "***************************************\n"
                    printf "Provisioning in VDI mode-1\n"
                    printf "***************************************\n"
                    provisioningMode=$MODE_1
                    NoOfVfs=1
                    global_vfExecQuantInMS=32
                    global_vfPreemptQuanInUS=64000

                    ;;

                "V3")
                    printf "***************************************\n"
                    printf "Provisioning in VDI mode-3\n"
                    printf "***************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=11
                    global_vfPreemptQuanInUS=22000
                    NoOfVfs=3

                    ;;

                "V6")
                    printf "***************************************\n"
                    printf "Provisioning in VDI mode-3\n"
                    printf "***************************************\n"
                    provisioningMode=$MODE_3
```

```
                    global_vfExecQuantInMS=5
                    global_vfPreemptQuanInUS=16000

                    NoOfVfs=6
                    ;;
                "M1")
                    printf "**************************************\n"
                    printf "Provisioning in MULTI mode-1\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_1
                    NoOfVfs=1
                    global_vfExecQuantInMS=64
                    global_vfPreemptQuanInUS=128000

                    ;;

                "M3")
                    printf "**************************************\n"
                    printf "Provisioning in MULTI mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=22
                    global_vfPreemptQuanInUS=44000
                    NoOfVfs=3

                    ;;

                "M6")
                    printf "**************************************\n"
                    printf "Provisioning in MULTI mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=16
                    global_vfPreemptQuanInUS=32000

                    NoOfVfs=6
                    ;;
                "M12")
                    printf "**************************************\n"
                    printf "Provisioning in mode-3\n"
                    printf "**************************************\n"
                    provisioningMode=$MODE_3
                    global_vfExecQuantInMS=8
                    global_vfPreemptQuanInUS=16000

                    NoOfVfs=12
                    ;;
                *)
                    printf "Error in received vgpu profile\n"
                    exit -1
            esac
            ;;

        *)
            printf "This script does not support this product yet\n"
            exit -1
    esac
```

```
    printf "provisioningMode = $provisioningMode\n"
    printf "NoOfVfs = $NoOfVfs\n"

        #Provision the VFs
        function_provisionVfsAuto $InCardPath $provisioningMode $NoOfVfs
        #Custom mode
    else
        printf "Provisioning VF manually\n"
        global_pfExecQuantInMS=1
        global_pfPreemptQuanInUS=2000
        global_cat_error_count=0
        global_doorbell_time_us=0
        global_engine_reset_count=0
        global_h2g_time_us=0
        global_irq_time_us=0
        global_page_fault_count=0
        global_pf_policies_sched_if_idle=$InPerfMode
        global_pf_policies_sample_period_ms=2
        global_pf_policies_engine_reset=0

        function_provisionVfsManual $InCardPath
        #Has input data. So, clear it.
        function_cleanUpTempDataCustom
        #provisionVfManually

fi

fi
#Cleanup Temporary data
function_cleanUpTempData
```

- Create a file **provisioning.sh** and add the following lines of scripts:

```
#!/bin/bash

declare -a drmContents
declare -a GfxCards
declare -a TilesInGfxCard
declare -a pfResources

declare global_devId
declare global_productName
declare global_maxVfs

vfLmemRoundingFactor=2097152 #2MB
vfGgttRoundingFactor=65536 #64kB
vfCtxsRoundingFactor=256

TEMP_ROOT_DIR="/tmp/sriov_provision"
TEMP_ROOT_DIR_CUSTOM_DATA="/tmp/sriov_provision_custom_data"
```

```
GFX_CARDS_PATH="/gfxCards"
NUM_OF_GFX_CARDS_DETECTED="/numOfGfxCardsDetected"
TILES_GFX_CARD="/Tiles_in_card_"
DB_MAX_QUOTA="/doorbells_max_quota"
CTX_MAX_QUOTA="/contexts_max_quota"
GGTT_MAX_QUOTA="/ggtt_max_quota"
LMEM_MAX_QUOTA="/lmem_max_quota"
IOV_PF_PATH="/iov/pf"
IOV_VF_PATH="/iov/vf"

IN_VF_NUMS_PATH="in_vfs"
IN_TILE_PATH="_in_tile"
IN_DBS_PATH="_in_dbs"
IN_CTXS_PATH="_in_ctxs"
IN_GGTT_PATH="_in_ggtt"
IN_LMEM_PATH="_in_lmem"
IN_EXEC_QUOTA_PATH="in_exec_quota"
IN_PREEMPT_QUOTA_PATH="in_preempt_quota"


DbsReservedForPF=16
GgttReservedForPF=2097152 #2MB
CtxsReservedForPF=1024
LmemReservedForPF=536870912 #512MB

MaxTiles=4
NumOfGfxCardsDetected=0

#Thresholds for different platforms
global_cat_error_count=0
global_doorbell_time_us=0
global_engine_reset_count=0
global_h2g_time_us=0
global_irq_time_us=0
global_page_fault_count=0

global_pf_policies_sample_period_ms=0
global_pf_policies_engine_reset=0
global_pf_policies_sched_if_idle=0

#Execution Quanta for different platforms
global_vfExecQuantInMS=0
global_pfExecQuantInMS=0




#Pre-emption Quanta for different platforms
global_vfPreemptQuanInUS=0
global_pfPreemptQuanInUS=0

#Constants for printing Color
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color
```

```
function function_printFuncName() {
    if [ $enableDebugLogs = "info" ]
    then
        printf "${GREEN}################################## \n"

        if [ $2 -eq 0 ]
        then
            printf "# \tEntered $1\t # \n"
        else
            printf "# \tExited $1\t #\n"
        fi

        printf "################################## ${NC}\n"
    fi
}

function function_debugPrints() {
    if [ $enableDebugLogs = "high" ] || [ $enableDebugLogs = "info"
    ] then
        str="$@"
        printf "${str}"
    fi
}

function function_setUpTempData() {
    #Create a directory in temp to store user data
    function_printFuncName ${FUNCNAME[0]} 0
    function_debugPrints "${YELLOW}Creating Directory for Temporary Data ${NC} \n"
    mkdir -p $TEMP_ROOT_DIR
    function_printFuncName ${FUNCNAME[0]} 1
}

function function_setUpTempDataCustom() {
    #Create a directory in temp to store user data
    function_printFuncName ${FUNCNAME[0]} 0
    function_debugPrints "${YELLOW}Creating Directory for Temporary Data
Custom $TEMP_ROOT_DIR_CUSTOM_DATA ${NC} \n"
    mkdir -p $TEMP_ROOT_DIR_CUSTOM_DATA
    function_printFuncName ${FUNCNAME[0]} 1
}

function function_cleanUpTempData() {
    function_printFuncName ${FUNCNAME[0]} 0
    function_debugPrints "${RED}Cleaning up Temporary Data ${NC} \n"
    rm -rf $TEMP_ROOT_DIR
    function_printFuncName ${FUNCNAME[0]} 1
}

function function_cleanUpTempDataCustom() {
    function_printFuncName ${FUNCNAME[0]} 0
    function_debugPrints "${RED}Cleaning up Temporary Data Custom ${NC} \n"
    rm -rf $TEMP_ROOT_DIR_CUSTOM_DATA
    function_printFuncName ${FUNCNAME[0]} 1
}
```

```
function function_GetDeviceId() {
    global_devId=0
    function_debugPrints $GfxCards
    PCI_ID=$(grep -i PCI_ID $GfxCards"/device/uevent")
    #printf "$PCI_ID\n"
    PREFIX="PCI_ID=8086:"
    global_devId=${PCI_ID#"$PREFIX"}
    printf "devId=0x$global_devId \n"

    if [[ 0x$global_devId -ge 0x200 ]] && [[ 0x$global_devId -le 0x210 ]]
    then
        global_productName="ATSP"
        global_maxVfs=63

    elif [[ 0x$global_devId -ge 0xbd0 ]] && [[ 0x$global_devId -le 0xbe5 ]]
    then
        global_productName="PVC"
        global_maxVfs=63

    #elif [[ 0x$global_devId -ge 0x56A0 ]] && [[ 0x$global_devId -le 0x56AF ]]
    elif [[ 0x$global_devId -eq 0x56C0 ]] then

        global_productName="ATSM_150"
        global_maxVfs=31

    elif [[ 0x$global_devId -eq 0x56C1 ]]
    #elif [[ 0x$global_devId -ge 0x56C0 ]] && [[ 0x$global_devId -le 0x56CF ]]
    then
        global_productName="ATSM_75"
        global_maxVfs=31
    fi

    echo "Identified $global_productName device"
}

function function_getNumOfCards() {

    function_printFuncName ${FUNCNAME[0]} 0

    SysFsDirPath="/sys/class/drm/"

    arrayIndex=0

    function_debugPrints "SysFsDirPath=$SysFsDirPath\n"
    #Get the directory list from /sys/class/drm/
    for entry in "$SysFsDirPath"*
    do
            #echo $entry
        #Store it in a array
        drmContents[$arrayIndex]=$entry
        (( arrayIndex += 1 ))
    done

    #Print the directory names from /sys/class/drm/
    function_debugPrints "****************************************\n"
```

```
    function_debugPrints "Contents of /sys/class/drm/ \n"
    function_debugPrints "****************************************\n"
    for i in ${!drmContents[@]};
    do
        function_debugPrints "${drmContents[$i]}\n"
        (( i += 1 ))
    done



    #Get Gfx card path from sysfs.
    GfxCardsIndex=0
    drmContentsIndex=0
    for drmContentsIndex in ${!drmContents[@]};
    do
        path="${drmContents[$drmContentsIndex]}/gt"
        if [ -d $path ]
        then
            GfxCards[$GfxCardsIndex]=${drmContents[$drmContentsIndex]}
            #Store card path in tmp directory and append to the same file
            echo ${GfxCards[$GfxCardsIndex]} >>
            $TEMP_ROOT_DIR$GFX_CARDS_PATH (( GfxCardsIndex += 1 ))
        fi
        (( drmContentsIndex += 1 ))
    done

    #Num of Gfx cards detected.
    NumOfGfxCardsDetected=$GfxCardsIndex
    printf "No. of cards detected are: $NumOfGfxCardsDetected\n"
    echo $NumOfGfxCardsDetected > $TEMP_ROOT_DIR$NUM_OF_GFX_CARDS_DETECTED


    #Print Num of cards and tiles
    function_debugPrints "****************************************\n"
    function_debugPrints "Tiles in a card \n"
    function_debugPrints "****************************************\n"

    #Get the number of tiles in a card
    for printCardNum in ${!GfxCards[@]};
    do
        NoOfTilePresent=0
        function_debugPrints "${GfxCards[$printCardNum]}\n"
        for TileNum in `seq 0 $MaxTiles`;
        do
            TilePath="${GfxCards[$printCardNum]}/gt/gt$TileNum"
            if [ -d $TilePath ]
            then
                function_debugPrints "TilePath=$TilePath\n"
                (( NoOfTilePresent += 1))
            fi
            (( TileNum += 1 ))
        done

        TilesInGfxCard[$printCardNum]=$NoOfTilePresent

        function_debugPrints " Tiles in ${GfxCards[$printCardNum]}
are ${TilesInGfxCard[$printCardNum]}\n"
```

```
        #Store tiles of each card in corresponding file name
        echo ${TilesInGfxCard[$printCardNum]} >
$TEMP_ROOT_DIR$TILES_GFX_CARD"$printCardNum"
    done


    function_printFuncName ${FUNCNAME[0]} 1
}

function function_getPfResources() {

    function_printFuncName ${FUNCNAME[0]} 0
    cardNum=0
    #Go over the gfxCards file and findout number of cards.
    while read -r line;
    do
        #Get the card path
        cardPath=$line
        #Find number of tiles in this card
        numofTilesInCard=$(cat $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum)



        #Get PF resources from each tile in a card
        (( numofTilesInCard -= 1 ))
        for tileNum in `seq 0 $numofTilesInCard`;
        do
            dbs=0
            ctxs=0
            ggtt=0
            lmem=0

            if [[ $global_productName = "ATSM_75" ]] || [[ $global_productName
= "ATSM_150" ]]
            then
                dbs=$(cat
$cardPath$IOV_PF_PATH"/gt/available/doorbells_max_quota")
                #Leave 32 doorbells for PF
                dbs=$(($dbs-$DbsReservedForPF))
                echo $dbs >
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum

                ctxs=$(cat
$cardPath$IOV_PF_PATH"/gt/available/contexts_max_quota")
                #Leave Some contexts for PF
                ctxs=$(($ctxs-$CtxsReservedForPF))
                echo $ctxs >
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum

                ggtt=$(cat $cardPath$IOV_PF_PATH"/gt/available/ggtt_max_quota")
                #Leave Some ggtt space for PF
                ggtt=$(($ggtt-$GgttReservedForPF))
                echo $ggtt >
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum

                lmem=$(cat $cardPath$IOV_PF_PATH"/gt/available/lmem_max_quota")
                #Leave Some lmem for PF
```

```
                lmem=$(($lmem-$LmemReservedForPF))
                echo $lmem >
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNum

            else
                dbs=$(cat
$cardPath$IOV_PF_PATH"/gt$tileNum/available/doorbells_max_quota")
                #Leave 32 doorbells for PF
                dbs=$(($dbs-$DbsReservedForPF))
                echo $dbs >
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum

                ctxs=$(cat
$cardPath$IOV_PF_PATH"/gt$tileNum/available/contexts_max_quota")
                #Leave Some contexts for PF
                ctxs=$(($ctxs-$CtxsReservedForPF))
                echo $ctxs >
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum

                ggtt=$(cat
$cardPath$IOV_PF_PATH"/gt$tileNum/available/ggtt_max_quota")
                #Leave Some ggtt space for PF
                ggtt=$(($ggtt-$GgttReservedForPF))
                echo $ggtt >
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum

                lmem=$(cat
$cardPath$IOV_PF_PATH"/gt$tileNum/available/lmem_max_quota")
                #Leave Some lmem for PF
                lmem=$(($lmem-$LmemReservedForPF))
                echo $lmem >
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNu
            m fi

            function_debugPrints
"************************************************************\n"
            function_debugPrints "Available Resources for all VFs tile ${RED}
gt$tileNum${NC} of card ${RED} $cardPath:${NC}\n"
            function_debugPrints " dbs = ${YELLOW} $dbs ${NC} \n ctxs = ${YELLOW}
$ctxs ${NC} \n ggtt = ${YELLOW} $ggtt ${NC} \n lmem = ${YELLOW} $lmem ${NC} \n"
            function_debugPrints
"************************************************************\n"
        done
        (( cardNum += 1 ))

    done < "$TEMP_ROOT_DIR$GFX_CARDS_PATH"
    function_printFuncName ${FUNCNAME[0]} 1
}



function function_clearProvisioning() {
    function_printFuncName ${FUNCNAME[0]} 0
    cardToConfigure=$1

    #Get relative card number.
    cardNum=0
```

```
    foundCard=0
    vfSupportedByHw=0

    #Get the number of cards connected.
    #function_getNumOfCards
    #Go over the gfxCards file and findout number of cards.
    while read -r line;
    do
        #Get the card path
        cardPath=$line
        if [[ $cardPath == *$cardToConfigure ]]
        then
            (( foundCard += 1 ))
            function_debugPrints "Found the card and path is $cardPath\n"

            #Get how many VFs are supported by this card.
            vfSupportedByHw=$(ls -d $cardPath$IOV_VF_PATH* | wc -l)
            function_debugPrints "${GREEN}vfSupportedByHw=$vfSupportedByHw${NC}
\n"

            function_debugPrints "Clearing provision of VFs\n"
            echo 0 > $cardPath$IOV_PF_PATH"/device/sriov_numvfs"
            #Find number of tiles in this card
            numofTilesInCard=$(cat $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum)
            global_no_of_tiles=$numofTilesInCard
            (( numofTilesInCard -= 1 ))

            for vfNum in `seq 1 $vfSupportedByHw`;
            do
                #echo "$vfSupportedByHw"
                #echo "$numofTilesInCard"

                #Go over all the tiles and provision VFs
                for tileNum in `seq 0 $numofTilesInCard`;
                do
                    dbs=0
                    ctxs=0
                    ggtt=0
                    lmem=0
                    exec_quanta=0
                    preempt_quanta=0
                    cat_error_count=0
                    doorbell_time_us=0
                    engine_reset_count=0
                    h2g_time_us=0
                    irq_time_us=0
                    page_fault_count=0
                    sched_if_idle=0
                    sample_period_ms=0
                    engine_reset=0

                if [[ $global_productName = "ATSM_75" ]] || [[ $global_productName
= "ATSM_150" ]]
                then
                    echo $dbs > $cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota"
                    echo $ctxs > $cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota"
                    echo $ggtt > $cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota"
```

```
                    echo $lmem > $cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota"
                    echo $exec_quanta >
$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
                    echo $preempt_quanta >
$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us"
                    echo $cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count"



                    echo $doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us"
                    echo $engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count"
                    echo $h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us"
                    echo $irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us"
                    echo $page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count"
                    echo $sched_if_idle >
$cardPath$IOV_PF_PATH"/gt/policies/sched_if_idle"
                    echo $sample_period_ms >
$cardPath$IOV_PF_PATH"/gt/policies/sample_period_ms"
                    echo $engine_reset >
$cardPath$IOV_PF_PATH"/gt/policies/engine_reset"


                else
                    #echo "Entered here vf=$vfNum, tile=$tileNum"
                    echo $dbs >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota"
                    echo $ctxs >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota"
                    echo $ggtt >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota"
                     echo $lmem >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota"
                    echo $exec_quanta >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms"
                    echo $preempt_quanta >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us"
                    echo $cat_error_count >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count"
                     echo $doorbell_time_us >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us"
                    echo $engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count"
                    echo $h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us"
                    echo $irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us"
                    echo $page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count"
                    echo $sched_if_idle >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle"
```

```
                    echo $sample_period_ms >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms"
                    echo $engine_reset >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset"

                fi
                done
            done
        fi
        (( cardNum += 1 ))
    done < "$TEMP_ROOT_DIR$GFX_CARDS_PATH"

    #Clean up data.
    function_cleanUpTempData

    function_printFuncName ${FUNCNAME[0]} 1
}

function provisionMode1() {
    function_printFuncName ${FUNCNAME[0]} 0
    cardToConfigure=$1
    vfNum=1
    printf "Configuring mode-1 for card:$cardToConfigure \n"

    #Get relative card number.
    cardNum=0
    foundCard=0
        #Go over the gfxCards file and findout number of cards.
        while read -r line;
        do
                #Get the card path
                cardPath=$line
        if [[ $cardPath == *$cardToConfigure ]]
        then
            (( foundCard += 1 ))
            function_debugPrints "Found the card and path is $cardPath\n"

            #Find number of tiles in this card
            numofTilesInCard=$(cat $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum)
            (( numofTilesInCard -= 1 ))
            #Go over all the tiles and provision VFs
            for tileNum in `seq 0 $numofTilesInCard`;



            do
                dbs=0
                ctxs=0
                ggtt=0
                lmem=0

                printf "*************************************************\n"
                printf "* \t ${GREEN} Resources alloted to VF$vfNum\t ${NC} * \n"
                printf "*************************************************\n"

                if [[ $global_productName = "ATSM_75" ]] || [[ $global_productName
= "ATSM_150" ]]
```

```
                then
                    dbs=$(cat
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $dbs > $cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota" %10s
= ${GREEN} $dbs ${NC} \n"

                    ctxs=$(cat
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $ctxs > $cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota" %11s =
${GREEN} $ctxs ${NC} \n"

                    ggtt=$(cat
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $ggtt > $cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota" %15s =
${GREEN} $ggtt ${NC} \n"

                    lmem=$(cat
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $lmem > $cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota" %15s =
${GREEN} $lmem ${NC} \n"

                    echo $global_vfExecQuantInMS >
$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
%10s = ${GREEN} $global_vfExecQuantInMS ${NC} \n"

                    echo $global_vfPreemptQuanInUS >
$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us"
%7s = ${GREEN} $global_vfPreemptQuanInUS ${NC} \n"

                    echo $global_pfExecQuantInMS
> $cardPath$IOV_PF_PATH"/gt/exec_quantum_ms"
                    printf "$cardPath$IOV_PF_PATH"/gt/exec_quantum_ms" %14s =
${GREEN} $global_pfExecQuantInMS ${NC} \n"

                    echo $global_pfPreemptQuanInUS >
$cardPath$IOV_PF_PATH"/gt/preempt_timeout_us"
                    printf "$cardPath$IOV_PF_PATH"/gt/preempt_timeout_us" %14s
= ${GREEN} $global_pfPreemptQuanInUS ${NC} \n"

                    echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count" %14s = ${GREEN}
$global_cat_error_count ${NC} \n"

                    echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us" %14s = ${GREEN}
$global_doorbell_time_us ${NC} \n"
```

```
                    echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count" %14s = ${GREEN}
$global_engine_reset_count ${NC} \n"

                    echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us"
%14s = ${GREEN} $global_h2g_time_us ${NC} \n"

                    echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us"
%14s = ${GREEN} $global_irq_time_us ${NC} \n"

                    echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count" %14s = ${GREEN}
$global_page_fault_count ${NC} \n"


                    echo $global_pf_policies_sched_if_idle
> $cardPath$IOV_PF_PATH"/gt/policies/sched_if_idle"
                    printf "$cardPath$IOV_PF_PATH"/gt/policies/sched_if_idle" %14s
= ${GREEN} $global_pf_policies_sched_if_idle ${NC} \n"

                    echo $global_pf_policies_sample_period_ms
> $cardPath$IOV_PF_PATH"/gt/policies/sample_period_ms"
                    printf "$cardPath$IOV_PF_PATH"/gt/policies/sample_period_ms"
%14s = ${GREEN} $global_pf_policies_sample_period_ms ${NC} \n"

                    echo $global_pf_policies_engine_reset >
$cardPath$IOV_PF_PATH"/gt/policies/engine_reset"
                    printf "$cardPath$IOV_PF_PATH"/gt/policies/engine_reset" %14s
= ${GREEN} $global_pf_policies_engine_reset ${NC} \n"
                else
                    dbs=$(cat
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $dbs >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota" %10s = ${GREEN} $dbs
${NC} \n"

                    ctxs=$(cat
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $ctxs >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota" %11s = ${GREEN} $ctxs
${NC} \n"

                    ggtt=$(cat
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
```

```
                    echo $ggtt >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota"
%15s = ${GREEN} $ggtt ${NC} \n"

                    lmem=$(cat
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $lmem >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota"
%15s = ${GREEN} $lmem ${NC} \n"

                    echo $global_vfExecQuantInMS >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms" %10s =
${GREEN} $global_vfExecQuantInMS ${NC} \n"

                    echo $global_vfPreemptQuanInUS >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us" %7s = ${GREEN}
$global_vfPreemptQuanInUS ${NC} \n"

                    echo $global_pfExecQuantInMS >
$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
                    printf "$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms" %7s
= ${GREEN} $global_pfExecQuantInMS ${NC} \n"

                    echo $global_pfPreemptQuanInUS >
$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
                    printf "$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
%7s = ${GREEN} $global_pfPreemptQuanInUS ${NC} \n"

                    echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count" %7s = ${GREEN}
$global_cat_error_count ${NC} \n"

                    echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us" %7s =
${GREEN} $global_doorbell_time_us ${NC} \n"

                    echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count" %7s =
${GREEN} $global_engine_reset_count ${NC} \n"

                    echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us" %7s = ${GREEN}
$global_h2g_time_us ${NC} \n"
```

```
                    echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us" %7s = ${GREEN}
$global_irq_time_us ${NC} \n"


                    echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count" %7s =
${GREEN} $global_page_fault_count ${NC} \n"

                    echo $global_pf_policies_sched_if_idle >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle"
                    printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle" %14s =
${GREEN} $global_pf_policies_sched_if_idle ${NC} \n"
                    echo $global_pf_policies_sample_period_ms
> $cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms"
                    printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms" %14s = ${GREEN}
$global_pf_policies_sample_period_ms ${NC} \n"

                    echo $global_pf_policies_engine_reset >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset"
                    printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset" %14s =
${GREEN} $global_pf_policies_engine_reset ${NC} \n"
                fi

            done
            #Configure Number of VFs
            function_debugPrints "Enabling VFs for mode-1 \n"
            echo 0 > $cardPath$IOV_PF_PATH"/device/sriov_drivers_autoprobe"
            echo 1 > $cardPath$IOV_PF_PATH"/device/sriov_numvfs"

        fi

        (( cardNum += 1 ))
    done < "$TEMP_ROOT_DIR$GFX_CARDS_PATH"

    if [ "$foundCard" -eq 0 ]
    then
        printf "*******************************************\n"
        printf "%% \t\t No card found \t\t%% \n"
        printf "******************************************* \n"
    fi

    function_printFuncName ${FUNCNAME[0]} 1
}

function provisionMode2() {
    function_printFuncName ${FUNCNAME[0]} 0
    cardToConfigure=$1
```

```
    vfNum=0
    function_debugPrints "Configuring mode-2 for card:$cardToConfigure \n"

    #Get relative card number.
    cardNum=0
    foundCard=0
        #Go over the gfxCards file and findout number of cards.
        while read -r line;
        do
                #Get the card path
                cardPath=$line
        if [[ $cardPath == *$cardToConfigure ]]
        then
            (( foundCard += 1 ))
            function_debugPrints "Found the card and path is $cardPath\n"

            #Find number of tiles in this card
            numofTilesInCard=$(cat $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum)
            (( numofTilesInCard -= 1 ))
            #Go over all the tiles and provision VFs
            for tileNum in `seq 0 $numofTilesInCard`;
            do
                vfNum=$(($tileNum + 1))




                printf "**************************************************\n"
                printf "* \t ${GREEN} Resources alloted to VF$vfNum ${NC}\t * \n"
                printf "**************************************************\n"

                dbs=0
                ctxs=0
                ggtt=0
                lmem=0

                if [[ $global_productName = "ATSM_75" ]] || [[ $global_productName
= "ATSM_150" ]]
                then
                    dbs=$(cat
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $dbs > $cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota" %10s
= ${GREEN} $dbs ${NC} \n"

                    ctxs=$(cat
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $ctxs > $cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota" %11s =
${GREEN} $ctxs ${NC} \n"

                    ggtt=$(cat
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $ggtt > $cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota" %15s =
${GREEN} $ggtt ${NC} \n"
```

```
                    lmem=$(cat
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $lmem > $cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota" %15s =
${GREEN} $lmem ${NC} \n"

                    echo $global_vfExecQuantInMS >
$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
%10s = ${GREEN} $global_vfExecQuantInMS ${NC} \n"

                    echo $global_vfPreemptQuanInUS >
$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us"
%7s = ${GREEN} $global_vfPreemptQuanInUS ${NC} \n"

                    echo $global_pfExecQuantInMS
> $cardPath$IOV_PF_PATH"/gt/exec_quantum_ms"
                    printf "$cardPath$IOV_PF_PATH"/gt/exec_quantum_ms" %7s
= ${GREEN} $global_pfExecQuantInMS ${NC} \n"

                    echo $global_pfPreemptQuanInUS >
$cardPath$IOV_PF_PATH"/gt/preempt_timeout_us"
                    printf "$cardPath$IOV_PF_PATH"/gt/preempt_timeout_us" %7s
= ${GREEN} $global_pfPreemptQuanInUS ${NC} \n"

                    echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count" %7s = ${GREEN}
$global_cat_error_count ${NC} \n"

                    echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us" %7s = ${GREEN}
$global_doorbell_time_us ${NC} \n"

                    echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count" %7s = ${GREEN}
$global_engine_reset_count ${NC} \n"

                    echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us"
%7s = ${GREEN} $global_h2g_time_us ${NC} \n"

                    echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us"
%7s = ${GREEN} $global_irq_time_us ${NC} \n"

                    echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count"
```

```
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count" %7s = ${GREEN}
$global_page_fault_count ${NC} \n"



                    echo $global_pf_policies_sched_if_idle
> $cardPath$IOV_PF_PATH"/gt/policies/sched_if_idle"
                    printf "$cardPath$IOV_PF_PATH"/gt/policies/sched_if_idle" %14s
= ${GREEN} $global_pf_policies_sched_if_idle ${NC} \n"

                    echo $global_pf_policies_sample_period_ms
> $cardPath$IOV_PF_PATH"/gt/policies/sample_period_ms"
                    printf "$cardPath$IOV_PF_PATH"/gt/policies/sample_period_ms"
%14s = ${GREEN} $global_pf_policies_sample_period_ms ${NC} \n"

                    echo $global_pf_policies_engine_reset
> $cardPath$IOV_PF_PATH"/gt/policies/engine_reset"
                    printf "$cardPath$IOV_PF_PATH"/gt/policies/engine_reset"
%14s = ${GREEN} $global_pf_policies_engine_reset ${NC} \n"

                else
                    dbs=$(cat
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $dbs >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota" %10s = ${GREEN} $dbs
${NC} \n"

                    ctxs=$(cat
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                    echo $ctxs >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota" %11s = ${GREEN} $ctxs
${NC} \n"

                    ggtt=$(cat
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                     echo $ggtt >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota"
%15s = ${GREEN} $ggtt ${NC} \n"

                    lmem=$(cat
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                     echo $lmem >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota"
                    printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota"
%15s = ${GREEN} $lmem ${NC} \n"

                    echo $global_vfExecQuantInMS >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms" %10s =
${GREEN} $global_vfExecQuantInMS ${NC} \n"
```

```
                    echo $global_vfPreemptQuanInUS >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us" %7s = ${GREEN}
$global_vfPreemptQuanInUS ${NC} \n"

                    echo $global_pfExecQuantInMS >
$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
                    printf "$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms" %7s
= ${GREEN} $global_pfExecQuantInMS ${NC} \n"

                    echo $global_pfPreemptQuanInUS >
$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
                    printf "$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
%7s = ${GREEN} $global_pfPreemptQuanInUS ${NC} \n"

                    echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count" %7s = ${GREEN}
$global_cat_error_count ${NC} \n"

                    echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us" %7s =
${GREEN} $global_doorbell_time_us ${NC} \n"

                    echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count" %7s =
${GREEN} $global_engine_reset_count ${NC} \n"

                    echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us" %7s = ${GREEN}
$global_h2g_time_us ${NC} \n"

                    echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us" %7s = ${GREEN}
$global_irq_time_us ${NC} \n"

                    echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count"
                    printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count" %7s =
${GREEN} $global_page_fault_count ${NC} \n"

                    echo $global_pf_policies_sched_if_idle
> $cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle"
```

```
                    printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle" %14s = ${GREEN}
$global_pf_policies_sched_if_idle ${NC} \n"

                    echo $global_pf_policies_sample_period_ms
> $cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms"
                    printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms" %14s = ${GREEN}
$global_pf_policies_sample_period_ms ${NC} \n"

                    echo $global_pf_policies_engine_reset >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset"
                    printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset" %14s =
${GREEN} $global_pf_policies_engine_reset ${NC} \n"
                fi

            done
            #Configure Number of VFs
            function_debugPrints "Enabling VFs for mode-2 \n"
            echo 0 > $cardPath$IOV_PF_PATH"/device/sriov_drivers_autoprobe"
            echo 2 > $cardPath$IOV_PF_PATH"/device/sriov_numvfs"

        fi

        (( cardNum += 1 ))
    done < "$TEMP_ROOT_DIR$GFX_CARDS_PATH"

    if [ "$foundCard" -eq 0 ]
    then
        printf "*********************************************\n"
        printf "%% \t\t No card found \t\t%% \n"
        printf "********************************************* \n"
    fi

    function_printFuncName ${FUNCNAME[0]} 1
}

function provisionMode3() {
    function_printFuncName ${FUNCNAME[0]} 0

    cardToConfigure=$1
    NoOfVfsToBeEnabled=$2
    vfNum=0
    resDivFactor=0
    execQuantaOfEachVM=0
    function_debugPrints "Configuring mode-3 for card:$cardToConfigure \n"

    #Get relative card number.
    cardNum=0
    foundCard=0
        #Go over the gfxCards file and findout number of cards.
        while read -r line;
        do
                #Get the card path
                cardPath=$line
        if [[ $cardPath == *$cardToConfigure ]]
```

```
        then
            (( foundCard += 1 ))
            function_debugPrints "Found the card and path is $cardPath\n"

            #Find number of tiles in this card
            numofTilesInCard=$(cat $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum)


            #Find the resource division factor.
            #We need to divide PF resources equally to all VFs.
            resDivFactor=$(($NoOfVfsToBeEnabled/$numofTilesInCard)
            ) execQuantaOfEachVM=$global_vfExecQuantInMS
            printf "Dividing all resources equally into $resDivFactor
for $numofTilesInCard tiles\n"
            (( numofTilesInCard -= 1 ))
            #Initialize VF number
            vfNum=1
            #Go over all the tiles and provision VFs
            for tileNum in `seq 0 $numofTilesInCard`;
            do

                vfCountOnEachTile=1
                for vfCountOnEachTile in `seq 1 $resDivFactor`;
                do

                    dbs=0
                    ctxs=0
                    ggtt=0
                    lmem=0

                    printf "*************************************************\n"
                    printf "* \t ${GREEN} Resources alloted to VF$vfNum\t ${NC} *
\n"
                    printf "*************************************************\n"

                    if [[ $global_productName = "ATSM_75" ]] ||
[[ $global_productName = "ATSM_150" ]]
                    then
                        dbs=$(cat
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        dbs=$((dbs / resDivFactor))
                        echo $dbs >
$cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota"
                        printf "$cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota"
%10s = ${GREEN} $dbs ${NC} \n"

                        ctxs=$(cat
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        ctxs=$((ctxs / resDivFactor))
                        #Round ctxs to vfCtxsRoundingFactor
                        ctxs=$((ctxs / vfCtxsRoundingFactor))
                        ctxs=$((ctxs * vfCtxsRoundingFactor))
                        echo $ctxs >
$cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota"
                        printf "$cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota"
%11s = ${GREEN} $ctxs ${NC} \n"
```

```
                        ggtt=$(cat
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        ggtt=$((ggtt/resDivFactor))
                        #Round ggtt to vfGgttRoundingFactor
                        ggtt=$((ggtt/vfGgttRoundingFactor))
                        ggtt=$((ggtt*vfGgttRoundingFactor))
                        echo $ggtt > $cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota"
                        printf "$cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota" %15s =
${GREEN} $ggtt ${NC} \n"

                        lmem=$(cat
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        lmem=$((lmem / resDivFactor))
                        #Round lmem to vfLmemRoundingFactor
                        lmem=$((lmem / vfLmemRoundingFactor))
                        lmem=$((lmem * vfLmemRoundingFactor))
                        echo $lmem > $cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota"
                        printf "$cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota" %15s =
${GREEN} $lmem ${NC} \n"

                        echo $execQuantaOfEachVM >
$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
                        printf "$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
%7s = ${GREEN} $execQuantaOfEachVM ${NC} \n"

                        echo $global_vfPreemptQuanInUS >
$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us" %7s = ${GREEN}
$global_vfPreemptQuanInUS ${NC} \n"

                        echo $global_pfExecQuantInMS
> $cardPath$IOV_PF_PATH"/gt/exec_quantum_ms"
                        printf "$cardPath$IOV_PF_PATH"/gt/exec_quantum_ms" %7s
= ${GREEN} $global_pfExecQuantInMS ${NC} \n"

                        echo $global_pfPreemptQuanInUS
> $cardPath$IOV_PF_PATH"/gt/preempt_timeout_us"
                        printf "$cardPath$IOV_PF_PATH"/gt/preempt_timeout_us"
%7s = ${GREEN} $global_pfPreemptQuanInUS ${NC} \n"

                        echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count" %7s =
${GREEN} $global_cat_error_count ${NC} \n"

                        echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us" %7s =
${GREEN} $global_doorbell_time_us ${NC} \n"

                        echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count"
```

```
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count" %7s = ${GREEN}
$global_engine_reset_count ${NC} \n"

                        echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us" %7s = ${GREEN}
$global_h2g_time_us ${NC} \n"

                        echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us" %7s = ${GREEN}
$global_irq_time_us ${NC} \n"

                         echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count" %7s =
${GREEN} $global_page_fault_count ${NC} \n"

                        echo $global_pf_policies_sched_if_idle >
$cardPath$IOV_PF_PATH"/gt/policies/sched_if_idle"
                        printf "$cardPath$IOV_PF_PATH"/gt/policies/sched_if_idle"
%14s = ${GREEN} $global_pf_policies_sched_if_idle ${NC} \n"

                        echo $global_pf_policies_sample_period_ms >
 $cardPath$IOV_PF_PATH"/gt/policies/sample_period_ms"
                         printf
"$cardPath$IOV_PF_PATH"/gt/policies/sample_period_ms" %14s =
${GREEN} $global_pf_policies_sample_period_ms ${NC} \n"

                        echo $global_pf_policies_engine_reset >
$cardPath$IOV_PF_PATH"/gt/policies/engine_reset"
                        printf "$cardPath$IOV_PF_PATH"/gt/policies/engine_reset"
%14s = ${GREEN} $global_pf_policies_engine_reset ${NC} \n"
                    else
                         dbs=$(cat
$TEMP_ROOT_DIR$DB_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        dbs=$((dbs / resDivFactor))
                        echo $dbs >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota"
                         printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota" %10s = ${GREEN}
$dbs ${NC} \n"

                         ctxs=$(cat
$TEMP_ROOT_DIR$CTX_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        ctxs=$((ctxs / resDivFactor))
                        #Round ctxs to vfCtxsRoundingFactor
                        ctxs=$((ctxs / vfCtxsRoundingFactor))
                        ctxs=$((ctxs * vfCtxsRoundingFactor))
                        echo $ctxs >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota"
```

```
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota" %11s = ${GREEN} $ctxs
${NC} \n"

                         ggtt=$(cat
$TEMP_ROOT_DIR$GGTT_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        ggtt=$((ggtt/resDivFactor))
                        #Round ggtt to vfGgttRoundingFactor
                        ggtt=$((ggtt/vfGgttRoundingFactor))
                        ggtt=$((ggtt*vfGgttRoundingFactor))
                        echo $ggtt >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota" %15s = ${GREEN} $ggtt
${NC} \n"


                         lmem=$(cat
$TEMP_ROOT_DIR$LMEM_MAX_QUOTA"_card"$cardNum"_gt"$tileNum)
                        lmem=$((lmem / resDivFactor))
                        #Round lmem to vfLmemRoundingFactor
                        lmem=$((lmem / vfLmemRoundingFactor))
                        lmem=$((lmem * vfLmemRoundingFactor))
                        echo $lmem >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota" %15s = ${GREEN} $lmem
${NC} \n"

                        echo $execQuantaOfEachVM >
 $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms"
                         printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms" %10s =
${GREEN} $execQuantaOfEachVM ${NC} \n"

                         echo $global_vfPreemptQuanInUS >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us" %7s =
${GREEN} $global_vfPreemptQuanInUS ${NC} \n"

                        echo $global_pfExecQuantInMS
> $cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
                        printf "$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
%7s = ${GREEN} $global_pfExecQuantInMS ${NC} \n"


                        echo $global_pfPreemptQuanInUS
> $cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
                        printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us" %7s = ${GREEN}
$global_pfPreemptQuanInUS ${NC} \n"


                         echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count" %7s = ${GREEN}
$global_cat_error_count ${NC} \n"
```

```
                        echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us" %7s
= ${GREEN} $global_doorbell_time_us ${NC} \n"
                        echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count" %7s =
${GREEN} $global_engine_reset_count ${NC} \n"

                        echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us" %7s =
${GREEN} $global_h2g_time_us ${NC} \n"
                        echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us" %7s =
${GREEN} $global_irq_time_us ${NC} \n"
                        echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count"
                        printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count" %7s
= ${GREEN} $global_page_fault_count ${NC} \n"
                        echo $global_pf_policies_sched_if_idle >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle"
                        printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle" %14s =
${GREEN} $global_pf_policies_sched_if_idle ${NC} \n"
                        echo $global_pf_policies_sample_period_ms >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms"
                        printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms" %14s = ${GREEN}
$global_pf_policies_sample_period_ms ${NC} \n"
                        echo $global_pf_policies_engine_reset >
 $cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset"
                         printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset" %14s =
${GREEN} $global_pf_policies_engine_reset ${NC} \n"
                    fi
                    (( vfNum += 1 ))
                done
            done
```

```
                #Configure Number of VFs
                function_debugPrints "Enabling VFs for mode-3 \n"
                echo 0 > $cardPath$IOV_PF_PATH"/device/sriov_drivers_autoprobe"
                echo $NoOfVfsToBeEnabled > $cardPath$IOV_PF_PATH"/device/sriov_numvfs"
                #echo "2 > $cardPath$IOV_PF_PATH"/device/sriov_numvfs""

            fi

            (( cardNum += 1 ))
        done < "$TEMP_ROOT_DIR$GFX_CARDS_PATH"

        if [ "$foundCard" -eq 0 ]
        then
            printf "*********************************************\n"
            printf "%% \t\t No card found \t\t%% \n"
            printf "********************************************* \n"
        fi

        function_printFuncName ${FUNCNAME[0]} 1
    }

    function function_provisionVfsAuto() {
        function_printFuncName ${FUNCNAME[0]} 0
        card=$1
        mode=$2
        Vfs=$3

        function_debugPrints "mode = $mode\n"
        function_debugPrints "Vfs = $Vfs\n"

        function_getNumOfCards
        function_getPfResources

        #provision for mode-1
        if [ "$mode" -eq 1 ]
        then
            provisionMode1 $card $Vfs

        elif [ "$mode" -eq 2 ]
        then
            provisionMode2 $card $Vfs

        elif [ "$mode" -eq 3 ]
        then
            provisionMode3 $card $Vfs

        fi
        function_printFuncName ${FUNCNAME[0]} 1
    }

    function function_provisionVfsManual() {

        function_printFuncName ${FUNCNAME[0]} 0
        #Get the card path send as input
        InCard=$1

        #Get the number of cards present
```

```
function_getNumOfCards

#Find the exact card path from sysfs.
cardNum=0
foundCard=0
numOfTilesInCard=0



    #Go over the gfxCards file and findout number of cards.
    while read -r line;
    do
            #Get the card path
            cardPath=$line

    if [[ $cardPath == *$InCard ]]
    then
        (( foundCard += 1 ))
        function_debugPrints "Found Card path: $cardPath\n"
        numOfTilesInCard=$(cat
        $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum) break
    fi

    (( cardNum+=1 ))
done < "$TEMP_ROOT_DIR$GFX_CARDS_PATH"

if [ "$foundCard" -eq 0 ]
    then
            printf "*********************************************\n"
            printf "%% \t\t No card found \t\t%% \n"
            printf "********************************************* \n"
    fi

#Configure resources
while read -r line;
do
    vfNum=$line
    function_debugPrints "vfNum=$vfNum\n"
    printf "************************************************\n"
    printf "* \t ${GREEN} Resources alloted to VF$vfNum\t ${NC} * \n"
    printf "************************************************\n"

    #get Tile number to which this VF belongs to
    tileNum=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_TILE_PATH)

    if [ $numOfTilesInCard -le $tileNum ]
    then
        printf "${RED}Error in provided tile Num of Custom params\n"
        printf "Configuration is incomplete${NC}\n"

        function_cleanUpTempData
        function_cleanUpTempDataCustom
        exit -1
    fi
```

```
        if [[ $global_productName = "ATSM_75" ]] || [[ $global_productName
= "ATSM_150" ]]
        then
            dbs=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_DBS_PATH) echo
            $dbs > $cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota" printf
            "$cardPath$IOV_VF_PATH$vfNum"/gt/doorbells_quota" %10s =
${GREEN} $dbs ${NC} \n"

            ctxs=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_CTXS_PATH) echo
            $ctxs > $cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota" printf
            "$cardPath$IOV_VF_PATH$vfNum"/gt/contexts_quota" %11s =
${GREEN} $ctxs ${NC} \n"

            ggtt=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_GGTT_PATH)
            echo $ggtt > $cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/ggtt_quota" %15s =
${GREEN} $ggtt ${NC} \n"

            lmem=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_LMEM_PATH)
            echo $lmem > $cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/lmem_quota" %15s =
${GREEN} $lmem ${NC} \n"

            exec_quanta=$(cat
$TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_EXEC_QUOTA_PATH)
            echo $exec_quanta> $cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/exec_quantum_ms" %10s =
${GREEN} $exec_quanta ${NC} \n"

            preempt_quota=$(cat
$TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_PREEMPT_QUOTA_PATH)
            echo $preempt_quota >
$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/preempt_timeout_us" %7s =
${GREEN} $preempt_quota ${NC} \n"

            echo $global_pfExecQuantInMS >
$cardPath$IOV_PF_PATH"/gt/exec_quantum_ms"
            printf "$cardPath$IOV_PF_PATH"/gt/exec_quantum_ms" %7s = ${GREEN}
$global_pfExecQuantInMS ${NC} \n"

            echo $global_pfPreemptQuanInUS >
$cardPath$IOV_PF_PATH"/gt/preempt_timeout_us
"
            printf "$cardPath$IOV_PF_PATH"/gt/preempt_timeout_us" %7s =
${GREEN} $global_pfPreemptQuanInUS ${NC} \n"

            echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/cat_error_count" %7s
= ${GREEN} $global_cat_error_count ${NC} \n"

            echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/doorbell_time_us"
%7s = ${GREEN} $global_doorbell_time_us ${NC} \n"
```

```
            echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/engine_reset_count"
%7s = ${GREEN} $global_engine_reset_count ${NC} \n"

            echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/h2g_time_us" %7s
= ${GREEN} $global_h2g_time_us ${NC} \n"

            echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/irq_time_us" %7s
= ${GREEN} $global_irq_time_us ${NC} \n"

            echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt/threshold/page_fault_count"
%7s = ${GREEN} $global_page_fault_count ${NC} \n"


        else
            dbs=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_DBS_PATH)
            echo $dbs > $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/doorbells_quota" %10s
= ${GREEN} $dbs ${NC} \n"

            ctxs=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_CTXS_PATH)
            echo $ctxs > $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/contexts_quota" %11s =
${GREEN} $ctxs ${NC} \n"

            ggtt=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_GGTT_PATH) echo
            $ggtt > $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota" printf
            "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/ggtt_quota" %15s =
${GREEN} $ggtt ${NC} \n"

            lmem=$(cat $TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_LMEM_PATH) echo
            $lmem > $cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota" printf
            "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/lmem_quota" %15s =
${GREEN} $lmem ${NC} \n"

            exec_quanta=$(cat
$TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_EXEC_QUOTA_PATH)
            echo $exec_quanta>
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/exec_quantum_ms"
%10s = ${GREEN} $exec_quanta ${NC} \n"

            preempt_quota=$(cat
$TEMP_ROOT_DIR_CUSTOM_DATA"/"$vfNum$IN_PREEMPT_QUOTA_PATH)
            echo $preempt_quota >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/preempt_timeout_us"
%7s = ${GREEN} $preempt_quota ${NC} \n"
```

```
            echo $global_pfExecQuantInMS >
$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
            printf "$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms" %7s
= ${GREEN} $global_pfExecQuantInMS ${NC} \n"

            echo $global_pfPreemptQuanInUS >
$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
            printf "$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us" %7s
= ${GREEN} $global_pfPreemptQuanInUS ${NC} \n"


            echo $global_cat_error_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count"
            printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/cat_error_count" %7s = ${GREEN}
$global_cat_error_count ${NC} \n"

            echo $global_doorbell_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us"
            printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/doorbell_time_us" %7s =
${GREEN} $global_doorbell_time_us ${NC} \n"

            echo $global_engine_reset_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count"
            printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/engine_reset_count" %7s =
${GREEN} $global_engine_reset_count ${NC} \n"

            echo $global_h2g_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/h2g_time_us"
%7s = ${GREEN} $global_h2g_time_us ${NC} \n"

            echo $global_irq_time_us >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us"
            printf "$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/irq_time_us"
%7s = ${GREEN} $global_irq_time_us ${NC} \n"

            echo $global_page_fault_count >
$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count"
            printf
"$cardPath$IOV_VF_PATH$vfNum"/gt$tileNum/threshold/page_fault_count" %7s =
${GREEN} $global_page_fault_count ${NC} \n"

            echo $global_pf_policies_sched_if_idle >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle"
            printf "$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sched_if_idle" %14s
= ${GREEN} $global_pf_policies_sched_if_idle ${NC} \n"

            echo $global_pf_policies_sample_period_ms >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms"
            printf "$cardPath$IOV_PF_PATH"/gt$tileNum/policies/sample_period_ms"
%14s = ${GREEN} $global_pf_policies_sample_period_ms ${NC} \n"

            echo $global_pf_policies_engine_reset >
$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset"
```

```
            printf "$cardPath$IOV_PF_PATH"/gt$tileNum/policies/engine_reset" %14s
= ${GREEN} $global_pf_policies_engine_reset ${NC}
        \n" fi
    done < "$TEMP_ROOT_DIR_CUSTOM_DATA"/"$IN_VF_NUMS_PATH"

    printf "Enabling All the VFs\n"
    echo 0 > $cardPath$IOV_PF_PATH"/device/sriov_drivers_autoprobe"
    echo $global_maxVfs > $cardPath$IOV_PF_PATH"/device/sriov_numvfs"

    function_printFuncName ${FUNCNAME[0]} 1
}
```

- Create a file **provision_from_config_files.sh** and add the following lines of scripts:

```
#!/bin/bash

source ./provisioning.sh

CONFIG_FILE_PATH="config_files/"
ATSM75_CONFIG_FILE_INT_NAME="ATSM/ATSM75_int.csv"
ATSM75_CONFIG_FILE_VFS_NAME="ATSM/ATSM75_vfs.csv"
ATSM150_CONFIG_FILE_INT_NAME="ATSM/ATSM150_int.csv"
ATSM150_CONFIG_FILE_VFS_NAME="ATSM/ATSM150_vfs.csv"
PVC_CONFIG_1T_FILE_INT_NAME="PVC/PVC1_int.csv"
PVC_CONFIG_1T_FILE_VFS_NAME="PVC/PVC1_vfs.csv"
PVC_CONFIG_2T_FILE_INT_NAME="PVC/PVC2_int.csv"
PVC_CONFIG_2T_FILE_VFS_NAME="PVC/PVC2_vfs.csv"

PROVISION_FROM_CONFIG_SCRIPT_VERSION="1.1" printf
"${RED}Provision From Config Files Script
Version:${GREEN}$PROVISION_FROM_CONFIG_SCRIPT_VERSION ${NC}\n"



vfLmemEccEffective_percentage=85
pfLmemEccEffective_percentage=15
roundPageSize_64k=65536

declare global_card_path
declare global_no_of_tiles
declare global_mem_ecc_enable


function function_getCardPath() {
    #Get relative card number.
    cardNum=0
    foundCard=0
    #Go over the gfxCards file and findout number of cards.
    while read -r line;
```

```
    do
        #Get the card path
        cardPath=$line
        if [[ $cardPath == *$cardToConfigure ]]
        then
            (( foundCard += 1 ))
            function_debugPrints "Found the card and path is $cardPath\n"
            global_card_path=$cardPath
        fi
        (( cardNum += 1 ))
    done < "$TEMP_ROOT_DIR$GFX_CARDS_PATH"
}

: '
#################
Input Params for func_config_vfs_fileData

param[0] = function name
param[1] = card number
param[2] = product Name
param[3] = Filepath
param[4] = In vgpu profile id (M1/M3 etc..)
################
'

function func_config_vfs_fileData () {
    function_printFuncName ${FUNCNAME[0]} 0



    function_getCardPath $1
    cardPath=$global_card_path

    numvfs=$(cat $cardPath/device/sriov_totalvfs)
    productId=$2
    INPUT=$3
    InVgpuProfileId=$4


    profileType="${InVgpuProfileId:0:1}" printf
    "profileType=$profileType\n"
    numOfVfsFromInVgpuProfileId=${InVgpuProfileId:1}
    printf "numOfVfsFromInVgpuProfileId=$numOfVfsFromInVgpuProfileId\n"

    OLDIFS=$IFS
    IFS=','
    n=$((numvfs+0))

    #PROCESS THE PER_VF VALUES in <>_vfs.csv
    line=1
    [ ! -f $INPUT ] && { echo "$INPUT file not found"; exit 99; }
    while read profid profdesc schedmode pfschedexecq pfschedtimeout
vfschedexecq vfschedtimeout schedifidle
    do

        function_debugPrints "######################"
```

```
        function_debugPrints "profid=$profid"
        function_debugPrints "profdesc=$profdesc"
        function_debugPrints "schedmode=$schedmode"
        function_debugPrints "pfschedexecq=$pfschedexecq"
        function_debugPrints "pfschedtimeout=$pfschedtimeout"
        function_debugPrints "vfschedexecq=$vfschedexecq"
        function_debugPrints "vfschedtimeout=$vfschedtimeout"
        function_debugPrints "schedifidle=$schedifidle"
        function_debugPrints "######################"

        test $line -eq 1 && ((line=line+1)) && continue
        NVFS_OFFSET=0
        TTYPE_OFFSET=0

        if [ "$productId" = "PVC" ]
        then
            NVFS_OFFSET=6

        elif [ "$productId" = "ATSM_75" ]
        then
            NVFS_OFFSET=8

        elif [ "$productId" = "ATSM_150" ]
        then
            NVFS_OFFSET=9

        fi

        NVFS=${profid:$NVFS_OFFSET}
        TYPE=${profid:$NVFS_OFFSET-1:1}

        n=$(($NVFS+0))
        if [ $TYPE = $profileType ] ; then
            if (( $n==$numOfVfsFromInVgpuProfileId )) ; then
                ##ATS-M specific programming.
                if [[ "$productId" = "ATSM_75" ]] || [[ "$productId" = "ATSM_150"
]]
                then
                    echo $pfschedexecq >
$cardPath$IOV_PF_PATH"/gt/exec_quantum_ms"
                    printf "$cardPath$IOV_PF_PATH/gt/exec_quantum_ms %10s
= ${GREEN} $pfschedexecq ${NC}\n"



                    echo $pfschedtimeout >
$cardPath$IOV_PF_PATH/gt/preempt_timeout_us
                    printf "$cardPath$IOV_PF_PATH/gt/preempt_timeout_us %10s
= ${GREEN} $pfschedtimeout ${NC}\n"
                    if [ $schedifidle = 'F' ] ; then
                        schedifidle=1
                    else
                        schedifidle=0
                    fi
```

```
                    echo $schedifidle >
$cardPath$IOV_PF_PATH/gt/policies/sched_if_idle
                    printf "$cardPath$IOV_PF_PATH/gt/policies/sched_if_idle %10s =
${GREEN} $schedifidle ${NC}\n"

                    for (( i = 1; i <= $n; i++ ))
                    do
                        echo $vfschedexecq >
$cardPath$IOV_VF_PATH$i/gt/exec_quantum_ms
                        printf "$cardPath$IOV_VF_PATH$i/gt/exec_quantum_ms %10s
= ${GREEN} $vfschedexecq ${NC}\n"

                        echo $vfschedtimeout >
$cardPath$IOV_VF_PATH$i/gt/preempt_timeout_us
                        printf "$cardPath$IOV_VF_PATH$i/gt/preempt_timeout_us %10s
=  ${GREEN} $vfschedtimeout
                    ${NC}\n" done
                    ##PVC specific programming.
                elif [[ "$productId" = "PVC" ]]
                then
                    numofTilesInCard=$global_no_of_tiles
                     #We need to divide PF resources equally to all VFs.
                     resDivFactor=$(($n/$numofTilesInCard))
                    (( numofTilesInCard -= 1 ))

                    #For mode-1

                    if [ $NVFS -eq 1 ]
                    then
                        vfNum=1
                        for (( tileNum=0;tileNum<$global_no_of_tiles;tileNum++ ))
                        do
                            printf
"${YELLOW}**************************************************${NC}\n"
                            printf "* \t ${YELLOW} Resources alloted to
tile$tileNum\t ${NC} * \n"
                            printf
"${YELLOW}**************************************************${NC}\n"

                            echo $pfschedexecq >
$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
                            printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms" %10s = ${GREEN} $pfschedexecq
${NC}\n"

                             echo $pfschedtimeout >
$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/preempt_timeout_us %10s = ${GREEN}
$pfschedtimeout ${NC}\n"

                            if [ $schedifidle = 'F' ] ; then
                                schedifidle=1
                            else
                                 schedifidle=0
                            fi
```

```
                            echo $schedifidle >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/sched_if_idle
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/sched_if_idle %10s = ${GREEN}
$schedifidle ${NC}\n"

                            echo $vfschedexecq >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/exec_quantum_ms
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/exec_quantum_ms %10s = ${GREEN}
$vfschedexecq ${NC}\n"

                             echo $vfschedtimeout >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/preempt_timeout_us
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/preempt_timeout_us %10s = ${GREEN}
$vfschedtimeout ${NC}\n"

                        done



                    #For mode-2
                    elif [ $NVFS -eq 2 ]
                    then
                        vfNum=1
                        for (( tileNum=0;tileNum<$global_no_of_tiles;tileNum++ ))
                        do
                            printf
"${YELLOW}*************************************************${NC}\n"
                            printf "* \t ${YELLOW} Resources alloted to
tile$tileNum\t ${NC} * \n"
                            printf
"${YELLOW}*************************************************${NC}\n"
                            vfNum=$(($tileNum + 1))

                            echo $pfschedexecq >
$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
                            printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms" %10s = ${GREEN} $pfschedexecq
${NC}\n"

                             echo $pfschedtimeout >
$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/preempt_timeout_us %10s = ${GREEN}
$pfschedtimeout ${NC}\n"

                            if [ $schedifidle = 'F' ] ; then
                                schedifidle=1
                            else
                                 schedifidle=0
                            fi

                            echo $schedifidle >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/sched_if_idle
```

```
                                printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/sched_if_idle %10s =  ${GREEN}
$schedifidle ${NC}\n"

                                echo $vfschedexecq >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/exec_quantum_ms
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/exec_quantum_ms %10s = ${GREEN}
$vfschedexecq ${NC}\n"

                                 echo $vfschedtimeout >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/preempt_timeout_us
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/preempt_timeout_us %10s = ${GREEN}
$vfschedtimeout ${NC}\n"
                            done

                        #For mode-3
                        elif [ $NVFS -ge 3 ]
                        then
                            vfNum=1
                            resDivFactor=$(($n/$global_no_of_tiles))
                            for (( tileNum=0;tileNum<$global_no_of_tiles;tileNum++ ))
                            do
                                printf
"${YELLOW}*************************************************${NC}\n"
                                printf "* \t ${YELLOW} Resources alloted to
tile$tileNum\t ${NC} * \n"
                                printf
"${YELLOW}*************************************************${NC}\n"

                                echo $pfschedexecq >
$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms"
                                printf
"$cardPath$IOV_PF_PATH"/gt$tileNum/exec_quantum_ms" %10s = ${GREEN} $pfschedexecq
${NC}\n"

                                 echo $pfschedtimeout >
$cardPath$IOV_PF_PATH"/gt$tileNum/preempt_timeout_us"
                                printf
"$cardPath$IOV_PF_PATH/gt$tileNum/preempt_timeout_us %10s = ${GREEN}
$pfschedtimeout ${NC}\n"

                                if [ $schedifidle = 'F' ] ; then
                                    schedifidle=1
                                else
                                     schedifidle=0
                                fi

                                echo $schedifidle >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/sched_if_idle
                                printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/sched_if_idle %10s = ${GREEN}
$schedifidle ${NC}\n"
```

```
                            vfCountOnEachTile=1
                            echo $vfCountOnEachTile $resDivFactor $vfNum
                            for ((
vfCountOnEachTile=1;vfCountOnEachTile<=$resDivFactor; vfCountOnEachTile++ ));
                            do
                                echo $vfschedexecq >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/exec_quantum_ms
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/exec_quantum_ms %10s = ${GREEN}
$vfschedexecq ${NC}\n"

                                echo $vfschedtimeout >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/preempt_timeout_u
                                s printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/preempt_timeout_us %10s = ${GREEN}
$vfschedtimeout ${NC}\n"

                                 (( vfNum += 1 ))
                            done
                        done
                    else
                        echo "error in the received VF number to be enabled"
                    fi

                #Error case
                else
                    echo "Error in received product id\n"

                fi # End of productId

            fi # End of if (( $n==$numOfVfsFromInVgpuProfileId ))

        fi # End of if [ $TYPE = $profileType ]

    done < $INPUT #Endof while read profid profdesc schedmode
pfschedexecq pfschedtimeout vfschedexecq vfschedtimeout schedifidle

    function_printFuncName ${FUNCNAME[0]} 1

    IFS=$OLDIFS
}

: '
#################
Input Params for func_config_int_fileData

param[0] = function name
param[1] = card number
param[2] = product Name
param[3] = Filepath
param[4] = In vgpu profile id (M1/M3 etc..)
################
'


function func_config_int_fileData () {
```

```
    function_printFuncName ${FUNCNAME[0]} 0

    function_getCardPath $1
    cardPath=$global_card_path

    numvfs=$(cat $cardPath/device/sriov_totalvfs)
    productId=$2
    INPUT=$3
    InVgpuProfileId=$4


    profileType="${InVgpuProfileId:0:1}" printf
    "profileType=$profileType\n"
    numOfVfsFromInVgpuProfileId=${InVgpuProfileId:1}
    printf "numOfVfsFromInVgpuProfileId=$numOfVfsFromInVgpuProfileId\n"



    OLDIFS=$IFS
    IFS=','
    n=$((numvfs+0))

    if [ "$productId" = "PVC" ]
    then
        NVFS_OFFSET=6
        #global_no_of_tiles=$(cat $TEMP_ROOT_DIR$TILES_GFX_CARD$cardNum)

    elif [ "$productId" = "ATSM_75" ]
    then
        NVFS_OFFSET=8

    elif [ "$productId" = "ATSM_150" ]
    then
        NVFS_OFFSET=9

    fi

    line=1
    [ ! -f $INPUT ] && { echo "$INPUT file not found"; exit 99; }
    while read profid schedreset provmode pflmem pfcontexts pfdoorbells
pfggtt vflmem vfcontexts vfdoorbells vfggtt ae_sample_period \
        ae_gt_caterr ae_g2pfnc_caterr ae_g2pfnf_caterr \
        ae_gt_pagefault ae_g2pfnc_pagefault ae_g2pfnf_pagefault \
        ae_gt_h2gstorm ae_g2pfnc_h2gstorm ae_g2pfnf_h2gstorm \
        ae_gt_dbstorm ae_g1pfnc_dbstorm ae_g2pfnf_dbstorm \
        ae_gt_irqstorm ae_g2pfnc_irqstorm ae_g2pfnf_irqstorm \
        ae_gt_enginereset ae_g2pfnc_enginereset ae_g2pfnf_enginereset
    do
        function_debugPrints "profid=$profid"
        function_debugPrints "schedreset=$schedreset"
        function_debugPrints "provmode=$provmode"
        function_debugPrints "pflmem=$pflmem"
        function_debugPrints "pfcontexts=$pfcontexts"
        function_debugPrints "pfdoorbells=$pfdoorbells"
        function_debugPrints "pfggtt=$pfggtt"
        function_debugPrints "vflmem=$vflmem"
        function_debugPrints "vfcontexts=$vfcontexts"
```

```
function_debugPrints "vfdoorbells=$vfdoorbells"
function_debugPrints "vfggtt=$vfggtt"
function_debugPrints "ae_sample_period=$ae_sample_period"
function_debugPrints "ae_gt_caterr=$ae_gt_caterr"
function_debugPrints "ae_g2pfnc_caterr=$ae_g2pfnc_caterr"
function_debugPrints "ae_g2pfnf_caterr=$ae_g2pfnf_caterr"
function_debugPrints "ae_gt_pagefault=$ae_gt_pagefault"
function_debugPrints "ae_g2pfnc_pagefault=$ae_g2pfnc_pagefault"
function_debugPrints "ae_g2pfnf_pagefault=$ae_g2pfnf_pagefault"
function_debugPrints "ae_gt_h2gstorm=$ae_gt_h2gstorm"
function_debugPrints "ae_g2pfnc_h2gstorm=$ae_g2pfnc_h2gstorm"
function_debugPrints "ae_g2pfnf_h2gstorm=$ae_g2pfnf_h2gstorm"
function_debugPrints "ae_gt_dbstorm=$ae_gt_dbstorm"
function_debugPrints "ae_g1pfnc_dbstorm=$ae_g1pfnc_dbstorm"
function_debugPrints "ae_g2pfnf_dbstorm=$ae_g2pfnf_dbstorm"
function_debugPrints "ae_gt_irqstorm=$ae_gt_irqstorm"
function_debugPrints "ae_g2pfnc_irqstorm=$ae_g2pfnc_irqstorm"
function_debugPrints "ae_g2pfnf_irqstorm=$ae_g2pfnf_irqstorm"
function_debugPrints "ae_gt_enginereset=$ae_gt_enginereset"
function_debugPrints "ae_g2pfnc_enginereset=$ae_g2pfnc_enginereset"
function_debugPrints "ae_g2pfnf_enginereset=$ae_g2pfnf_enginereset"

test $line -eq 1 && ((line=line+1)) && continue
NVFS=${profid:$NVFS_OFFSET}
TYPE=${profid:$NVFS_OFFSET-1:1} n=$(($NVFS+0))



if [ $TYPE = $profileType ] ; then
    if (( $n==$numOfVfsFromInVgpuProfileId )) ; then
        if [ $ae_gt_caterr='Max value' ]; then
            ae_gt_caterr=0
        fi
        if [ $ae_gt_pagefault='Max value' ]; then
            ae_gt_pagefault=0
        fi
        if [ $ae_gt_h2gstorm='Max value' ]; then
            ae_gt_h2gstorm=0
        fi
        if [ $ae_gt_dbstorm='Max value' ]; then
            ae_gt_dbstorm=0
        fi
        if [ $ae_gt_irqstorm='Max value' ]; then
            ae_gt_irqstorm=0
        fi
        if [ $ae_gt_enginereset='Max value' ]; then
            ae_gt_enginereset=0
        fi

        if [ $schedreset = 'F' ] ; then
            schedreset=0
        else
            schedreset=1
        fi
```

```
                  ##ATS-M specific programming.
                 if [[ "$productId" = "ATSM_75" ]] || [[ "$productId" = "ATSM_150"
]]
                 then

                     echo $schedreset >
$cardPath$IOV_PF_PATH/gt/policies/engine_reset
                     printf "$cardPath$IOV_PF_PATH/gt/policies/engine_reset %10s
= ${GREEN} $schedreset ${NC}\n"
                     #echo "echo '$provmode' | tee -a $cardPath$IOV_PF_PATH/gt/"
                     #echo "echo '$pflmem' | tee -a $cardPath$IOV_PF_PATH/gt/"
                     #echo $pfcontexts > $cardPath$IOV_PF_PATH/gt/contexts_quota
                     #printf "$cardPath$IOV_PF_PATH/gt/contexts_quota %10s =
${GREEN} $pfcontexts ${NC}\n"

                     echo $pfdoorbells > $cardPath$IOV_PF_PATH/gt/doorbells_quota
                     printf "$cardPath$IOV_PF_PATH/gt/doorbells_quota %10s =
${GREEN} $pfdoorbells ${NC}\n"
                     #echo "echo '$pfggtt' | tee -a
$cardPath$IOV_PF_PATH/gt/pfexecggtt"

                     echo $ae_sample_period >
$cardPath$IOV_PF_PATH/gt/policies/sample_period_ms
                     printf "$cardPath$IOV_PF_PATH/gt/policies/sample_period_ms
%10s = ${GREEN} $ae_sample_period ${NC}\n"
                     for (( i = 1; i <= $n; i++ ))
                     do

                         printf
"************************************************\n"
                         printf "* \t ${GREEN} Resources alloted to VF$i\t ${NC} *
\n"
                          printf
"************************************************\n"

                         echo $vfdoorbells >
$cardPath$IOV_VF_PATH$i/gt/doorbells_quota
                         printf "$cardPath$IOV_VF_PATH$i/gt/doorbells_quota %10s
= ${GREEN} $vfdoorbells ${NC}\n"
                         echo $vfcontexts >
$cardPath$IOV_VF_PATH$i/gt/contexts_quota
                         printf "$cardPath$IOV_VF_PATH$i/gt/contexts_quota %10s
= ${GREEN} $vfcontexts ${NC}\n"
                         if (( $global_mem_ecc_enable == 1 )); then
                             vfLmemCalculated=$(($(($vflmem *
$vfLmemEccEffective_percentage / 100))-$(($pflmem * $pfLmemEccEffective_percentage
/ (100 * $n)))))
                         else
                              vfLmemCalculated=$vflmem
                         fi



                         #Align LMEM size
```

```
                        AlignedLmem=$(($vfLmemCalculated/ $roundPageSize_64k *
$roundPageSize_64k))
                        if [ $AlignedLmem -ne $vfLmemCalculated ]; then
                            printf "${RED} LMEM size $vfLmemCalculated is aligned
to $AlignedLmem ${NC} \n"
                        fi

                        echo $AlignedLmem > $cardPath$IOV_VF_PATH$i/gt/lmem_quota
                        printf "$cardPath$IOV_VF_PATH$i/gt/lmem_quota %10s =
${GREEN} $AlignedLmem ${NC}\n"

                        #Align the GGTT size
                        AlignedGgtt=$(($vfggtt / $roundPageSize_64k *
$roundPageSize_64k))
                        if [ $AlignedGgtt -ne $vfggtt ]; then
                            printf "${RED} GGTT size $vfggtt is aligned to
$AlignedGgtt ${NC} \n"
                        fi
                        echo $AlignedGgtt > $cardPath$IOV_VF_PATH$i/gt/ggtt_quota
                        printf "$cardPath$IOV_VF_PATH$i/gt/ggtt_quota %10s =
${GREEN} $AlignedGgtt ${NC}\n"

                        echo $ae_gt_caterr >
 $cardPath$IOV_VF_PATH$i/gt/threshold/cat_error_count
                         printf
"$cardPath$IOV_VF_PATH$i/gt/threshold/cat_error_count %10s = ${GREEN}
$ae_gt_caterr ${NC}\n"

                        echo $ae_gt_dbstorm >
$cardPath$IOV_VF_PATH$i/gt/threshold/doorbell_time_us
                        printf
"$cardPath$IOV_VF_PATH$i/gt/threshold/doorbell_time_us %10s =  ${GREEN}
$ae_gt_dbstorm ${NC}\n"

                        echo $ae_gt_enginereset >
$cardPath$IOV_VF_PATH$i/gt/threshold/engine_reset_count
                        printf
"$cardPath$IOV_VF_PATH$i/gt/threshold/engine_reset_count %10s =  ${GREEN}
$ae_gt_enginereset ${NC}\n"

                        echo $ae_gt_h2gstorm >
$cardPath$IOV_VF_PATH$i/gt/threshold/h2g_time_us
                        printf "$cardPath$IOV_VF_PATH$i/gt/threshold/h2g_time_us
%10s = ${GREEN} $ae_gt_h2gstorm ${NC}\n"

                        echo $ae_gt_irqstorm >
$cardPath$IOV_VF_PATH$i/gt/threshold/irq_time_us
                        printf "$cardPath$IOV_VF_PATH$i/gt/threshold/irq_time_us
%10s = ${GREEN} $ae_gt_irqstorm ${NC}\n"

                        echo $ae_gt_pagefault >
$cardPath$IOV_VF_PATH$i/gt/threshold/page_fault_count
                        printf
"$cardPath$IOV_VF_PATH$i/gt/threshold/page_fault_count %10s =  ${GREEN}
$ae_gt_pagefault ${NC}\n"
                    done
                    ##PVC specific programming.
```

```
                elif [[ "$productId" = "PVC" ]]
                then

                    numofTilesInCard=$global_no_of_tiles
                    #We need to divide PF resources equally to all VFs.
                    resDivFactor=$(($n/$numofTilesInCard))
                    (( numofTilesInCard -= 1 ))

                    #For mode-1

                    if [ $NVFS -eq 1 ]
                    then
                        vfNum=1
                        #for tileNum in {0,$numofTilesInCard} ;
                        for (( tileNum=0;tileNum<$global_no_of_tiles;tileNum++ ))
                        do

                            printf
"${YELLOW}************************************************${NC}\n"
                            printf "* \t ${YELLOW} Resources alloted to
tile$tileNum\t ${NC} * \n"
                            printf
"${YELLOW}************************************************${NC}\n"

                            echo $schedreset >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/engine_reset
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/engine_reset %10s = ${GREEN}
$schedreset ${NC}\n"

                            echo $pfdoorbells >
$cardPath$IOV_PF_PATH/gt$tileNum/doorbells_quota
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/doorbells_quota %10s =  ${GREEN} $pfdoorbells
${NC}\n"


                             echo $ae_sample_period >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/sample_period_m
                            s printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/sample_period_ms %10s = ${GREEN}
$ae_sample_period ${NC}\n"


                            printf
"************************************************\n"
                            printf "* \t ${GREEN} Resources alloted to VF$vfNum\t
${NC} * \n"
                            printf
"************************************************\n"

                            echo $vfdoorbells >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/doorbells_quota
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/doorbells_quota %10s = ${GREEN}
$vfdoorbells ${NC}\n"
```

```
                            echo $vfcontexts >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/contexts_quot
                            a printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/contexts_quota %10s = ${GREEN}
$vfcontexts ${NC}\n"

                            vfLmemCalculated=$vflmem
                            #Align LMEM size
                            AlignedLmem=$(($vfLmemCalculated/ $roundPageSize_64k *
$roundPageSize_64k))
                            if [ $AlignedLmem -ne $vfLmemCalculated ]; then
                                printf "${RED} LMEM size $vfLmemCalculated is
aligned to $AlignedLmem ${NC} \n"
                            fi
                            echo $AlignedLmem >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/lmem_quota
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/lmem_quota = ${GREEN}
$AlignedLmem ${NC}\n"

                            #Align the GGTT size
                            AlignedGgtt=$(($vfggtt / $roundPageSize_64k *
$roundPageSize_64k))
                            if [ $AlignedGgtt -ne $vfggtt ]; then
                                printf "${RED} GGTT size $vfggtt is aligned to
$AlignedGgtt ${NC} \n"
                            fi
                            echo $AlignedGgtt >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/ggtt_quota
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/ggtt_quota %10s = ${GREEN} $AlignedGgtt
${NC}\n"

                            echo $ae_gt_caterr >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/cat_error_count
                             printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/cat_error_count %10s = ${GREEN}
$ae_gt_caterr ${NC}\n"

                             echo $ae_gt_dbstorm >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/doorbell_time_us
                             printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/doorbell_time_us %10s
= ${GREEN} $ae_gt_dbstorm ${NC}\n"

                            echo $ae_gt_enginereset >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/engine_reset_count
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/engine_reset_count %10s
= ${GREEN} $ae_gt_enginereset ${NC}\n"

                             echo $ae_gt_h2gstorm >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/h2g_time_us
                             printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/h2g_time_us %10s = ${GREEN}
$ae_gt_h2gstorm ${NC}\n"
```

```
                            echo $ae_gt_irqstorm >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/irq_time_us
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/irq_time_us %10s = ${GREEN}
$ae_gt_irqstorm ${NC}\n"

                            echo $ae_gt_pagefault >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/page_fault_count
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/page_fault_count %10s
= ${GREEN} $ae_gt_pagefault ${NC}\n"
                        done # End of for tileNum in `seq 0 ((
$global_no_of_tiles -= 1 ))`;
                    elif [ $NVFS -eq 2 ]
                    then
                        vfNum=1
                        for (( tileNum=0;tileNum<$global_no_of_tiles;tileNum++ ))
                        do
                            printf
"${YELLOW}************************************************${NC}\n"
                            printf "* \t ${YELLOW} Resources alloted to
tile$tileNum\t ${NC} * \n"
                            printf
"${YELLOW}************************************************${NC}\n"
                            vfNum=$(($tileNum + 1))



                            echo $schedreset >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/engine_reset
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/engine_reset %10s = ${GREEN}
$schedreset ${NC}\n"

                            echo $pfdoorbells >
$cardPath$IOV_PF_PATH/gt$tileNum/doorbells_quota
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/doorbells_quota %10s =  ${GREEN} $pfdoorbells
${NC}\n"

                            echo $ae_sample_period >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/sample_period_m
                            s printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/sample_period_ms %10s = ${GREEN}
$ae_sample_period ${NC}\n"


                            printf
"************************************************\n"
                            printf "* \t ${GREEN} Resources alloted to VF$vfNum\t
${NC} * \n"
                            printf
"************************************************\n"

                            echo $vfdoorbells >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/doorbells_quota
```

```
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/doorbells_quota %10s =  ${GREEN}
$vfdoorbells ${NC}\n"

                                echo $vfcontexts >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/contexts_quot
                                a printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/contexts_quota %10s = ${GREEN}

$vfcontexts ${NC}\n"

                                vfLmemCalculated=$vflmem

                                #Align LMEM size
                                AlignedLmem=$(($vfLmemCalculated/ $roundPageSize_64k *
$roundPageSize_64k))
                                if [ $AlignedLmem -ne $vfLmemCalculated ]; then
                                    printf "${RED} LMEM size $vfLmemCalculated is
aligned to $AlignedLmem ${NC} \n"
                                fi
                                echo $AlignedLmem >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/lmem_quota
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/lmem_quota = ${GREEN}

$AlignedLmem ${NC}\n"

                                #Align the GGTT size
                                AlignedGgtt=$(($vfggtt / $roundPageSize_64k *
$roundPageSize_64k))
                                if [ $AlignedGgtt -ne $vfggtt ]; then
                                    printf "${RED} GGTT size $vfggtt is aligned to
$AlignedGgtt ${NC} \n"
                                fi
                                echo $AlignedGgtt >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/ggtt_quota
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/ggtt_quota %10s = ${GREEN} $AlignedGgtt

${NC}\n"

                                echo $ae_gt_caterr >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/cat_error_count
                                 printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/cat_error_count %10s = ${GREEN}

$ae_gt_caterr ${NC}\n"

                                 echo $ae_gt_dbstorm >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/doorbell_time_us
                                 printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/doorbell_time_us %10s

= ${GREEN} $ae_gt_dbstorm ${NC}\n"

                                echo $ae_gt_enginereset >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/engine_reset_count
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/engine_reset_count %10s

= ${GREEN} $ae_gt_enginereset ${NC}\n"
```

```
                            echo $ae_gt_h2gstorm >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/h2g_time_us
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/h2g_time_us %10s = ${GREEN}
$ae_gt_h2gstorm ${NC}\n"

                            echo $ae_gt_irqstorm >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/irq_time_us
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/irq_time_us %10s = ${GREEN}
$ae_gt_irqstorm ${NC}\n"

                            echo $ae_gt_pagefault >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/page_fault_count
                            printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/page_fault_count %10s
= ${GREEN} $ae_gt_pagefault ${NC}\n"
                        done
                    elif [ $NVFS -ge 3 ]




                    then
                        vfNum=1
                        resDivFactor=$(($n/$global_no_of_tiles))
                        for (( tileNum=0;tileNum<$global_no_of_tiles;tileNum++ ))
                        do
                            printf
"${YELLOW}************************************************${NC}\n"
                            printf "* \t ${YELLOW} Resources alloted to
tile$tileNum\t ${NC} * \n"
                            printf
"${YELLOW}************************************************${NC}\n"

                            echo $schedreset >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/engine_reset
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/engine_reset %10s = ${GREEN}
$schedreset ${NC}\n"

                            echo $pfdoorbells >
$cardPath$IOV_PF_PATH/gt$tileNum/doorbells_quota
                            printf
"$cardPath$IOV_PF_PATH/gt$tileNum/doorbells_quota %10s =  ${GREEN} $pfdoorbells
${NC}\n"

                            echo $ae_sample_period >
$cardPath$IOV_PF_PATH/gt$tileNum/policies/sample_period_m
                            s printf
"$cardPath$IOV_PF_PATH/gt$tileNum/policies/sample_period_ms %10s = ${GREEN}
$ae_sample_period ${NC}\n"

                            vfCountOnEachTile=1
                            for ((
vfCountOnEachTile=1;vfCountOnEachTile<=$resDivFactor; vfCountOnEachTile++ ));
                            do
```

```
                                printf
"************************************************\n"
                                printf "* \t ${GREEN} Resources alloted to
VF$vfNum\t ${NC} * \n"
                                printf
"************************************************\n"

                                echo $vfdoorbells >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/doorbells_quota
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/doorbells_quota %10s = ${GREEN}
$vfdoorbells ${NC}\n"
                                echo $vfcontexts >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/contexts_quota
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/contexts_quota %10s = ${GREEN}
$vfcontexts ${NC}\n"
                                vfLmemCalculated=$vflmem
                                #Align LMEM size
                                AlignedLmem=$(($vfLmemCalculated/
$roundPageSize_64k * $roundPageSize_64k))
                                if [ $AlignedLmem -ne $vfLmemCalculated ]; then
                                    printf "${RED} LMEM size $vfLmemCalculated is
aligned to $AlignedLmem ${NC} \n"
                                fi
                                echo $AlignedLmem >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/lmem_quota
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/lmem_quota = ${GREEN}
$AlignedLmem ${NC}\n"
                                #Align the GGTT size
                                AlignedGgtt=$(($vfggtt / $roundPageSize_64k *
$roundPageSize_64k))
                                if [ $AlignedGgtt -ne $vfggtt ]; then
                                    printf "${RED} GGTT size $vfggtt is aligned to
$AlignedGgtt ${NC} \n"
                                fi
                                echo $AlignedGgtt >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/ggtt_quota
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/ggtt_quota %10s = ${GREEN} $AlignedGgtt
${NC}\n"
                                echo $ae_gt_caterr >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/cat_error_count
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/cat_error_count %10s = ${GREEN}
$ae_gt_caterr ${NC}\n"
                                echo $ae_gt_dbstorm >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/doorbell_time_us
                                printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/doorbell_time_us %10s
= ${GREEN} $ae_gt_dbstorm ${NC}\n"
```

```
                                    echo $ae_gt_enginereset >
 $cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/engine_reset_count
                                    printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/engine_reset_count %10s
= ${GREEN} $ae_gt_enginereset ${NC}\n"


                                    echo $ae_gt_h2gstorm >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/h2g_time_us
                                    printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/h2g_time_us %10s = ${GREEN}
$ae_gt_h2gstorm ${NC}\n"
                                    echo $ae_gt_irqstorm >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/irq_time_us
                                    printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/irq_time_us %10s = ${GREEN}
$ae_gt_irqstorm ${NC}\n"
                                    echo $ae_gt_pagefault >
$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/page_fault_count
                                    printf
"$cardPath$IOV_VF_PATH$vfNum/gt$tileNum/threshold/page_fault_count %10s
= ${GREEN} $ae_gt_pagefault ${NC}\n"
                                     (( vfNum += 1 ))
                                done
                            done
                            #Error case
                        else
                            echo "Error in received product id\n"
                        fi
                    else
                        echo "Error in identifying product"
                    fi # End of if [[ "$productId" = "ATSM_75" ]] || [[ "$productId" =
"ATSM_150" ]]
                fi #End of if (( $n==$numOfVfsFromInVgpuProfileId )) ;
            then fi #End of if [ $TYPE = $profileType ] ; then
        done < $INPUT #End of while read profid schedreset provmode pflmem
pfcontexts pfdoorbells pfggtt vflmem vfcontexts

        function_debugPrints "Enabling VFs \n"
        echo 0 > $cardPath$IOV_PF_PATH"/device/sriov_drivers_autoprobe"
        echo $numOfVfsFromInVgpuProfileId  >
$cardPath$IOV_PF_PATH"/device/sriov_numvfs"

        function_printFuncName ${FUNCNAME[0]} 1

        IFS=$OLDIFS
}
```

```
$ chmod +x *.sh
```