NVIDIA BlueField Modes of Operation

Table of contents

This document describes the modes of operation available for NVIDIA® BlueField® networking platforms (DPUs or SuperNICs).

Introduction

The NVIDIA® BlueField® networking platform (DPU or SuperNIC) has several modes of operation:

- DPU mode, or embedded function (ECPF) ownership, where the embedded Arm system controls the NIC resources and data path
- Zero-trust mode which is an extension of the ECPF ownership with additional restrictions on the host side
- NIC mode where BlueField behaves exactly like an adapter card from the perspective of the external host

> **Note**
>
> The default mode of operation for the BlueField DPU is DPU mode
>
> The default mode of operation for the BlueField SuperNIC is NIC mode

DPU Mode

This mode, known also as embedded CPU function ownership (ECPF) mode, is the default mode for the BlueField DPU.

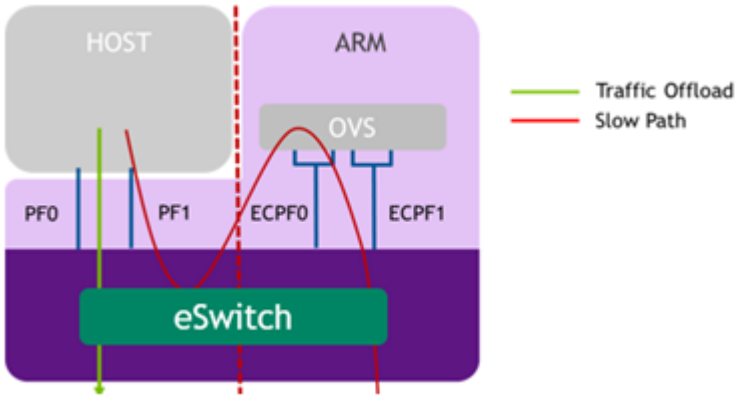
In DPU mode, the NIC resources and functionality are owned and controlled by the embedded Arm subsystem. All network communication to the host flows through a virtual switch control plane hosted on the Arm cores, and only then proceeds to the host. While working in this mode, the BlueField is the trusted function managed by the data center and host administrator—to load network drivers, reset an interface, bring an interface up and down, update the firmware, and change the mode of operation on BlueField.

A network function is still exposed to the host, but it has limited privileges. In particular:

1. The driver on the host side can only be loaded after the driver on the BlueField has loaded and completed NIC configuration.

2. All ICM (Interface Configuration Memory) is allocated by the ECPF and resides in the BlueField's memory.

3. The ECPF controls and configures the NIC embedded switch which means that traffic to and from the host (BlueField) interface always lands on the Arm side.



When the server and BlueField are initiated, the networking to the host is blocked until the virtual switch on the BlueField is loaded. Once it is loaded, traffic to the host is allowed by default.

There are two ways to pass traffic to the host interface: Either using representors to forward traffic to the host (every packet to/from the host would be handled also by the network interface on the embedded Arm side) or push rules to the embedded switch which allows and offloads this traffic.

In DPU mode, OpenSM must be run from the BlueField side (not the host side). Also, management tools (e.g., `sminfo`, `ibdev2netdev`, `ibnetdiscover`) can only be run from the BlueField side (not from the host side).

Zero-trust Mode

Zero-trust mode is a specialization of DPU mode which implements an additional layer of security where the host system administrator is prevented from accessing BlueField from the host. Once zero-trust mode is enabled, the data center administrator should control BlueField entirely through the Arm cores and/or BMC connection instead of through the host.

For security and isolation purposes, it is possible to restrict the host from performing operations that can compromise the BlueField. The following operations can be restricted individually when changing the BlueField host to zero-trust mode:

- Port ownership – the host cannot assign itself as port owner
- Hardware counters – t he host does not have access to hardware counters
- Tracer functionality is blocked
- RShim interface is blocked
- Firmware flash is restricted

Enabling Zero-trust Mode

To enable host restriction:

1. Start the MST service.
2. Set zero-trust mode. From the Arm side, run:

```
$ sudo mlxprivhost -d /dev/mst/<device> r --disable_rshim --disable_tracer --disable_counter_rd --disable_port_owner
```

   - If no `--disable_*` flags are used, users must perform BlueField system reboot .
   - If any `--disable_*` flags are used, users must perform BlueField system-level reset.

Disabling Zero-trust Mode

To disable host restriction:

1. Set the mode to privileged. Run:

```
$ sudo mlxprivhost -d /dev/mst/<device> p
```

2. Applying configuration:

    - I f host restriction had not been applied using any `--disable_*` flags, users must perform BlueField system reboot .
    - If host restriction had been applied using any `--disable_*` flags, users must perform BlueField system-level reset.

NIC Mode

In this mode, BlueField behaves exactly like an adapter card from the perspective of the external host.

> **Note**
>
> The following instructions presume BlueField to be operating in DPU mode. If BlueField is operating in zero-trust mode, please return to DPU mode before proceeding.

> **Note**
>
> The following notes are relevant for updating the BFB bundle in NIC mode:
>
> - During BFB Bundle installation, Linux is expected to boot to upgrade NIC firmware and BMC software
> - During the BFB Bundle installation, it is expected for the mlx5 driver to error messages on the x86 host. These prints may be ignored as they are resolved by a mandatory, post-installation power cycle.
> - It is mandatory to power cycle the host after the installation is complete for the changes to take effect

- As Linux is booting during BFB Bundle installation, it is expected for the mlx5 core driver to timeout on the BlueField Arm

NIC Mode for BlueField-3

Note

When BlueField-3 is configured to operate in NIC mode, Arm OS will not boot.

NIC mode for BlueField-3 saves power, improves device performance, and improves the host memory footprint.

Configuring NIC Mode on BlueField-3 from Linux

Enabling NIC Mode on BlueField-3 from Linux

Before moving to NIC mode, make sure you are operating in DPU mode by running:

```
host/bf> sudo mlxconfig -d /dev/mst/mt41692_pciconf0 -e q
```

The output should have `INTERNAL_CPU_MODEL= EMBBEDDED_CPU(1)` and `EXP_ROM_UEFI_ARM_ENABLE = True (1)` (default).

To enable NIC mode from DPU mode:

1. Run the following on the host or Arm:

```
host/bf> sudo mlxconfig -d /dev/mst/mt41692_pciconf0 s
INTERNAL_CPU_OFFLOAD_ENGINE=1
```

2. Perform BlueField system-level reset for the `mlxconfig` settings to take effect.

Disabling NIC Mode on BlueField-3 from Linux

To return to DPU mode from NIC mode:

1. Run the following on the host:

```
host> sudo mlxconfig -d /dev/mst/mt41692_pciconf0 s
INTERNAL_CPU_OFFLOAD_ENGINE=0
```

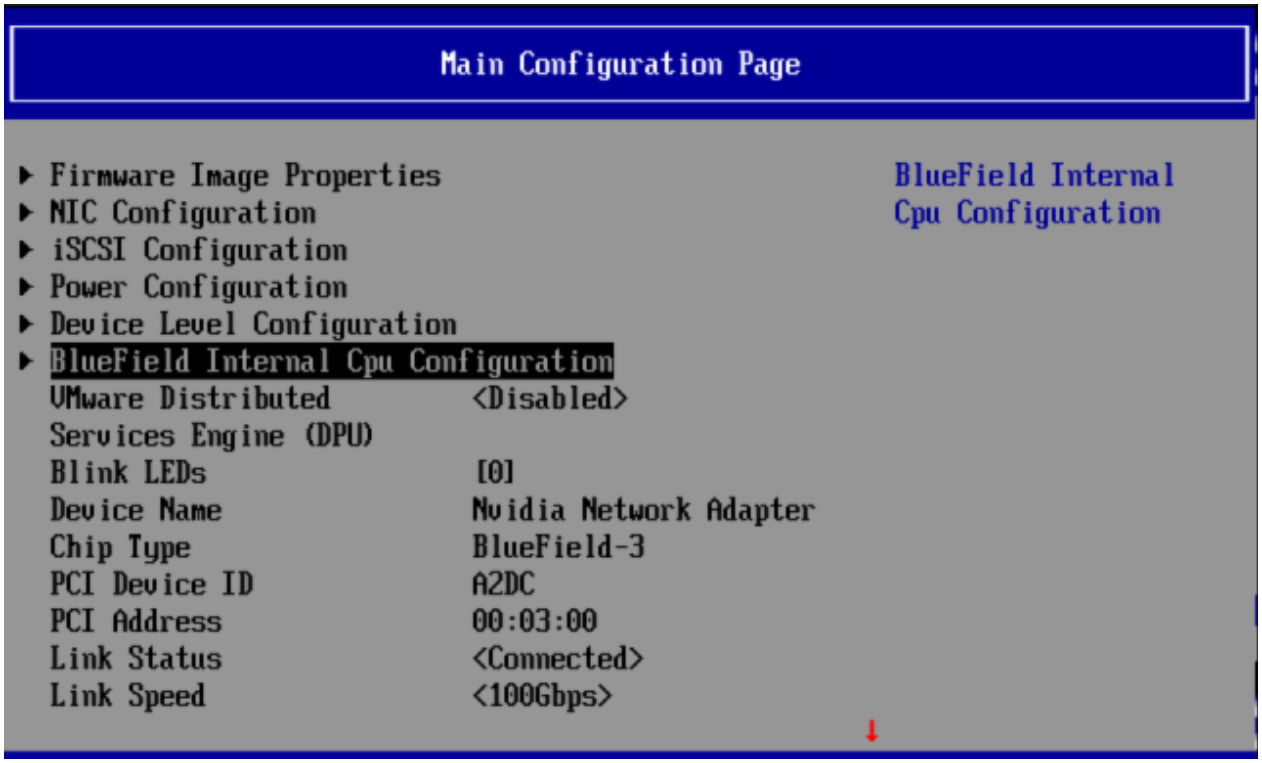
2. Perform BlueField system-level reset for the `mlxconfig` settings to take effect.

Configuring NIC Mode on BlueField-3 from Host BIOS HII UEFI Menu

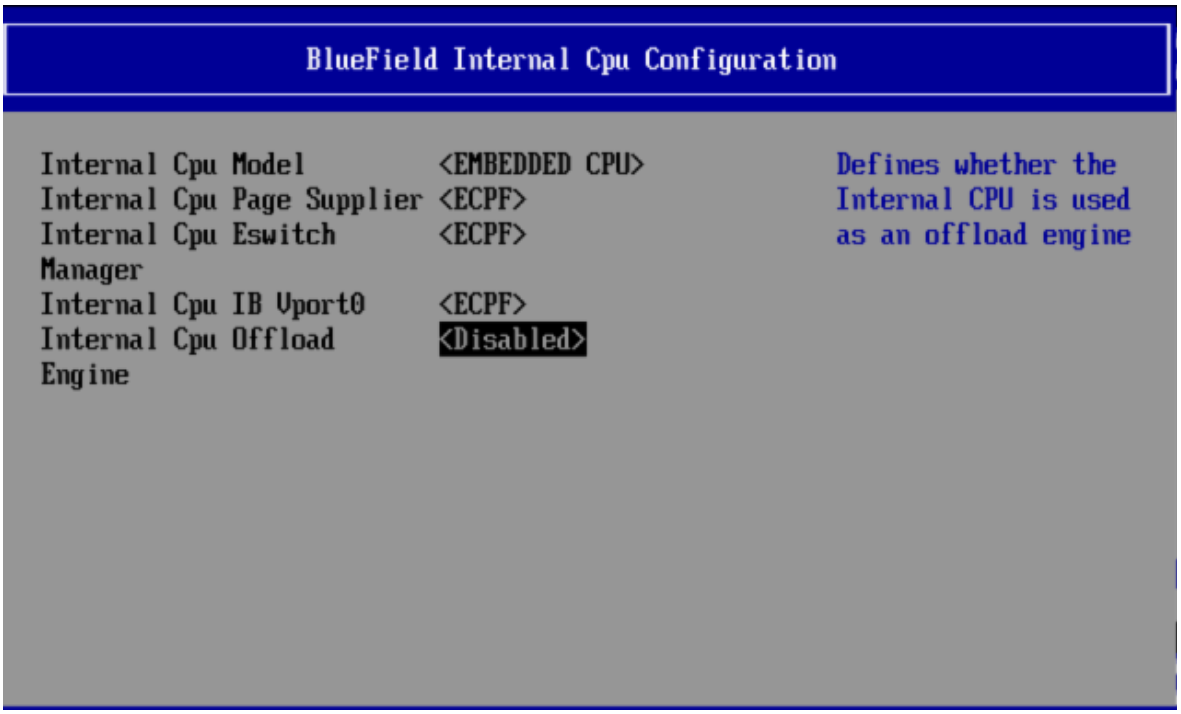
> **Info**
>
> The screenshots in this section are examples only and may vary depending on the vendor of your specific host.

1. Select the network device that presents the uplink (i.e., select the device with the uplink MAC address).

2. Select "BlueField Internal Cpu Configuration".

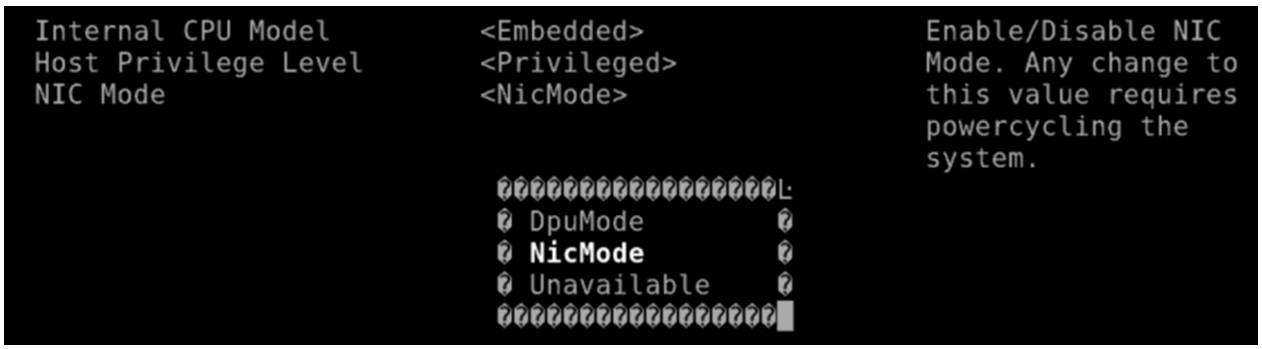


- To enable NIC mode, set "Internal Cpu Offload Engine" to "Disabled".
- To switch back to DPU mode, set "Internal Cpu Offload Engine" to "Enabled".



Configuring NIC Mode on BlueField-3 from Arm UEFI

1. Access the Arm UEFI menu by pressing the Esc button twice.
2. Select "Device Manager".
3. Select "System Configuration".
4. Select "BlueField Modes".
5. Set the "NIC Mode" field to `NicMode` to enable NIC mode.



> **Info**
>
> Configuring `Unavailable` is inapplicable.

6. Exit "BlueField Modes" and "System Configuration" and make sure to save the settings. Exit the UEFI setup using the 'reset' option. The configuration is not yet applied and BlueField is expected to boot regularly, still in DPU mode.
7. perform BlueField system-level reset to change to NIC mode.

Configuring NIC Mode on BlueField-3 Using Redfish

Run the following from the BlueField BMC:

1. Get the current BIOS attributes:

```
curl -k -u root:'password' -H 'content-type: application/json' -X GET
https://bmc_ip/redfish/v1/Systems/Bluefield/Oem/Nvidia
```

Output example (truncated):

```
{
  "Actions": {
    "#HostRshim.Set": {
      "Parameters": [
        {
          "AllowableValues": [
            "Disabled",
            "Enabled"
          ],
..
..
..

  "Mode": "DpuMode",
  "Truststore": {
    "Certificates": {
      "@odata.id": "/redfish/v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates"
    }
  }
}
```

2. Change BlueField mode from `DpuMode` to `NicMode`:

```
curl -k -u root:'password' -H 'content-type: application/json' -d '{"Mode":
"NicMode"}' -X POST
```

```
https://bmc_ip/redfish/v1/Systems/Bluefield/Oem/Nvidia/Actions
```

> **Info**
>
> To reverse this configuration, run the same command and replace `NicMode` with `DpuMode`.

Output example:


```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.16.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```


3. Perform BlueField-3 reset:

```
curl -k -u root:'password' -H 'content-type: application/json' -X POST
https://bmc_ip/redfish/v1/Systems/Bluefield/Actions/ComputerSy
-d '{"ResetType": "ForceRestart"}'
```

Output example:

```
{
  "@Message.ExtendedInfo": [
    {
      "@odata.type": "#Message.v1_1_1.Message",
      "Message": "The request completed successfully.",
      "MessageArgs": [],
      "MessageId": "Base.1.16.0.Success",
      "MessageSeverity": "OK",
      "Resolution": "None"
    }
  ]
}
```

4. Power cycle the host.
5. Once the host is up, verify that the mode changed:

```
curl -k -u root:'password' -H 'content-type: application/json' -X GET
https://bmc_ip/redfish/v1/Systems/Bluefield/Oem/Nvidia
```

Output example (truncated):

```
{
  "Actions": {
    "#HostRshim.Set": {
      "Parameters": [
        {
          "AllowableValues": [
            "Disabled",
            "Enabled"
          ],
..
```

```
. .
. .

},
 "HostRshim" :  "Enabled",
 "Mode" :  "NicMode",
 "Truststore" :  {
   "Certificates" :  {
     "@odata.id" :  "/redfish/v1/Systems/Bluefield/Oem/Nvidia/Truststore/Certificates"
   }
 }
}
```

> **Note**
>
> If moving from NIC mode to DPU mode, run the command `mlxfwreset -d /dev/mst/mt41692_pciconf0 -l 4 r` on the host to reboot it.

Updating Firmware Components in BlueField-3 NIC Mode

Once in NIC mode, updating ATF and UFEI can be done using the standard `*.bfb` image:

```
# bfb-install --bfb <BlueField-BSP>.bfb --rshim rshim0
```

NIC Mode for BlueField-2

In this mode, the ECPFs on the Arm side are not functional but the user is still able to access the Arm system and update `mlxconfig` options.

> **Note**
>
> When NIC mode is enabled, the drivers and services on the Arm are no longer functional.

Configuring NIC Mode on BlueField-2 from Linux

Enabling NIC Mode on BlueField-2 from Linux

To enable NIC mode from DPU mode:

1. Run the following from the x86 host side:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> s \
INTERNAL_CPU_PAGE_SUPPLIER=1 \
INTERNAL_CPU_ESWITCH_MANAGER=1 \
INTERNAL_CPU_IB_VPORT0=1 \
INTERNAL_CPU_OFFLOAD_ENGINE=1
```

> **Note**
>
> To restrict RShim PF (optional), make sure to configure `INTERNAL_CPU_RSHIM=1` as part of the `mlxconfig` command.

2. Perform BlueField system-level reset t o load the new configuration .

Info

Refer to the troubleshooting section of the guide for a step-by-step procedure.

Note

Multi-host is not supported when BlueField is operating in NIC mode.

Note

To obtain firmware BINs for BlueField-2 devices, please refer to the BlueField-2 firmware download page.

Disabling NIC Mode on BlueField-2 from Linux

To change from NIC mode back to DPU mode:

1. Install and start the RShim driver on the host.
2. Disable NIC mode. Run:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> s \
INTERNAL_CPU_PAGE_SUPPLIER=0 \
INTERNAL_CPU_ESWITCH_MANAGER=0 \
INTERNAL_CPU_IB_VPORT0=0 \
```

```
INTERNAL_CPU_OFFLOAD_ENGINE=0
```

> **Note**
>
> If `INTERNAL_CPU_RSHIM=1`, then make sure to configure `INTERNAL_CPU_RSHIM=0` as part of the `mlxconfig` command.

3. Perform a BlueField system reboot for the `mlxconfig` settings to take effect.

Configuring NIC Mode on BlueField-2 from Arm UEFI

Follow the same instructions in section "Configuring NIC Mode on BlueField-3 from Arm UEFI".

Configuring NIC Mode on BlueField-2 Using Redfish

Follow the same instructions in section "Configuring NIC Mode on BlueField-3 Using Redfish".

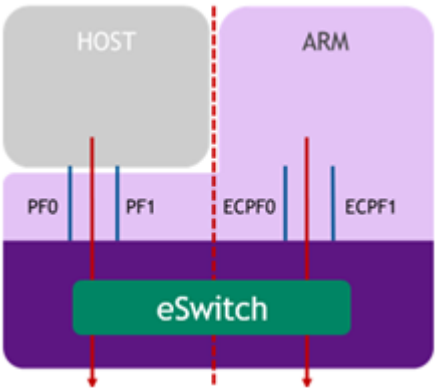
Separated Host Mode (Obsolete)

> **Warning**
>
> This BlueField mode of operation is obsolete. Please do not use it!

In separated host mode, a network function is assigned to both the Arm cores and the host cores. The ports/functions are symmetric in the sense that traffic is sent to both physical functions simultaneously. Each one of those functions has its own MAC address, which allows one to communicate with the other, and can send and receive Ethernet and RDMA over Converged Ethernet (RoCE) traffic. There is an equal bandwidth share between the two functions.

There is no dependency between the two functions. They can operate simultaneously or separately. The host can communicate with the embedded function as two separate hosts, each with its own MAC and IP addresses (configured as a standard interface).

In separated host mode, the host administrator is a trusted actor who can perform all configuration and management actions related to either network function.



This mode enables the same operational model of a SmartNIC (that does not have a separated control plane). In this case, the Arm control plane can be used for different functions but does not have any control on the host steering functions.

The limitations of this mode are as follows:

- Switchdev (virtual switch offload) mode is not supported on either of the functions
- SR-IOV is only supported on the host side

To configure separated host mode from DPU mode:

1. Enable separated host mode. Run:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> s INTERNAL_CPU_MODEL=0
```

2. Power cycle.

3. Verify configuration. Run:

```
$ mst start
$ mlxconfig -d /dev/mst/<device> q | grep -i model
```

4. Remove OVS bridges configuration from the Arm-side. Run:

```
$ ovs-vsctl list-br | xargs -r -l ovs-vsctl del-br
```

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF

ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.