NVIDIA DOCA Firefly Service

Guide

MLNX-15-060531 _v1.4 | November 2022

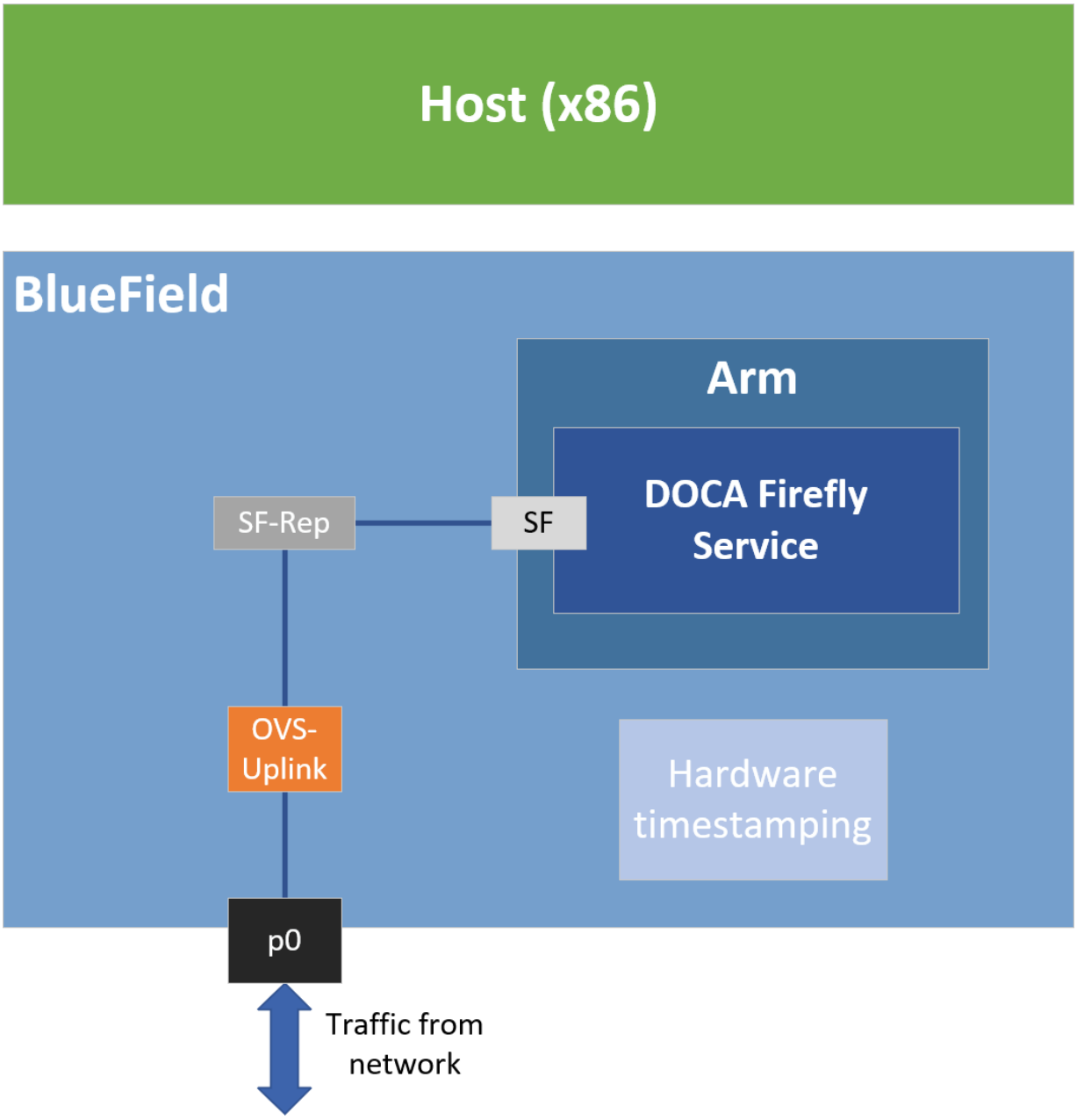
Table of Contents

Chapter 1. Introduction 1
Chapter 2. Requirements 3
2.1. Firmware Version 3
2.2. BlueField BSP Version 3
2.3. Embedded Mode 3
2.3.1. Configuring Firmware Settings on DPU for Embedded Mode 3
2.3.2. Helper Scripts 4
2.3.2.1. set_new_sf.sh 4
2.3.2.2. prepare_for_embedded_mode.sh 4
2.3.3. Setting Up Network Interfaces for Embedded Mode 5
2.4. Separated Mode 6
2.4.1. Configuring Firmware Settings on DPU for Separated Mode 6
2.4.2. Setting Up Network Interfaces for Separated Mode 6
Chapter 3. Service Deployment 7
Chapter 4. Configuration 8
4.1. Default Config File 8
4.2. Custom Config File 8
4.3. Overriding Specific Config File Parameters 8
4.4. PTP Monitoring Configuration 9
Chapter 5. Description 10
5.1. Providers 10
5.2. Profiles 10
5.3. Outputs 10
5.3.1. Container Output 10
5.3.2. Ptp4l Output 11
5.4. PTP Monitoring 12
Chapter 6. Troubleshooting 13
Chapter 7. PTP Profile Default Config Files 14
7.1. Media Profile 14
7.2. Default Profile 14

Chapter 1. Introduction

DOCA Firefly Service provides precision time protocol (PTP) based time syncing services to the BlueField DPU.

PTP is a protocol used to synchronize clocks in a network. When used in conjunction with hardware support, PTP is capable of sub-microsecond accuracy, which is far better than is what is normally obtainable with network time protocol (NTP). PTP support is divided between the kernel and user space. The ptp4l program implements the PTP boundary clock and ordinary clock. With hardware time stamping, it is used to synchronize the PTP hardware clock to the master clock.


Host (x86)
BlueField
Arm
DOCA Firefly Service
SF-Rep
SF
OVS-Uplink
Hardware timestamping
p0
Traffic from network

Chapter 2. Requirements

Some of the features provided by Firefly require specific hardware BlueField DPU capabilities:

- PPS – requires special BlueField DPUs with PPS capabilities
- SyncE – requires special BlueField DPUs with SyncE capabilities
- PTP – supported by all BlueField DPUs

Note that failure to run PPS or SyncE due to missing hardware support is noted in the container output, but the container will continue to run the timing services it can provide on the provided hardware, such as PTP.

2.1. Firmware Version

Your firmware version must be 24.33.1048 or higher.

2.2. BlueField BSP Version

The supported BlueField image versions are 3.9.0 and higher.

2.3. Embedded Mode

2.3.1. Configuring Firmware Settings on DPU for Embedded Mode

1. Set the DPU to embedded mode (default mode):
   ```
   sudo mlxconfig -y -d 03:00.0 s INTERNAL_CPU_MODEL=1
   ```
2. Enable the real time clock (RTC):
   ```
   sudo mlxconfig -d 03:00.0 set REAL_TIME_CLOCK_ENABLE=1
   ```
3. Power cycle the DPU to apply the configuration.
4. You may check the DPU mode using the following command:
   ```
   sudo mlxconfig -d 03:00.0 q | grep INTERNAL_CPU_MODEL
   # Example output
            INTERNAL_CPU_MODEL                  EMBEDDED_HOST(1)
   ```

2.3.2. Helper Scripts

Firefly's deployment contains a script to help with the configuration steps required for the network interface in embedded mode:

- `scripts/doca_firefly/set_new_sf.sh`
- `scripts/doca_firefly/prepare_for_embedded_mode.sh`

Both scripts are included as part of DOCA's container resource which can be downloaded according to the instructions in the NVIDIA DOCA Container Deployment Guide.

2.3.2.1. set_new_sf.sh

Creates a new trusted SF and marks it as "trusted".

Script arguments:

- PCIe address
- SF number (checks if already exists)
- MAC address (if absent, a random address is generated)

Examples:

- Create SF with number "4" over port 0 of the DPU:
  ```
  ./set_new_sf.sh 0000:03:00.0 4
  ```
- Create SF with number "5" over port 0 of the DPU and a specific MAC address:
  ```
  ./set_new_sf.sh 0000:03:00.0 5 aa:bb:cc:dd:ee:ff
  ```
- Create SF with number "4" over port 1 of the DPU:
  ```
  ./set_new_sf.sh 0000:03:00.1 4
  ```

The first two examples should work out of the box for a BlueField-2 device and create SF4 and SF5 respectively.

2.3.2.2. prepare_for_embedded_mode.sh

This script automates all the steps mentioned in section Setting Up Network Interfaces for Embedded Mode and configures a freshly installed BFB image to the settings required by DOCA Firefly.

Notes:

- The script deletes all previous OVS settings and creates a single OVS bridge that matches the definitions below
- The script should only be run once when connecting to the DPU for the first time or after a power cycle
- The only manual step required after using this script is configuring the IP address for the created network interface (step 5 in section Setting Up Network Interfaces for Embedded Mode)

Script arguments:

- SF number (checks if already exists)

Examples:

- Prepare OVS settings using an SF indexed 4:

```
./prepare_for_embedded_mode.sh 4
```

2.3.3. Setting Up Network Interfaces for Embedded Mode

1. Create a trusted SF to be used by the service according to the Scalable Function Setup Guide.

   Note: The following instructions assume that the SF has been created using index 4.

2. Create the required OVS setting as is shown in the architecture diagram:

```
$ sudo ovs-vsctl add-br uplink
$ sudo ovs-vsctl add-port uplink p0
$ sudo ovs-vsctl add-port uplink en3f0pf0sf4
```

   Note: If traffic from the host is required as well, make sure to add the following port to the OVS bridge:

```
$ sudo ovs-vsctl add-port uplink pf0hpf
```

3. Verify the OVS settings:

```
sudo ovs-vsctl show
    Bridge uplink
        Port uplink
            Interface uplink
                type: internal

        Port en3f0pf0sf4
            Interface en3f0pf0sf4

        Port p0
            Interface p0
```

4. Enable TX timestamping on the SF interface (not the representor):

```
# tx port timestamp offloading
sudo ethtool --set-priv-flags enp3s0f0s4 tx_port_ts on
```

5. Enable the interface and set an IP address for it:

```
# configure ip for the interface:
sudo ifconfig enp3s0f0s4 <ip addr> up
```

6. Configure OVS to support TX timestamping over this SF:

```
$ sudo ovs-vsctl set Bridge uplink mcast_snooping_enable=true
$ sudo ovs-vsctl set Port en3f0pf0sf4 other_config:mcast-snooping-flood=true
$ sudo ovs-vsctl set Port en3f0pf0sf4 other_config:mcast-snooping-flood-
reports=true
$ sudo tc filter add dev en3f0pf0sf4 protocol ipv4 parent ffff: prio 1000 flower
 ip_proto udp dst_port 319 skip_sw action mirred egress redirect dev p0
$ sudo tc filter add dev p0 protocol ipv4 parent ffff: prio 1000 flower ip_proto
 udp dst_port 319 skip_sw action mirred egress redirect dev en3f0pf0sf4
$ sudo tc filter add dev en3f0pf0sf4 protocol ipv4 parent ffff: prio 1000 flower
 ip_proto udp dst_port 320 skip_sw action mirred egress redirect dev p0
```

```
$ sudo tc filter add dev p0 protocol ipv4 parent ffff: prio 1000 flower ip_proto
 udp dst_port 320 skip_sw action mirred egress redirect dev en3f0pf0sf4
```

Note: If your OVS bridge uses a name other than `uplink`, make sure that the used name is reflected in the `ovs-vsctl set Bridge` command:

```
$ sudo ovs-vsctl set Bridge <bridge-name> mcast_snooping_enable=true
```

2.4. Separated Mode

2.4.1. Configuring Firmware Settings on DPU for Separated Mode

1. Set the DPU mode to "Separated":

   ```
   sudo mlxconfig -y -d 03:00.0 s INTERNAL_CPU_MODEL=0
   ```

2. Enable RTC:

   ```
   sudo mlxconfig -d 03:00.0 set REAL_TIME_CLOCK_ENABLE=1
   ```

3. Power cycle the DPU to apply the configuration.

4. You may check the DPU mode using the following command:

   ```
   sudo mlxconfig -d 03:00.0 q | grep INTERNAL_CPU_MODEL
   # Example output
           INTERNAL_CPU_MODEL                  SEPARATED_HOST(0)
   ```

2.4.2. Setting Up Network Interfaces for Separated Mode

1. Make sure that that `p0` is not connected to an OVS bridge:

   ```
   sudo ovs-vsctl show
   ```

2. Enable TX timestamping on the `p0` interface:

   ```
   # tx port timestamp offloading (assuming PTP interface is p0)
   sudo ethtool --set-priv-flags p0 tx_port_ts on
   ```

3. Enable the interface and set an IP address for it:

   ```
   # configure ip for the interface
   sudo ifconfig p0 <ip-addr> up
   ```

Chapter 3. Service Deployment

For more information about the deployment of DOCA containers on top of the BlueField DPU, refer to NVIDIA DOCA Container Deployment Guide.

DOCA Firefly service is available on NGC, NVIDIA's container catalog. Service-specific configuration steps and deployment instructions can be found under the service's container page.

Chapter 4. Configuration

The PTP program, `linuxptp`, has a configuration file that enables us to customize various PTP-related settings.

4.1. Default Config File

Each profile has its own base PTP configuration file. For example, the Media profile PTP configuration file is `ptp4l-media.conf`.

The default configuration files can be found in section PTP Profile Default Config Files.

4.2. Custom Config File

Instead of using a profile's base config file, users can create a file of their own.

To set a custom config file, users should locate their config file in the directory `/etc/firefly` and set the config file name in DOCA Firefly's YAML file.

For example, to set a custom `linuxptp` config file, the user can set the parameter `PTP_CONFIG_FILE` in the YAML file:

```
- name: PTP_CONFIG_FILE
  value: my_custom_ptp.conf
```

In this example, `my_custom_ptp.conf` should be placed at `/etc/firefly/my_custom_ptp.conf`.

4.3. Overriding Specific Config File Parameters

Instead of replacing the entire config file, users may opt to override specific parameters. This can be done using the following variable syntax in the YAML file: `CONF_<TYPE>_<SECTION>_<PARAMETER_NAME>`.

- `TYPE` - currently only PTP is supported

- `SECTION` - the section in the config file that the parameter should be placed in

> **Note:** If the specified section does not already exist in the config file, a new section is created.

- `PARAMETER_NAME` - the config parameter name as should be placed in the config file

> **Note:** If the parameter name already exists in the config file, then the value is changed according to the value provided in the `.yaml` file. If the parameter name does not already exist in the config file, then it is added.

For example, the following variable in the YAML file definition changes the value of the parameter `priority1` under section `global` in the PTP config file to `64`.

```
- name: CONF_PTP_global_priority1
  value: "64"
```

4.4. PTP Monitoring Configuration

DOCA Firefly contains an alpha feature for monitoring the PTP state during Firefly's execution. This feature could be activated using the following YAML lines:

```
- name: PTP_MONITOR
  value: "active"
```

Chapter 5. Description

5.1. Providers

DOCA Firefly Service uses the following third-party providers to provide time syncing services:

- Linuxptp – PTP service, provided by the PTP4L program
- Testptp – PPS settings service

5.2. Profiles

DOCA Firefly Service includes profiles which represent common use cases for the Firefly service that provide a different default configuration per profile:

Profiles	Media	Default
Purpose	Media productions	Any user requiring PTP
Content	PTP	PTP
PTP profile	SMPTE 2059-2	PTP default profile
PPS in	Enabled	Enabled
PPS out	Enabled	Enabled
PTP client/server*	Client only	Both

Note: Client only is only relevant to a single PTP interface. If more than one PTP interface is provided in the YAML file, both modes are enabled.

5.3. Outputs

5.3.1. Container Output

The output of the DOCA Firefly Service container can be viewed using the following command:

```
crictl logs <CONTAINER-ID>
```

Where `CONTANIER-ID` can be retrieved using the following command:

```
sudo crictl ps
```

For example, in the following output, the container ID is `8f368b98d025b`.

```
$ sudo crictl ps
CONTAINER           IMAGE               CREATED             STATE               NAME
                ATTEMPT             POD ID              POD
8f368b98d025b       289809f312b4c       2 seconds ago       Running
 doca-firefly        0                   5af59511b4be4       doca-firefly-some-
computer-name
```

The output of the container depends on the services supported by the hardware and enabled via configuration and the profile selected. However, note that any of the configurations runs PTP, so when DOCA FireFly is running successfully expect to see the line `Running ptp4l`.

The following is an example of the expected container output when running the default profile on a DPU that supports PPS:

```
set pin function okay
PPS in set
set pin function okay
PPS out set
name mlx5_pps0 index 0 func 1 chan 0
name mlx5_pps1 index 1 func 2 chan 0
periodic output request okay
Running ptp4l
```

The following is an example of the expected container output when running the default profile on a DPU that does not support PPS:

```
PPS capability not found, seems that card doesn't support
capabilities:
  100000000 maximum frequency adjustment (ppb)
  0 programmable alarms
  0 external time stamp channels
  0 programmable periodic signals
  0 pulse per second
  0 programmable pins
  0 cross timestamping
Running ptp4l
```

5.3.2. Ptp4l Output

The ptp4l output can be found in the file `/var/log/doca/firefly/ptp4l.log`.

Example output:

```
ptp4l[95877.202]: selected /dev/ptp2 as PTP clock
ptp4l[95877.203]: port 1 (enp3s0f0s0): INITIALIZING to LISTENING on INIT_COMPLETE
ptp4l[95877.204]: port 0 (/var/run/ptp4l): INITIALIZING to LISTENING on
 INIT_COMPLETE
ptp4l[95877.204]: port 0 (/var/run/ptp4lro): INITIALIZING to LISTENING on
 INIT_COMPLETE
ptp4l[95884.191]: port 1 (enp3s0f0s0): LISTENING to MASTER on
 ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES
ptp4l[95884.191]: selected local clock 021898.fffe.daee36 as best master
ptp4l[95884.191]: port 1 (enp3s0f0s0): assuming the grand master role
PPS capability not found, seems that card doesn't support
```

5.4. PTP Monitoring

Note: Supported at alpha level only.

PTP monitoring periodically queries for various PTP-related information and prints it to the container's log.

The following is a sample output of this tool:

```
gmIdentity:                 d26434.fffe.8f8345
master_offset:              -8
gmPresent:                  true
ptp_stable:                 Recovered
UtcOffset:                  37
timeTraceable:              0
frequencyTraceable:         0
grandmasterPriority1:       128
gmClockClass:               248
gmClockAccuracy:            0xfe
grandmasterPriority2:       127
gmOffsetScaledLogVariance: 0xffff
ptp_time:                   Tue Oct 18 08:45:23 2022
system_time:                Tue Oct 18 08:45:23 2022
error_count:                1
last_err_time:              Tue Oct 18 08:41:24 2022
```

Among others, this monitoring provides the following information:

- Details about the Grandmaster the DPU is syncing to
- Current PTP timestamp
- Health information such as connection errors during execution and whether they have been recovered from

PTP monitoring is disabled by default and can be activated by uncommenting the relevant lines as shown in the configuration section above.

Once activated, the information can viewed from the container using the following command:

```
sudo crictl logs --tail=18 <CONTAINER-ID>
```

It is recommended to use the following watch command to actively monitor the PTP state:

```
sudo watch -n 1 crictl logs --tail=18 <CONTAINER-ID>
```

Chapter 6. Troubleshooting

For general troubleshooting purposes, refer to NVIDIA DOCA Troubleshooting Guide.

For container-related troubleshooting, refer to the "Troubleshooting" section in the NVIDIA DOCA Container Deployment Guide.

The following are additional troubleshooting tips for DOCA Firefly Service:

- If no pod is created, verify that your YAML is written correctly (see NVIDIA DOCA Troubleshooting Guide) and check the output of the following command:
  ```
  sudo journalctl -u kubelet
  ```
- If the pod's STATE fails to be marked as `Ready` (check using `circtl pods`), check if the container has run and exited:
  1. Check the container's state:
     ```
     sudo crictl ps -a
     ```
  2. If the container did exit, use the container's ID to check the log output by running:
     ```
     sudo crictl logs <CONTAINER-ID>
     ```
- If the error `custom config file not found` appears in the container log, check the custom file name written in the YAML file and make sure that you properly placed the file with that name under the `/etc/firefly/` directory.
- If the error `profile <name> not supported. Aborting` appears in the container log, verify that the profile you selected in the YAML file matches one of the optional profiles as listed in the profiles table.
- If the message `PPS capability not found, seems that card doesn't support` appears in the container log, then the DPU hardware does not support PPS. However, PTP can still run on this HW and you should see the line "`Running ptp4l`" in the container log which indicatest that PTP is running successfully.

Chapter 7. PTP Profile Default Config Files

7.1. Media Profile

```
[global]
domainNumber                    127
priority1                       128
priority2                       127
use_syslog                      1
logging_level                   6
tx_timestamp_timeout            30
hybrid_e2e                      1
dscp_event                      46
dscp_general                    46
logAnnounceInterval             -2
announceReceiptTimeout          3
logSyncInterval                 -3
logMinDelayReqInterval          -3
delay_mechanism                 E2E
network_transport UDPv          4
boundary_clock_jbod             1
```

7.2. Default Profile

```
# This config file is based on linuxptp config file default.cfg[global]
#
# Default Data Set
#
twoStepFlag             1
clientOnly              0
socket_priority         0
priority1               128
priority2               128
domainNumber            0
#utc_offset             37
clockClass              248
clockAccuracy           0xFE
offsetScaledLogVariance 0xFFFF
free_running            0
freq_est_interval       1
dscp_event              0
dscp_general            0
dataset_comparison      ieee1588
G.8275.defaultDS.localPriority  128
```

```
maxStepsRemoved         255
#
# Port Data Set
#
logAnnounceInterval     1
logSyncInterval         0
operLogSyncInterval     0
logMinDelayReqInterval  0
logMinPdelayReqInterval 0
operLogPdelayReqInterval 0
announceReceiptTimeout  3
syncReceiptTimeout      0
delay_response_timeout  0
delayAsymmetry          0
fault_reset_interval    4
neighborPropDelayThresh 20000000
serverOnly              0
G.8275.portDS.localPriority     128
asCapable               auto
BMCA                    ptp
inhibit_announce        0
inhibit_delay_req       0
ignore_source_id        0#
# Run time options
#
assume_two_step         0
logging_level           6
path_trace_enabled      0
follow_up_info          0
hybrid_e2e              0
inhibit_multicast_service       0
net_sync_monitor        0
tc_spanning_tree        0
tx_timestamp_timeout    10
unicast_listen          0
unicast_master_table    0
unicast_req_duration    3600
use_syslog              1
verbose                 0
summary_interval        0
kernel_leap             1
check_fup_sync          0
clock_class_threshold   248#
# Servo Options
#
pi_proportional_const   0.0
pi_integral_const       0.0
pi_proportional_scale   0.0
pi_proportional_exponent        -0.3
pi_proportional_norm_max        0.7
pi_integral_scale       0.0
pi_integral_exponent    0.4
pi_integral_norm_max    0.3
step_threshold          0.0
first_step_threshold    0.00002
max_frequency           900000000
clock_servo             pi
sanity_freq_limit       200000000
ntpshm_segment          0
msg_interval_request    0
servo_num_offset_values 10
servo_offset_threshold  0
write_phase_mode        0#
# Transport options
#
transportSpecific       0x0
ptp_dst_mac             01:1B:19:00:00:00
```

```
p2p_dst_mac             01:80:C2:00:00:0E
udp_ttl                 1
udp6_scope              0x0E
uds_address             /var/run/ptp4l
uds_ro_address          /var/run/ptp4lro#
# Default interface options
#
clock_type              OC
network_transport       UDPv4
delay_mechanism         E2E
time_stamping           hardware
tsproc_mode             filter
delay_filter            moving_median
delay_filter_length     10
egressLatency           0
ingressLatency          0
# For multipule interfaces
boundary_clock_jbod     1#
# Clock description
#
productDescription      ;;
revisionData            ;;
manufacturerIdentity    00:00:00
userDescription         ;
timeSource              0xA0
```

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 nor any of its direct or indirect subsidiaries and affiliates (collectively: "NVIDIA") make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume 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, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world¬wide basis. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2022 NVIDIA Corporation & affiliates. All rights reserved.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com