Low-Latency IPs with AAT Demo Instruction

Low-Latency IPs with AAT Demo Instruction

Rev1.1 30-Jun-23

This document describes how to setup Alveo accelerator card and prepare the test environment for running AAT (Accelerated Algorithmic Trading) demo. The default demo is provided by Xilinx from following link.
https://www.xilinx.com/applications/data-center/financial-technology/accelerated-algorithmic-trading.html

The AAT demo is modified to use LL10GEMAC-IP, UDP10GRx-IP, and TOE10GLL-IP from Design Gateway instead of 10G/25G Ethernet subsystem, UDP/IP kernel, and TCP/IP kernel to achieve the lower latency time. More details of LL10GEMAC-IP, UDP10GRx-IP, and TOE10GLL-IP latency time are described in the datasheet.
https://dgway.com/products/IP/Lowlatency-IP/dg_ll10gemacip_data_sheet_xilinx_en.pdf
https://dgway.com/products/IP/Lowlatency-IP/dg_udp10grxip_data_sheet_xilinx_en.pdf
https://dgway.com/products/IP/Lowlatency-IP/dg_toe10gllip_data_sheet_xilinx_en.pdf

The AAT demo is run by using Alveo accelerator card plugged into the system that supports to run the accelerator card demo. The accelerator card has QSFP+ connector which supports up to 4x10G Ethernet connection. The demo uses at least two of 10G Ethernet connections, one for transferring example market data via UDP protocol and another for transferring the trade order via FIX over TCP. Therefore, the additional system with two of 10G Ethernet connections must be prepared.

In this document, Test PC is prepared integrating a 10G Ethernet card that supports two 10G Ethernet connections. “tcpreplay” is run to send the sample market data. While “TCP port” is opened to receive the trade order from the accelerator card via TCP. The accelerator card is plugged-in to Turnkey accelerator system and the user runs the test operation on the accelerator card by using “aat_dgllip_shell_exe”.

1 Test environment

Before running the test, please prepare following test environment.

- Alveo accelerator card: U250 card or U50 card
- Turnkey accelerator system, TKAS-D2101, for Alveo accelerator card
  https://dgway.com/AcceleratorCards.html
- 10 Gb Ethernet cable: QSFP+ to four SFP+ cable
  https://www.finisar.com/active-optical-cables/fcbn510qe2cxx
- Test PC for 10Gb Ethernet transferring with Turnkey accelerator system
  - Ubuntu 20.04 LTS Server OS
  - Sample market data for running AAT demo
  - TCPreplay package for transmitting market data
  - Two ports of 10G Ethernet connection such as Intel X710-DA2

    https://ark.intel.com/content/www/us/en/ark/products/83964/intel-ethernet-converged-network-adapter-x710da2.html

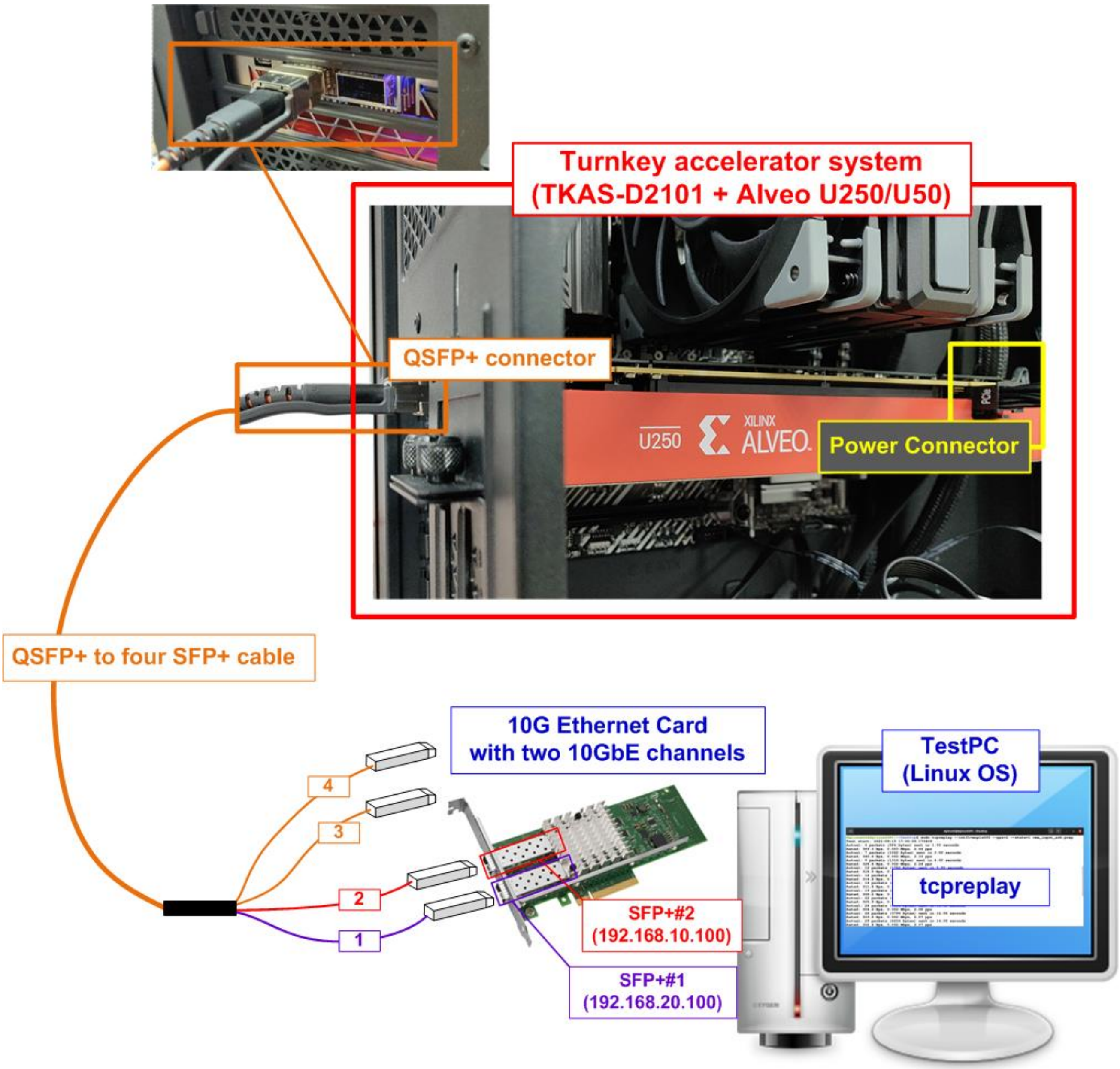


Figure 1-1 DG low-latency IPs with AAT demo (FPGA <-> PC) on Alveo U250/U50 card

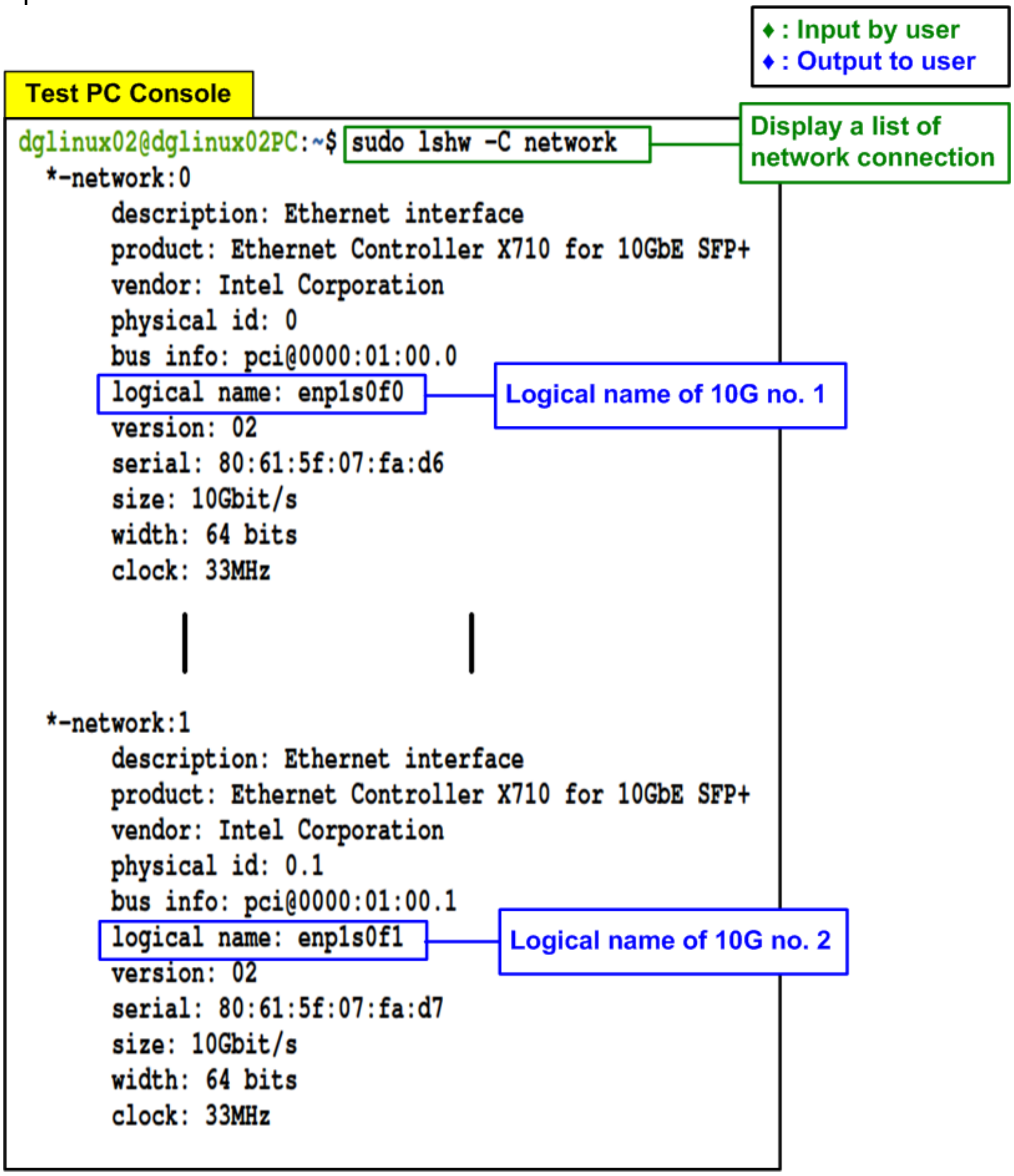
2 Test PC setup

This topic shows how to prepare Test PC for transferring market data and the trade order with Alveo accelerator card. The example is the setting on Ubuntu 20.04 LTS Server OS.

2.1 IP Address setting for two ports of 10Gb Ethernet

Please confirm the logical name of Ethernet port that connects to SFP+#1 and SFP+#2 cable. The logical name depends on the test environment. It needs to configure the correct IP address for SFP+#1 and SFP+#2 connection.

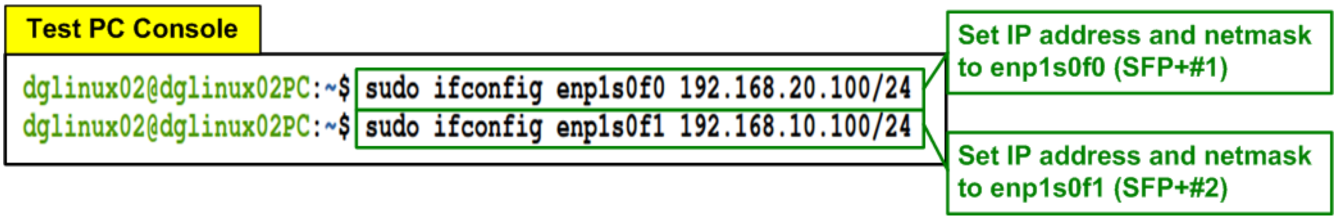
1) To list the logical name of 10G Ethernet port on Linux terminal, type "lshw -C network". Figure 2-1 shows the example result of the logical name of two 10Gb Ethernet connections. In this figure, "enp1s0f0" is Ethernet port of SFP+#1 while "enp1s0f1" is Ethernet port of SFP+#2.


♦ : Input by user
♦ : Output to user
Test PC Console
Display a list of network connection
Logical name of 10G no. 1
Logical name of 10G no. 2

```
dglinux02@dglinux02PC:~$ sudo lshw -C network
  *-network:0
       description: Ethernet interface
       product: Ethernet Controller X710 for 10GbE SFP+
       vendor: Intel Corporation
       physical id: 0
       bus info: pci@0000:01:00.0
       logical name: enp1s0f0
       version: 02
       serial: 80:61:5f:07:fa:d6
       size: 10Gbit/s
       width: 64 bits
       clock: 33MHz

  *-network:1
       description: Ethernet interface
       product: Ethernet Controller X710 for 10GbE SFP+
       vendor: Intel Corporation
       physical id: 0.1
       bus info: pci@0000:01:00.1
       logical name: enp1s0f1
       version: 02
       serial: 80:61:5f:07:fa:d7
       size: 10Gbit/s
       width: 64 bits
       clock: 33MHz
```

Figure 2-1 Display logical name of 10G Ethernet port

2) Configure IP address of SFP+#1 (enp1s0f0) to “192.168.20.100” and SFP+#2 (enp1s0f1) to “192.168.10.100” respectively by using “ifconfig” command, as shown in Figure 2-2. Besides, the netmask 24 is set by using the same command.



Test PC Console

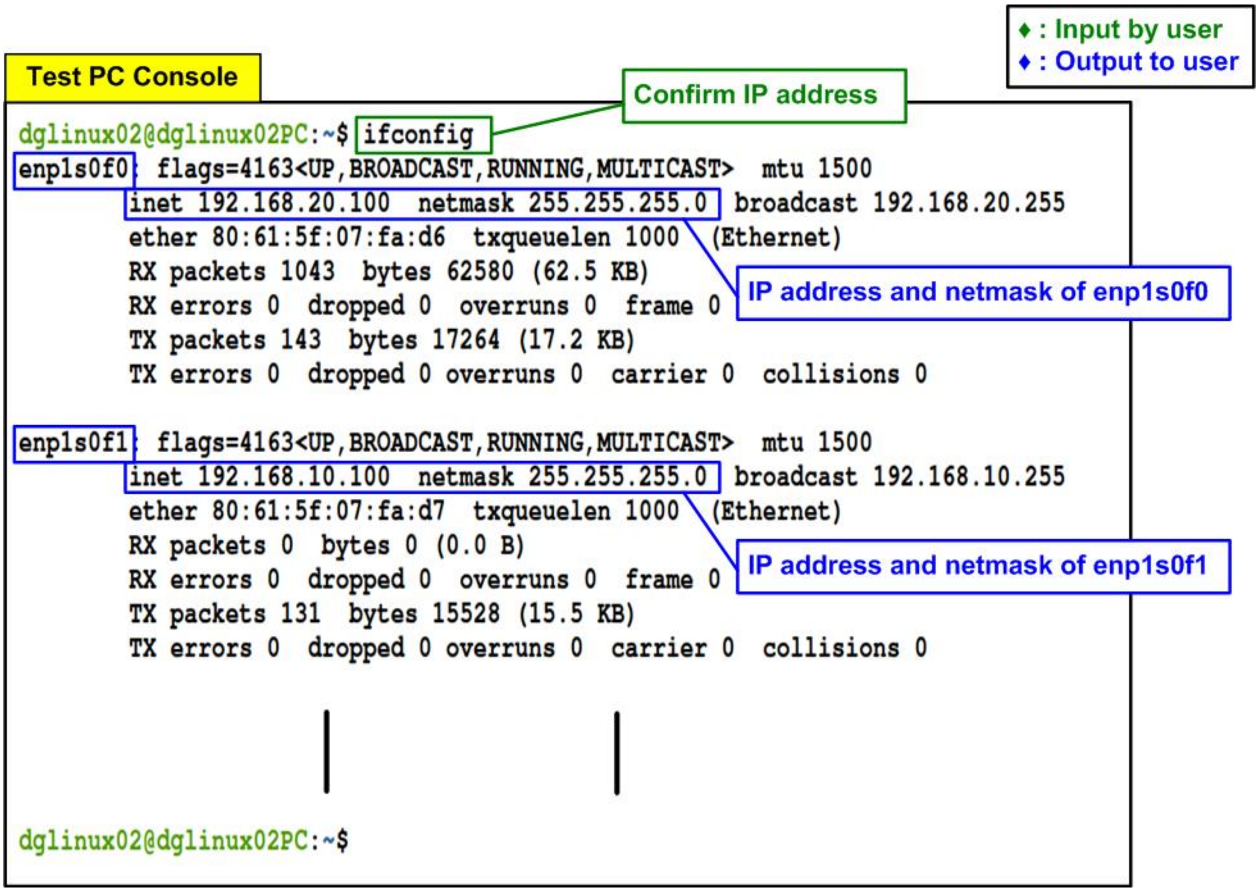
```
dglinux02@dglinux02PC:~$ sudo ifconfig enp1s0f0 192.168.20.100/24
dglinux02@dglinux02PC:~$ sudo ifconfig enp1s0f1 192.168.10.100/24
```

Set IP address and netmask to enp1s0f0 (SFP+#1)

Set IP address and netmask to enp1s0f1 (SFP+#2)

Figure 2-2 Configure IP address and netmask

3) Use “ifconfig” command to confirm IP address and netmask after setting completely.



♦ : Input by user
♦ : Output to user

Test PC Console

Confirm IP address

```
dglinux02@dglinux02PC:~$ ifconfig
enp1s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.20.100  netmask 255.255.255.0  broadcast 192.168.20.255
        ether 80:61:5f:07:fa:d6  txqueuelen 1000  (Ethernet)
        RX packets 1043  bytes 62580 (62.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 143  bytes 17264 (17.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp1s0f1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.100  netmask 255.255.255.0  broadcast 192.168.10.255
        ether 80:61:5f:07:fa:d7  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 131  bytes 15528 (15.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

dglinux02@dglinux02PC:~$
```

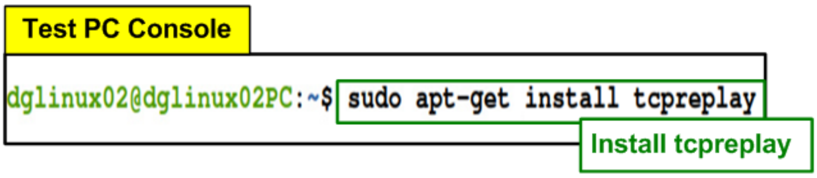
IP address and netmask of enp1s0f0

IP address and netmask of enp1s0f1

Figure 2-3 Verify IP address and netmask setting

2.2 “tcpreplay” installation

Test PC needs to install tcpreplay for running AAT demo. To install the package, type the command “sudo apt-get install tcpreplay” on the terminal, as shown in Figure 2-4.



Test PC Console

```
dglinux02@dglinux02PC:~$ sudo apt-get install tcpreplay
```

Install tcpreplay

Figure 2-4 Install tcpreplay

3 Test environment setting

This topic shows the steps to prepare Turnkey acceleration system (TKAS-D2101 with U250/U50) to run AAT demo.

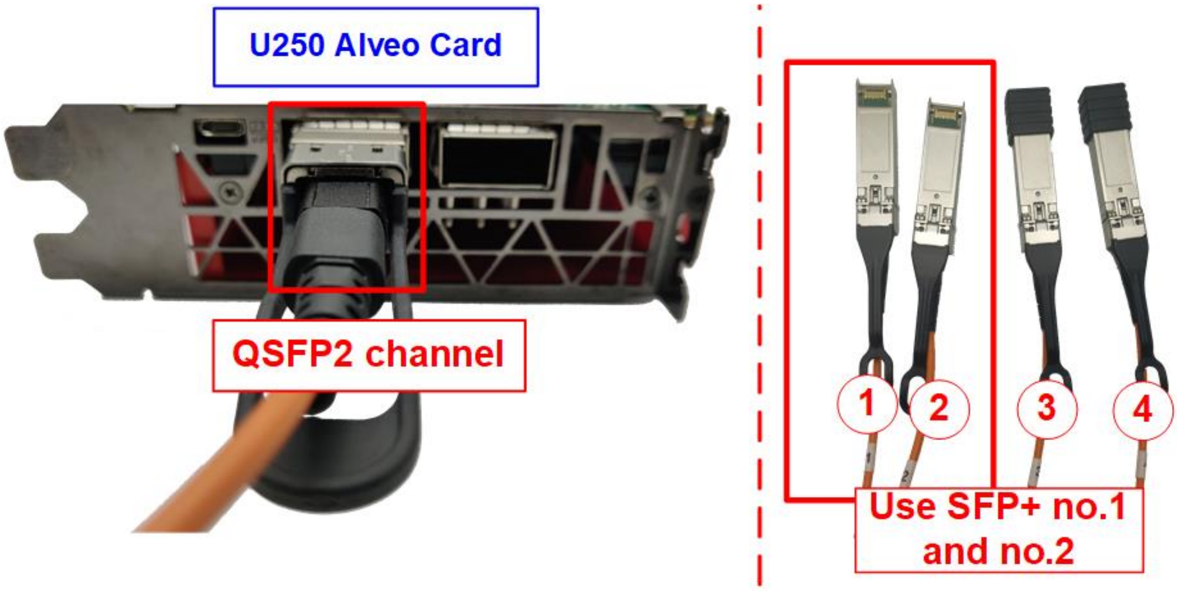
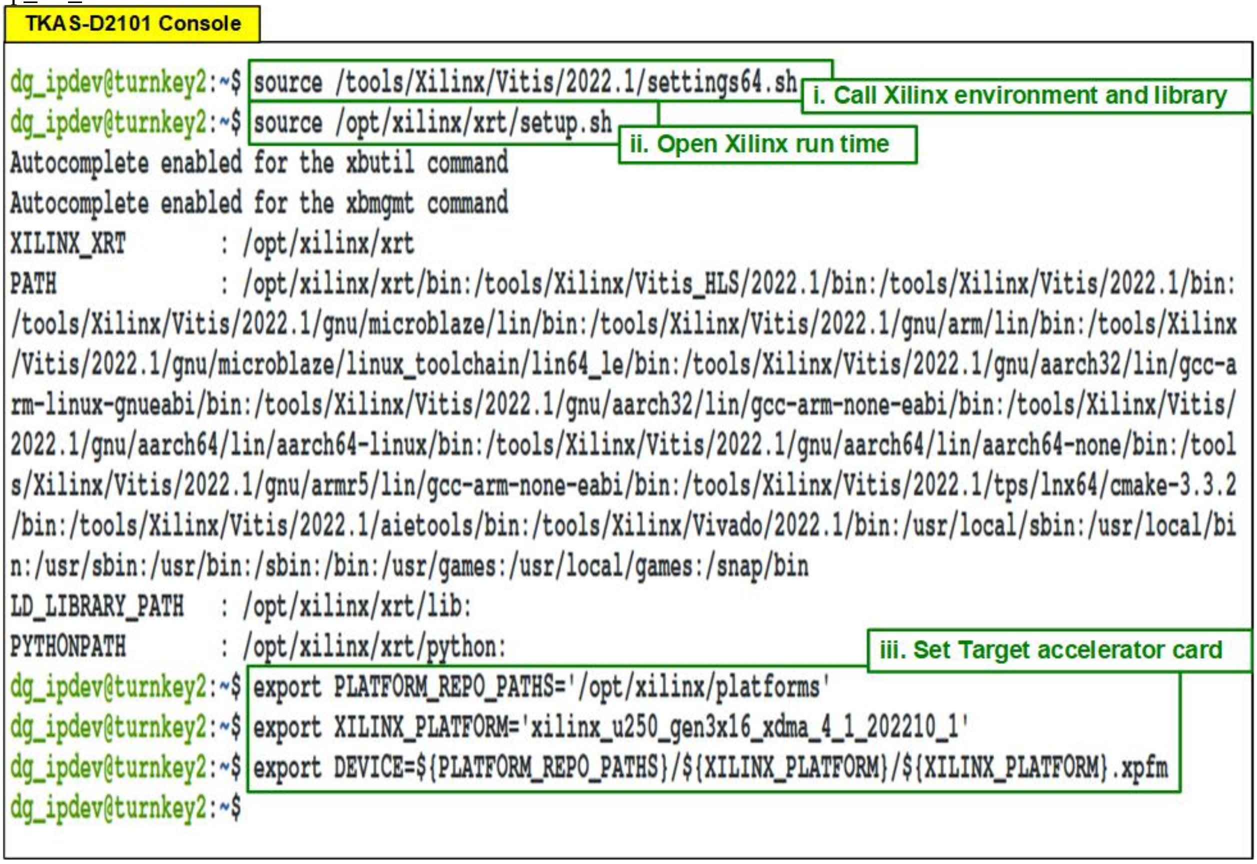


Figure 3-1 QSFP+ channel using on U250 board

1) Connect QSFP+ to four SFP+ cable (4x10G Ethernet cable) between Alveo accelerator card (U250/U50) and Test PC. Two SFP+ connectors (SFP+ no.1 and no.2) are applied.
   i. On U250 card, it has two QSFP+ channels, so insert QSFP+ cable to QSFP2 channel. While there is only one QSFP+ channel that can insert QSFP+ cable on U50 card.
   ii. Connect SFP+ no.1 (192.168.20.100) and SFP+ no.2 (192.168.10.100) to 10Gb Ethernet channel on Test PC.


TKAS-D2101 Console
i. Call Xilinx environment and library
ii. Open Xilinx run time
iii. Set Target accelerator card

```
dg_ipdev@turnkey2:~$ source /tools/Xilinx/Vitis/2022.1/settings64.sh
dg_ipdev@turnkey2:~$ source /opt/xilinx/xrt/setup.sh
Autocomplete enabled for the xbutil command
Autocomplete enabled for the xbmgmt command
XILINX_XRT        : /opt/xilinx/xrt
PATH              : /opt/xilinx/xrt/bin:/tools/Xilinx/Vitis_HLS/2022.1/bin:/tools/Xilinx/Vitis/2022.1/bin:
/tools/Xilinx/Vitis/2022.1/gnu/microblaze/lin/bin:/tools/Xilinx/Vitis/2022.1/gnu/arm/lin/bin:/tools/Xilinx
/Vitis/2022.1/gnu/microblaze/linux_toolchain/lin64_le/bin:/tools/Xilinx/Vitis/2022.1/gnu/aarch32/lin/gcc-a
rm-linux-gnueabi/bin:/tools/Xilinx/Vitis/2022.1/gnu/aarch32/lin/gcc-arm-none-eabi/bin:/tools/Xilinx/Vitis/
2022.1/gnu/aarch64/lin/aarch64-linux/bin:/tools/Xilinx/Vitis/2022.1/gnu/aarch64/lin/aarch64-none/bin:/tool
s/Xilinx/Vitis/2022.1/gnu/armr5/lin/gcc-arm-none-eabi/bin:/tools/Xilinx/Vitis/2022.1/tps/lnx64/cmake-3.3.2
/bin:/tools/Xilinx/Vitis/2022.1/aietools/bin:/tools/Xilinx/Vivado/2022.1/bin:/usr/local/sbin:/usr/local/bi
n:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
LD_LIBRARY_PATH   : /opt/xilinx/xrt/lib:
PYTHONPATH        : /opt/xilinx/xrt/python:
dg_ipdev@turnkey2:~$ export PLATFORM_REPO_PATHS='/opt/xilinx/platforms'
dg_ipdev@turnkey2:~$ export XILINX_PLATFORM='xilinx_u250_gen3x16_xdma_4_1_202210_1'
dg_ipdev@turnkey2:~$ export DEVICE=${PLATFORM_REPO_PATHS}/${XILINX_PLATFORM}/${XILINX_PLATFORM}.xpfm
dg_ipdev@turnkey2:~$
```

Figure 3-2 Xilinx environment setting on TKAS-D2101 for U250

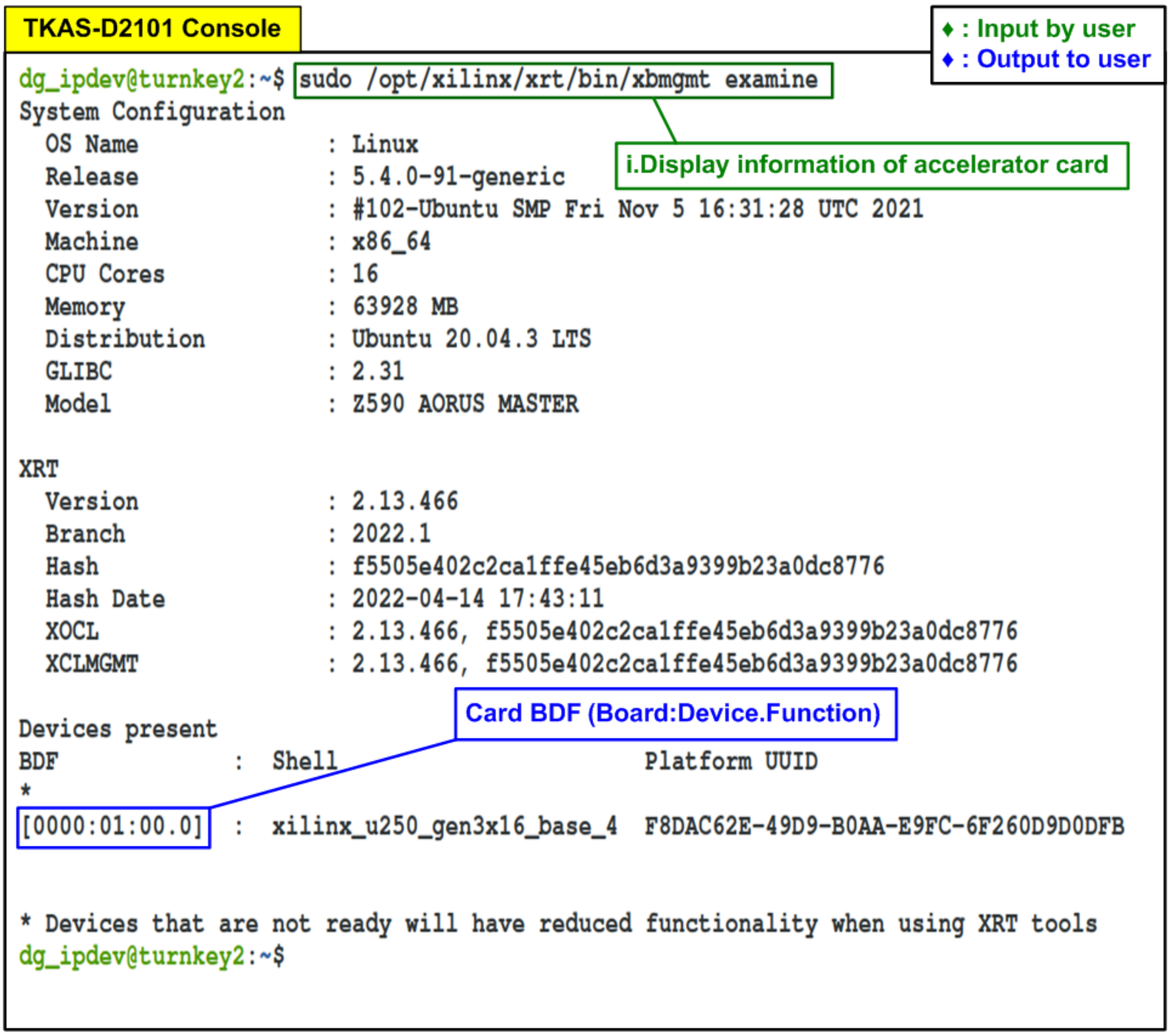
2) Prepare Xilinx environment and library on TKAS-D2101 which connects U250/U50 through PCIe connector.
   i. Input command to setup Xilinx environment and library on terminal
      >> source <install path>/Vivado/2022.1/settings64.sh
   ii. Input command to open Xilinx run time
      >> source /opt/xilinx/xrt/setup.sh
   iii. Input command to set environment for the accelerator card.
      >> export PLATFORM_REPO_PATHS='/opt/xilinx/platforms'

   For U250 card
   >> export XILINX_PLATFORM='xilinx_u250_gen3x16_xdma_4_1_202210_1'
   For U50 card
   >> export XILINX_PLATFORM='xilinx_u50_gen3x16_xdma_5_202210_1'

   >> export DEVICE=${PLATFORM_REPO_PATHS}/${XILINX_PLATFORM}/
   ${XILINX_PLATFORM}.xpfm

3) For U250 card, it needs to program the shell partition to accelerator card once after system bootup. More details are described in page 26 of UG1301 (V2.0).
   https://www.xilinx.com/support/documentation/boards_and_kits/accelerator-cards/2_0/ug1301-getting-started-guide-alveo-accelerator-cards.pdf

   i. Detect the accelerator card which connects on TKAS-D2101 to get the Card BDF ID by using following command.
      >> sudo /opt/xilinx/xrt/bin/xbmgmt examine



TKAS-D2101 Console

♦ : Input by user
♦ : Output to user

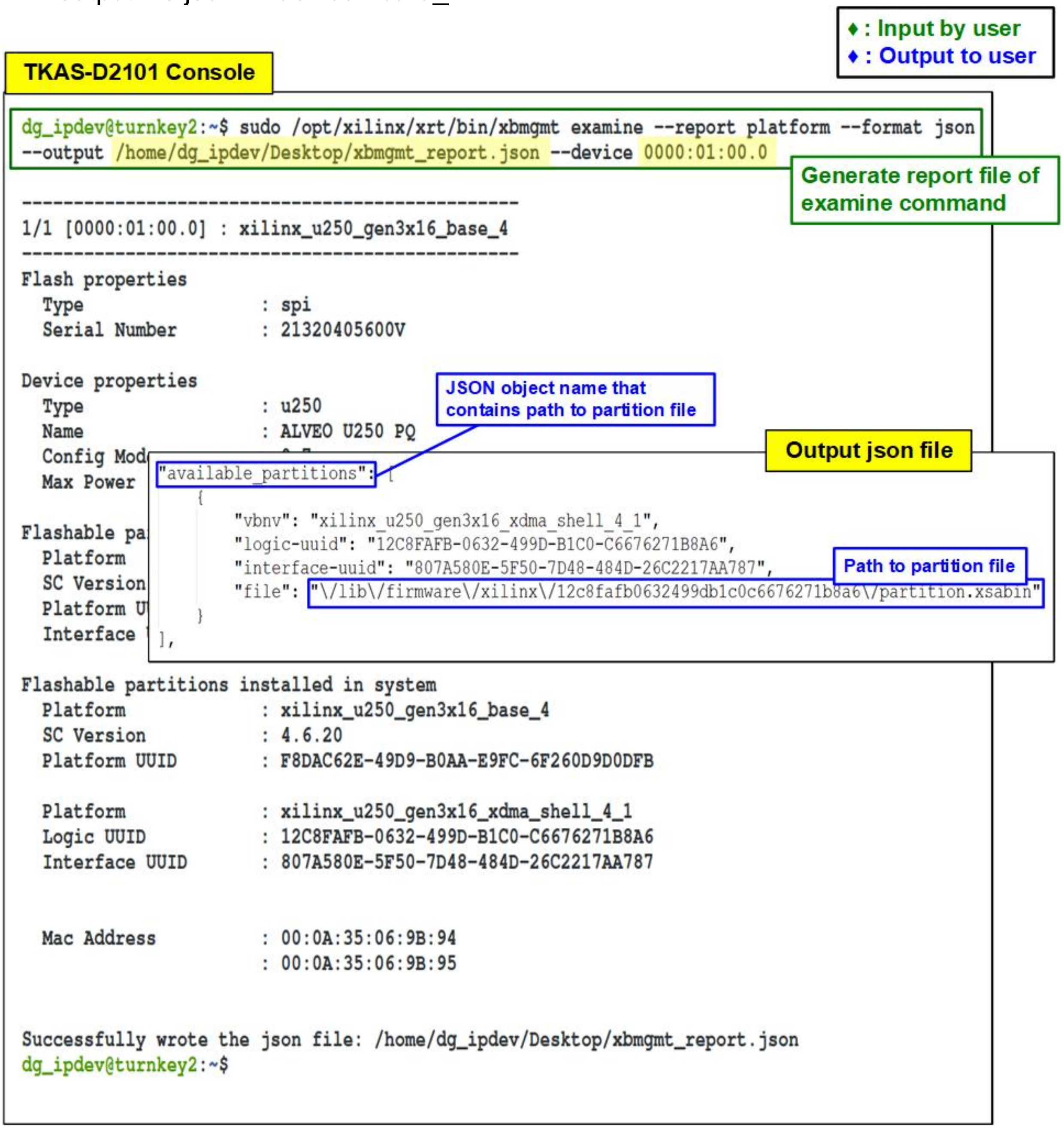
```
dg_ipdev@turnkey2:~$ sudo /opt/xilinx/xrt/bin/xbmgmt examine
System Configuration
  OS Name                  : Linux
  Release                  : 5.4.0-91-generic
  Version                  : #102-Ubuntu SMP Fri Nov 5 16:31:28 UTC 2021
  Machine                  : x86_64
  CPU Cores                : 16
  Memory                   : 63928 MB
  Distribution             : Ubuntu 20.04.3 LTS
  GLIBC                    : 2.31
  Model                    : Z590 AORUS MASTER

XRT
  Version                  : 2.13.466
  Branch                   : 2022.1
  Hash                     : f5505e402c2ca1ffe45eb6d3a9399b23a0dc8776
  Hash Date                : 2022-04-14 17:43:11
  XOCL                     : 2.13.466, f5505e402c2ca1ffe45eb6d3a9399b23a0dc8776
  XCLMGMT                  : 2.13.466, f5505e402c2ca1ffe45eb6d3a9399b23a0dc8776

Devices present
BDF             :  Shell                        Platform UUID
*
[0000:01:00.0]  :  xilinx_u250_gen3x16_base_4   F8DAC62E-49D9-B0AA-E9FC-6F260D9D0DFB


* Devices that are not ready will have reduced functionality when using XRT tools
dg_ipdev@turnkey2:~$
```

i.Display information of accelerator card

Card BDF (Board:Device.Function)

Figure 3-3 Examine the accelerator card

ii. Determine the partition file with full path by using following command to generate the JSON output file which contains the path of partition file. "card_BDF" (output from step 3i) is applied to be a parameter for this command.
>> sudo /opt/xilinx/xrt/bin/xbmgmt examine --report platform --format json --output <output file.json> --device <card_BDF>

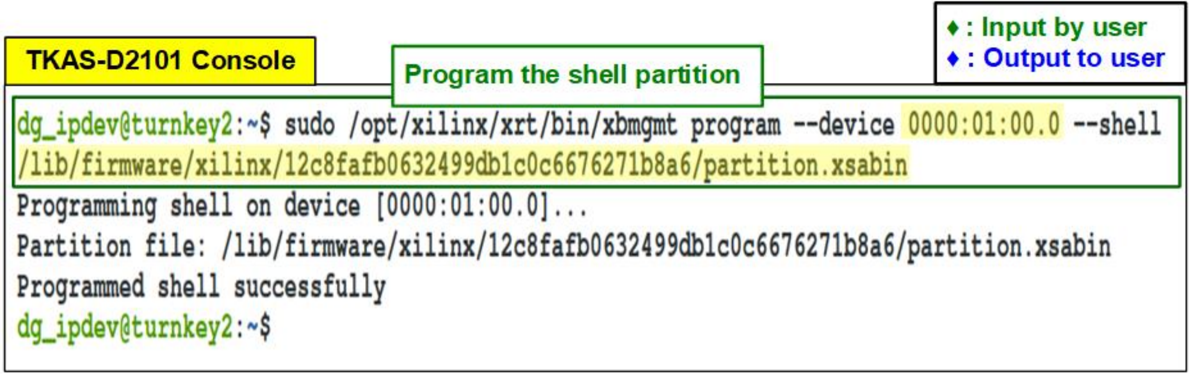

♦ : Input by user
♦ : Output to user
TKAS-D2101 Console
Generate report file of examine command
JSON object name that contains path to partition file
Output json file
Path to partition file

```
dg_ipdev@turnkey2:~$ sudo /opt/xilinx/xrt/bin/xbmgmt examine --report platform --format json
--output /home/dg_ipdev/Desktop/xbmgmt_report.json --device 0000:01:00.0

-----------------------------------------------
1/1 [0000:01:00.0] : xilinx_u250_gen3x16_base_4
-----------------------------------------------
Flash properties
  Type                 : spi
  Serial Number        : 21320405600V

Device properties
  Type                 : u250
  Name                 : ALVEO U250 PQ
  Config Mod
  Max Power

Flashable pa
  Platform
  SC Version
  Platform U
  Interface

Flashable partitions installed in system
  Platform             : xilinx_u250_gen3x16_base_4
  SC Version           : 4.6.20
  Platform UUID        : F8DAC62E-49D9-B0AA-E9FC-6F260D9D0DFB

  Platform             : xilinx_u250_gen3x16_xdma_shell_4_1
  Logic UUID           : 12C8FAFB-0632-499D-B1C0-C6676271B8A6
  Interface UUID       : 807A580E-5F50-7D48-484D-26C2217AA787


  Mac Address          : 00:0A:35:06:9B:94
                       : 00:0A:35:06:9B:95


Successfully wrote the json file: /home/dg_ipdev/Desktop/xbmgmt_report.json
dg_ipdev@turnkey2:~$
```

```
"available_partitions": [
    {
        "vbnv": "xilinx_u250_gen3x16_xdma_shell_4_1",
        "logic-uuid": "12C8FAFB-0632-499D-B1C0-C6676271B8A6",
        "interface-uuid": "807A580E-5F50-7D48-484D-26C2217AA787",
        "file": "\/lib\/firmware\/xilinx\/12c8fafb0632499db1c0c6676271b8a6\/partition.xsabin"
    }
],
```

Figure 3-4 Generate JSON report file of xbmgmt

iii. Program the shell partition to the target Alveo card by the following command along with the "card_BDF" and the path of partition file which are determined by step 3i and step 3ii. The path to partition file in output JSON file needs to remove the backslash character ('\') before using in this command.
>> sudo /opt/xilinx/xrt/bin/xbmgmt program --device <card_BDF> --shell <path to partition file>

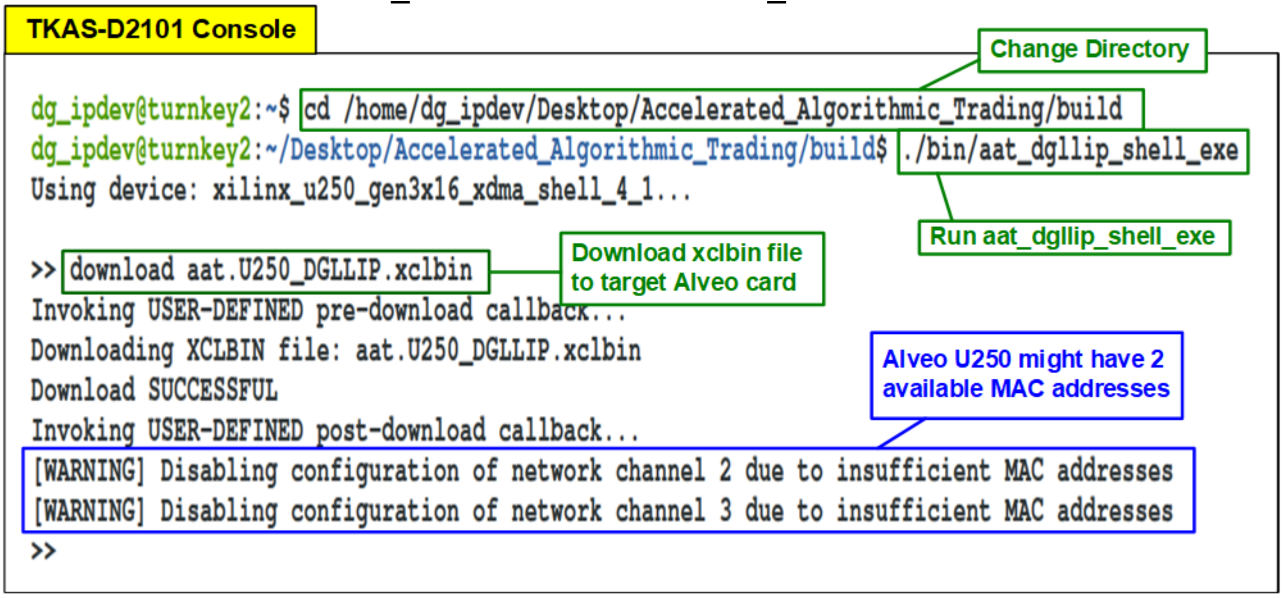


```
dg_ipdev@turnkey2:~$ sudo /opt/xilinx/xrt/bin/xbmgmt program --device 0000:01:00.0 --shell
/lib/firmware/xilinx/12c8fafb0632499db1c0c6676271b8a6/partition.xsabin
Programming shell on device [0000:01:00.0]...
Partition file: /lib/firmware/xilinx/12c8fafb0632499db1c0c6676271b8a6/partition.xsabin
Programmed shell successfully
dg_ipdev@turnkey2:~$
```

Figure 3-5 Program the shell partition

4) Copy "aat.U250/U50_DGLLIP.xclbin" file that is provided by DesignGateway to the directory "../Accelerated_Algorithmic_Trading/build".
The default path of software application "aat_dgllip_shell_exe" in AAT demo design is "../Accelerated_Algorithmic_Trading/build/bin".
Note: Xilinx AAT reference design uses "aat_shell_exe" as a software name. Since the software is modified, it is re-named to "aat_dgllip_shell_exe". If there is no "aat_dgllip_shell_exe" in bin directory, user needs to build it following the description in chapter 7 (Building and Running the AAT) of "UG1067 Accelerated Algorithmic Trading User Guide" document.

5) Browse to directory that includes "./bin/aat_dgllip_shell_exe" and xclbin file of the demo. After that, run the demo using following command.
>> cd <directory of xclbin file>
>> ./bin/aat_dgllip_shell_exe
>> download aat.U250_DGLLIP.xclbin or aat.U50_DGLLIP.xclbin


TKAS-D2101 Console
Change Directory
Run aat_dgllip_shell_exe
Download xclbin file to target Alveo card
Alveo U250 might have 2 available MAC addresses

```
dg_ipdev@turnkey2:~$ cd /home/dg_ipdev/Desktop/Accelerated_Algorithmic_Trading/build
dg_ipdev@turnkey2:~/Desktop/Accelerated_Algorithmic_Trading/build$ ./bin/aat_dgllip_shell_exe
Using device: xilinx_u250_gen3x16_xdma_shell_4_1...

>> download aat.U250_DGLLIP.xclbin
Invoking USER-DEFINED pre-download callback...
Downloading XCLBIN file: aat.U250_DGLLIP.xclbin
Download SUCCESSFUL
Invoking USER-DEFINED post-download callback...
[WARNING] Disabling configuration of network channel 2 due to insufficient MAC addresses
[WARNING] Disabling configuration of network channel 3 due to insufficient MAC addresses
>>
```

Figure 3-6 Download xclbin file

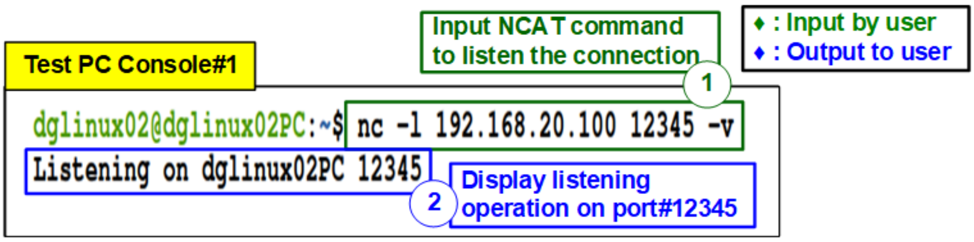
4 Run AAT Demo

To run the demo, there are three steps for user running. The first step is the initialization process for preparing the connection and parameter configuration. Next is market data transmission process to start sending market data from Test PC. The last step is the test status monitoring that is returned by the accelerator card to show the result on the console of TKAS-D2101. More details of each step are described as follows.

4.1 Initialization

To run AAT demo, the user must input the command on Test PC console to listen the specific port that is applied to receive the order packet from accelerator card after finishing processing market data. Similarly, the accelerator card must be configured to set up the parameters for receiving market data and sending the order packet by using script file "demo_setup_cfg". More details for system initialization are described as follows.

1) On Test PC console, type following command to listen the port no. 12345.
>> nc -l 192.168.20.100 12345 -v



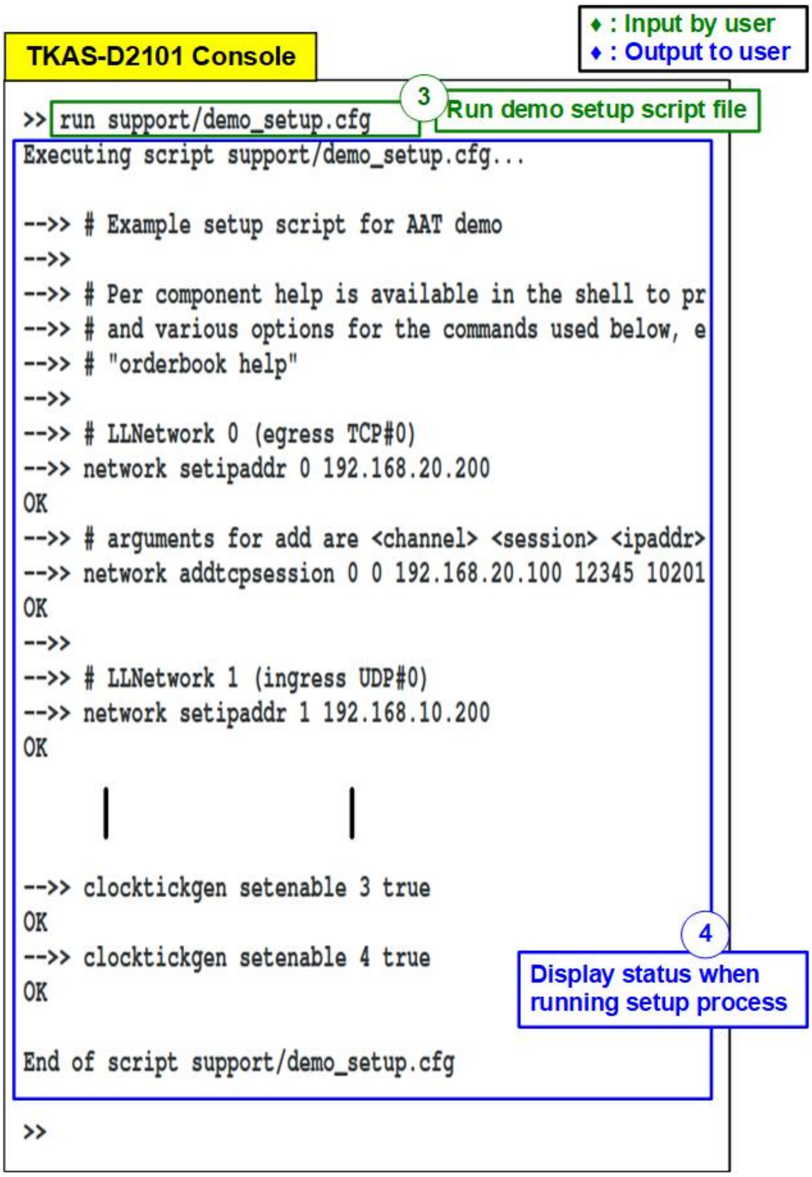
Test PC Console#1
Input NCAT command to listen the connection
1
♦ : Input by user
♦ : Output to user
2
Display listening operation on port#12345

```
dglinux02@dglinux02PC:~$ nc -l 192.168.20.100 12345 -v
Listening on dglinux02PC 12345
```

Figure 4-1 Listen TCP port on Test PC

2) After that, the confirmation message ("Listening on <Test PC name> 12345") is displayed on the console to confirm port is listening, as shown in Figure 4-1.

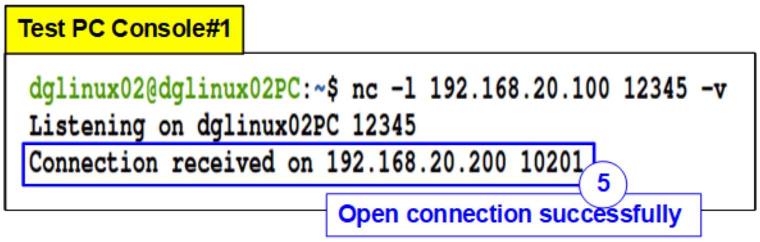
3) Run script file on TKAS-D2101 to setup the parameters for processing market data by using following command.
   >> run support/demo_setup.cfg
   Note: demo_setup.cfg can run only one time after downloading xclbin file. To re-run the script file, it needs to re-download xclbin file.


♦ : Input by user
♦ : Output to user
TKAS-D2101 Console
3
Run demo setup script file
4
Display status when running setup process

```
>> run support/demo_setup.cfg
Executing script support/demo_setup.cfg...

-->> # Example setup script for AAT demo
-->>
-->> # Per component help is available in the shell to pr
-->> # and various options for the commands used below, e
-->> # "orderbook help"
-->>
-->> # LLNetwork 0 (egress TCP#0)
-->> network setipaddr 0 192.168.20.200
OK
-->> # arguments for add are <channel> <session> <ipaddr>
-->> network addtcpsession 0 0 192.168.20.100 12345 10201
OK
-->>
-->> # LLNetwork 1 (ingress UDP#0)
-->> network setipaddr 1 192.168.10.200
OK



-->> clocktickgen setenable 3 true
OK
-->> clocktickgen setenable 4 true
OK

End of script support/demo_setup.cfg

>>
```

Figure 4-2 Run demo setup script

4) After that, TKAS-D2101 displays the message from setup process, as shown in Figure 4-2.
5) If the parameter is configured successfully, Test PC console displays the message that the port is opened successfully ("Connection received on 192.168.20.200 10201"), as shown in Figure 4-3.



```
dglinux02@dglinux02PC:~$ nc -l 192.168.20.100 12345 -v
Listening on dglinux02PC 12345
Connection received on 192.168.20.200 10201
```

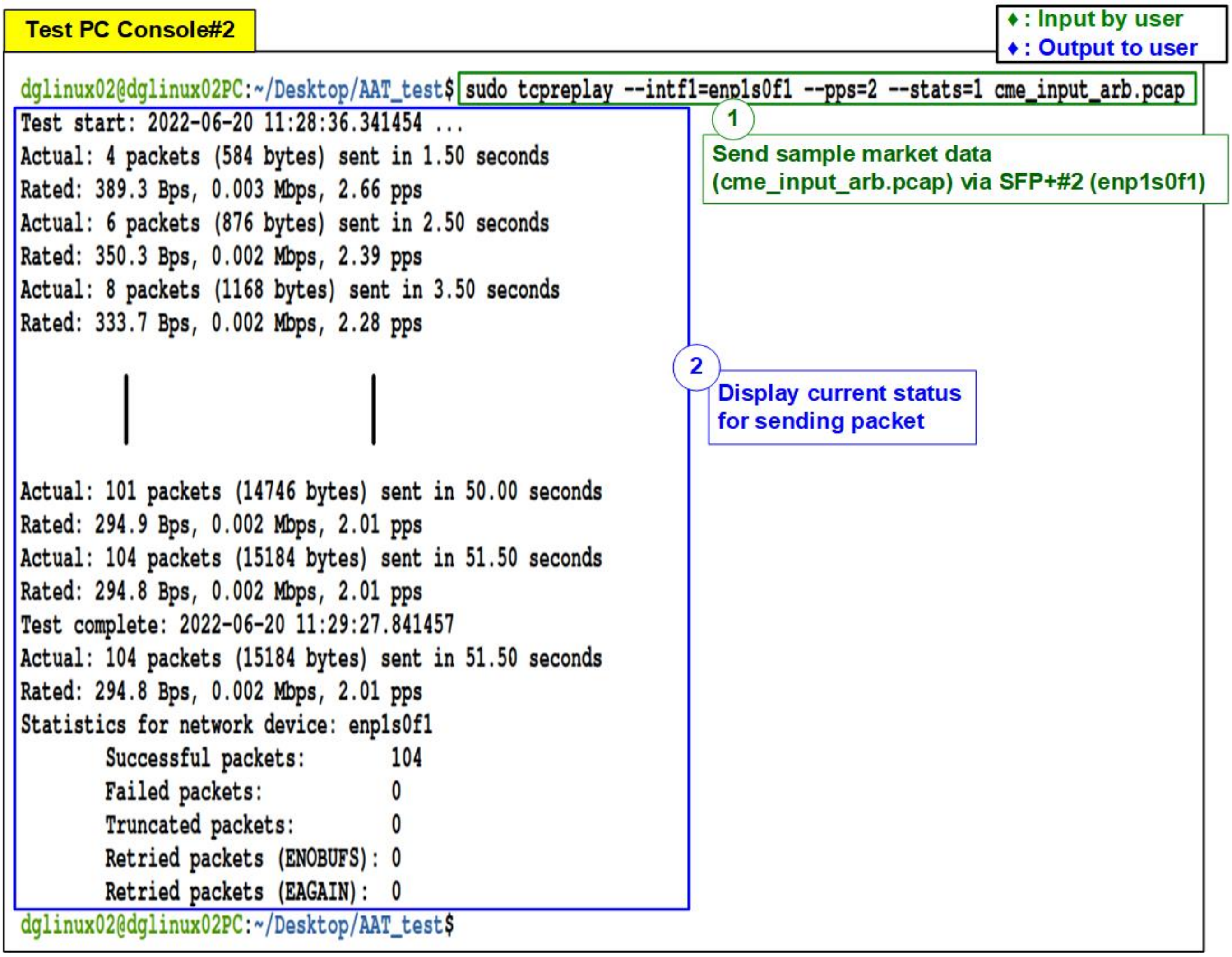
Figure 4-3 Open connection is done

4.2 Market data transmission

This topic shows how to run “tcpreplay” on Test PC to send sample market data. Also, the result on Test PC is displayed on the listening port when the accelerator card returns the order packet. Therefore, the user needs to open two consoles on Test PC, Test PC Console#1 and Test PC Console#2. Test PC Console#1 shows the details of the order packet while Test PC Console#2 is applied to send the sample market data. More details to transmit sample market data are described as follows.

1) Send sample market data (cme_input_arb.pcap) by using “tcpreplay” command. Type following command with four parameters.
   >> sudo tcpreplay –intf1=<eth I/F> --pps=<pac/sec> --stats=<stat period> <replay file>
   i) <eth I/F> : Ethernet interface for sending market data (SFP+#2: enp1s0f1)
   ii) <pac/sec> : Transfer speed by setting number of packets per second
   iii) <stat period> : Set period time in second unit to display status on console
   iv) <replay file> : File name to transmit the data, cme_input_arb.pcap

   <u>Note</u>: cme_input_arb.pcap is the sample market data, provided by Xilinx AAT demo. Please contact Xilinx to request the sample market data and AAT demo.



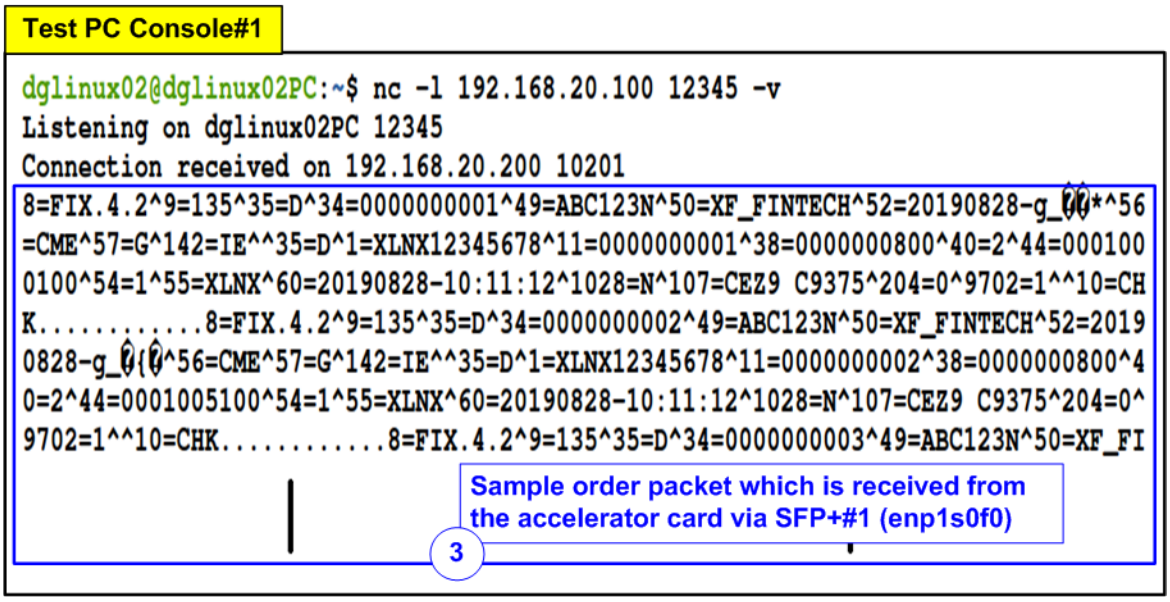
```
dglinux02@dglinux02PC:~/Desktop/AAT_test$ sudo tcpreplay --intf1=enp1s0f1 --pps=2 --stats=1 cme_input_arb.pcap
Test start: 2022-06-20 11:28:36.341454 ...
Actual: 4 packets (584 bytes) sent in 1.50 seconds
Rated: 389.3 Bps, 0.003 Mbps, 2.66 pps
Actual: 6 packets (876 bytes) sent in 2.50 seconds
Rated: 350.3 Bps, 0.002 Mbps, 2.39 pps
Actual: 8 packets (1168 bytes) sent in 3.50 seconds
Rated: 333.7 Bps, 0.002 Mbps, 2.28 pps

        |                    |

Actual: 101 packets (14746 bytes) sent in 50.00 seconds
Rated: 294.9 Bps, 0.002 Mbps, 2.01 pps
Actual: 104 packets (15184 bytes) sent in 51.50 seconds
Rated: 294.8 Bps, 0.002 Mbps, 2.01 pps
Test complete: 2022-06-20 11:29:27.841457
Actual: 104 packets (15184 bytes) sent in 51.50 seconds
Rated: 294.8 Bps, 0.002 Mbps, 2.01 pps
Statistics for network device: enp1s0f1
        Successful packets:        104
        Failed packets:            0
        Truncated packets:         0
        Retried packets (ENOBUFS): 0
        Retried packets (EAGAIN):  0
dglinux02@dglinux02PC:~/Desktop/AAT_test$
```

Send sample market data (cme_input_arb.pcap) via SFP+#2 (enp1s0f1)

Display current status for sending packet

Figure 4-4 Send sample market data by “tcpreplay”

2) After that, the console displays the status to show total number of transmit packets every second.
3) On Test PC Console#1 which the port has already connected, the console displays the received data which is the sample order packet, returned by the accelerator card to be the AAT demo result.



```
dglinux02@dglinux02PC:~$ nc -l 192.168.20.100 12345 -v
Listening on dglinux02PC 12345
Connection received on 192.168.20.200 10201
8=FIX.4.2^9=135^35=D^34=0000000001^49=ABC123N^50=XF_FINTECH^52=20190828-g_��*^56
=CME^57=G^142=IE^^35=D^1=XLNX12345678^11=0000000001^38=0000000800^40=2^44=000100
0100^54=1^55=XLNX^60=20190828-10:11:12^1028=N^107=CEZ9 C9375^204=0^9702=1^^10=CH
K............8=FIX.4.2^9=135^35=D^34=0000000002^49=ABC123N^50=XF_FINTECH^52=2019
0828-g_�{�^56=CME^57=G^142=IE^^35=D^1=XLNX12345678^11=0000000002^38=0000000800^4
0=2^44=0001005100^54=1^55=XLNX^60=20190828-10:11:12^1028=N^107=CEZ9 C9375^204=0^
9702=1^^10=CHK............8=FIX.4.2^9=135^35=D^34=0000000003^49=ABC123N^50=XF_FI
```

Sample order packet which is received from the accelerator card via SFP+#1 (enp1s0f0)

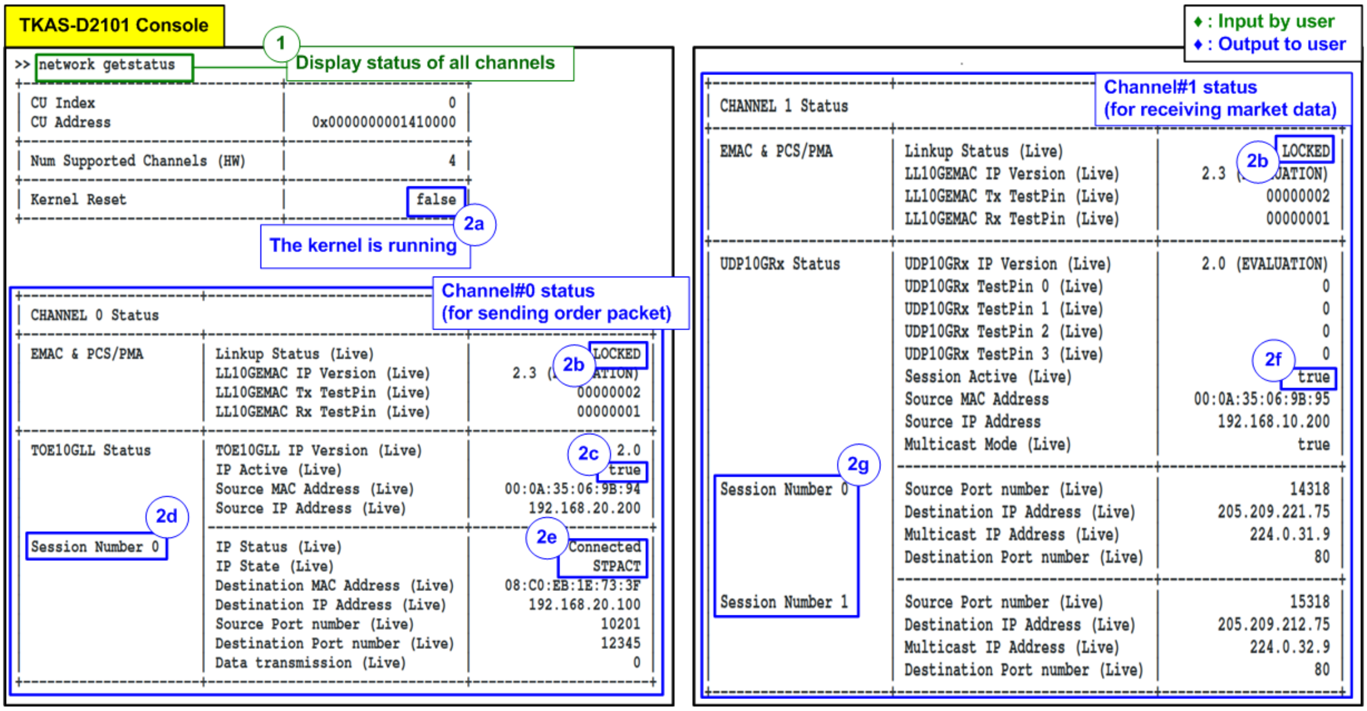
Figure 4-5 The sample data of order packet on SFP+#1 channel

4.3 AAT demo

This topic shows the example result of market data processing on the accelerator card. There are many kernels for processing the sample market data. This topic shows the result of four kernels in AAT demo design, i.e., Network kernel, Feed handler kernel, Order book kernel, and Order entry kernel. More details of the sample results are shown as follows.

4.3.1 Network Kernel

1) User inputs the following command to display the status of Network kernel.
>> network getstatus



```
>> network getstatus
+-------------------------+----------------------+
| CU Index                |                    0 |
| CU Address              | 0x0000000001410000   |
+-------------------------+----------------------+
| Num Supported Channels (HW) |                4 |
+-------------------------+----------------------+
| Kernel Reset            |                false |
+-------------------------+----------------------+

+---------------------+------------------------------+------------------+
| CHANNEL 0 Status                                                      |
+---------------------+------------------------------+------------------+
| EMAC & PCS/PMA      | Linkup Status (Live)         |           LOCKED |
|                     | LL10GEMAC IP Version (Live)  |    2.3 (   ATION) |
|                     | LL10GEMAC Tx TestPin (Live)  |         00000002 |
|                     | LL10GEMAC Rx TestPin (Live)  |         00000001 |
+---------------------+------------------------------+------------------+
| TOE10GLL Status     | TOE10GLL IP Version (Live)   |              2.0 |
|                     | IP Active (Live)             |             true |
|                     | Source MAC Address (Live)    | 00:0A:35:06:9B:94 |
|                     | Source IP Address (Live)     |   192.168.20.200 |
|                     +------------------------------+------------------+
| Session Number 0    | IP Status (Live)             |        Connected |
|                     | IP State (Live)              |           STPACT |
|                     | Destination MAC Address (Live) | 08:C0:EB:1E:73:3F |
|                     | Destination IP Address (Live) |  192.168.20.100 |
|                     | Source Port number (Live)    |            10201 |
|                     | Destination Port number (Live) |          12345 |
|                     | Data transmission (Live)     |                0 |
+---------------------+------------------------------+------------------+
```

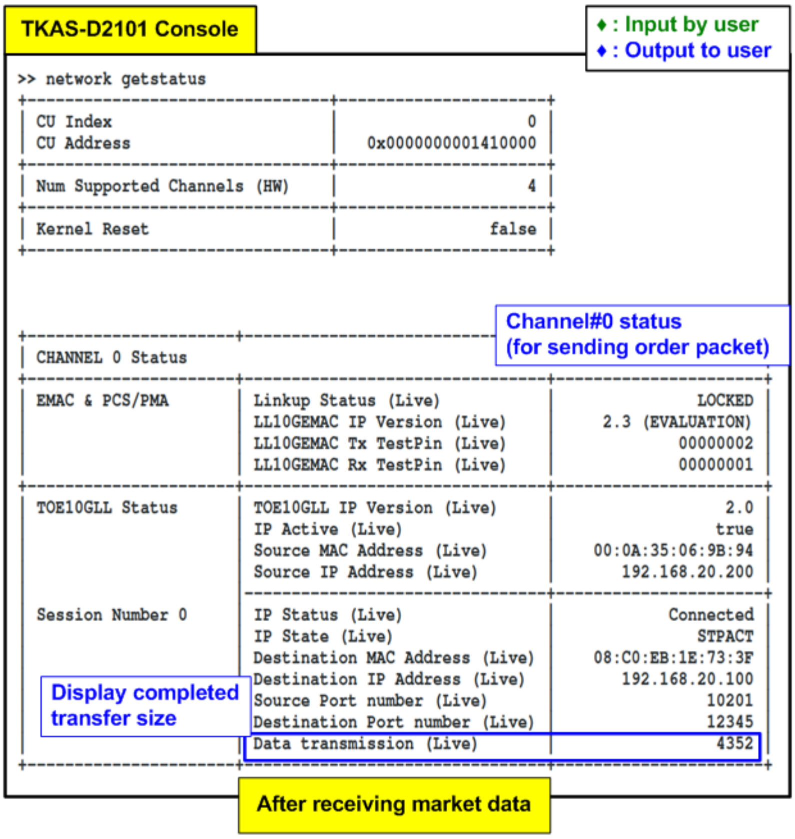
```
+---------------------+------------------------------+------------------+
| CHANNEL 1 Status                                                      |
+---------------------+------------------------------+------------------+
| EMAC & PCS/PMA      | Linkup Status (Live)         |           LOCKED |
|                     | LL10GEMAC IP Version (Live)  |    2.3 (   UATION) |
|                     | LL10GEMAC Tx TestPin (Live)  |         00000002 |
|                     | LL10GEMAC Rx TestPin (Live)  |         00000001 |
+---------------------+------------------------------+------------------+
| UDP10GRx Status     | UDP10GRx IP Version (Live)   | 2.0 (EVALUATION) |
|                     | UDP10GRx TestPin 0 (Live)    |                0 |
|                     | UDP10GRx TestPin 1 (Live)    |                0 |
|                     | UDP10GRx TestPin 2 (Live)    |                0 |
|                     | UDP10GRx TestPin 3 (Live)    |                0 |
|                     | Session Active (Live)        |             true |
|                     | Source MAC Address           | 00:0A:35:06:9B:95 |
|                     | Source IP Address            |   192.168.10.200 |
|                     | Multicast Mode (Live)        |             true |
|                     +------------------------------+------------------+
| Session Number 0    | Source Port number (Live)    |            14318 |
|                     | Destination IP Address (Live) |  205.209.221.75 |
|                     | Multicast IP Address (Live)  |       224.0.31.9 |
|                     | Destination Port number (Live) |             80 |
|                     +------------------------------+------------------+
| Session Number 1    | Source Port number (Live)    |            15318 |
|                     | Destination IP Address (Live) |  205.209.212.75 |
|                     | Multicast IP Address (Live)  |       224.0.32.9 |
|                     | Destination Port number (Live) |             80 |
+---------------------+------------------------------+------------------+
```

Figure 4-6: Network kernel status

2) There are four network channels (channel0 – channel3) on AAT demo system. The document shows the example of using channel#0 for returning sample order packet and channel#1 for receiving sample market data. Please confirm the status of channel#0 and channel#1 are in good status.

a)	Kernel Reset (Main)	: false
b)	Linkup Status (channel0 and 1)	: LOCKED
c)	TOE-IP Session Active (channel0)	: true
d)	TOE-IP Session Number (channel0)	: Number 0
e)	TOE-IP Status and State (channel0)	: Connected and STPACT
f)	UDP-IP Session Active (channel1)	: true
g)	UDP-IP Session Number (channel1)	: Number 0 and 1

Note: Network parameters are configured using script file as described in topic 4.1.



TKAS-D2101 Console

♦ : Input by user
♦ : Output to user

```
>> network getstatus
+--------------------------------+----------------------+
| CU Index                       |                    0 |
| CU Address                     |   0x0000000001410000 |
+--------------------------------+----------------------+
| Num Supported Channels (HW)    |                    4 |
+--------------------------------+----------------------+
| Kernel Reset                   |                false |
+--------------------------------+----------------------+


+----------------------+----------------------------------------------------------+
| CHANNEL 0 Status                                                                |
+----------------------+--------------------------------+-------------------------+
| EMAC & PCS/PMA       | Linkup Status (Live)           |                  LOCKED |
|                      | LL10GEMAC IP Version (Live)    |        2.3 (EVALUATION) |
|                      | LL10GEMAC Tx TestPin (Live)    |                00000002 |
|                      | LL10GEMAC Rx TestPin (Live)    |                00000001 |
+----------------------+--------------------------------+-------------------------+
| TOE10GLL Status      | TOE10GLL IP Version (Live)     |                     2.0 |
|                      | IP Active (Live)               |                    true |
|                      | Source MAC Address (Live)      |       00:0A:35:06:9B:94 |
|                      | Source IP Address (Live)       |          192.168.20.200 |
|                      +--------------------------------+-------------------------+
| Session Number 0     | IP Status (Live)               |               Connected |
|                      | IP State (Live)                |                  STPACT |
|                      | Destination MAC Address (Live) |       08:C0:EB:1E:73:3F |
|                      | Destination IP Address (Live)  |          192.168.20.100 |
|                      | Source Port number (Live)      |                   10201 |
|                      | Destination Port number (Live) |                   12345 |
|                      | Data transmission (Live)       |                    4352 |
+----------------------+--------------------------------+-------------------------+
```

Channel#0 status
(for sending order packet)

Display completed
transfer size

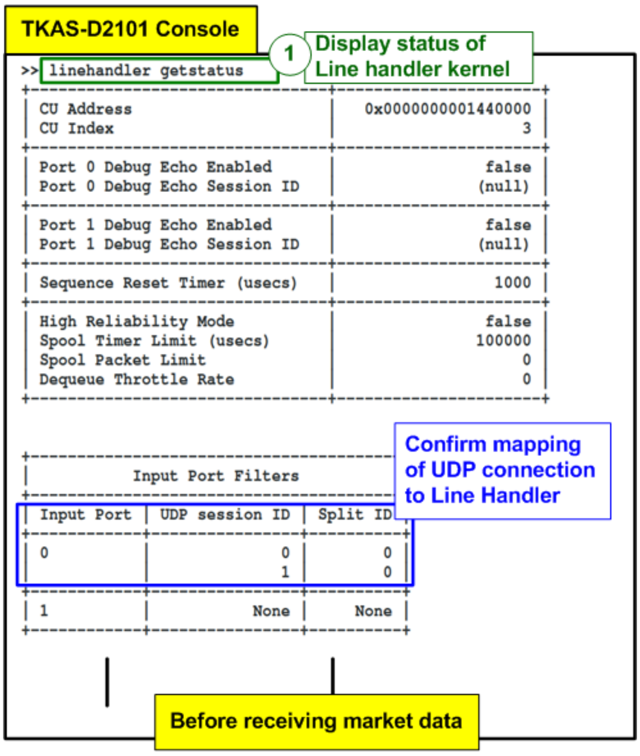
After receiving market data

Figure 4-7 Network kernel status after receiving market data

After transferring the market data, the trade order is generated via Channel#0. The data transmission to show the amount of completed transmitted data in byte unit is updated, as shown in Figure 4-7.

4.3.2 Line Handler Kernel

1) User inputs the following command to display the status of Line handler kernel.
   >> linehandler getstatus
2) Before sending the market data from Test PC, the Line Handler kernel must be set to map the UDP data from Network kernel to the specific split ID as follows.
   a) Line Handler input port 0 maps the UDP session number 0 to Split ID 0
   b) Line Handler input port 0 maps the UDP session number 1 to Split ID 0



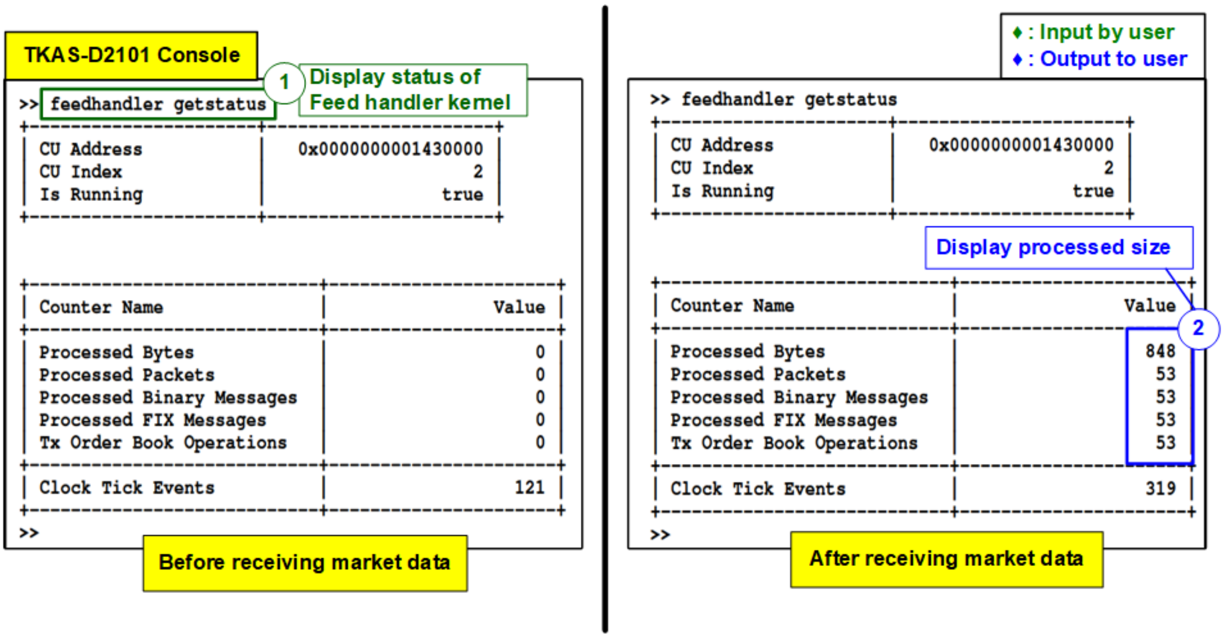
TKAS-D2101 Console
1
Display status of Line handler kernel
Confirm mapping of UDP connection to Line Handler
Before receiving market data

```
>> linehandler getstatus
+--------------------------------+----------------------+
| CU Address                     |   0x0000000001440000 |
| CU Index                       |                    3 |
+--------------------------------+----------------------+
| Port 0 Debug Echo Enabled      |                false |
| Port 0 Debug Echo Session ID   |               (null) |
+--------------------------------+----------------------+
| Port 1 Debug Echo Enabled      |                false |
| Port 1 Debug Echo Session ID   |               (null) |
+--------------------------------+----------------------+
| Sequence Reset Timer (usecs)   |                 1000 |
+--------------------------------+----------------------+
| High Reliability Mode          |                false |
| Spool Timer Limit (usecs)      |               100000 |
| Spool Packet Limit             |                    0 |
| Dequeue Throttle Rate          |                    0 |
+--------------------------------+----------------------+


+---------------------------------------
|           Input Port Filters
+---------------------------------------
| Input Port | UDP session ID | Split ID
+------------+----------------+----------+
| 0          |              0 |        0 |
|            |              1 |        0 |
+------------+----------------+----------+
| 1          |           None |     None |
+------------+----------------+----------+
```

Figure 4-8 Line Handler status

4.3.3 Feed Handler Kernel

1) User inputs the following command to display the status of Feed handler kernel.
   >> feedhandler getstatus
2) The console displays the processed data count in several units such as bytes, packets, and messages. As shown in Figure 4-9, the left window shows the processed data count is equal to zero before starting transmitting sample market data. After finishing transmitting market data, the processed data count is not equal to zero.



```
>> feedhandler getstatus
+-----------------------+-----------------------+
| CU Address            |    0x0000000001430000 |
| CU Index              |                     2 |
| Is Running            |                  true |
+-----------------------+-----------------------+


+-----------------------------+---------------------+
| Counter Name                |               Value |
+-----------------------------+---------------------+
| Processed Bytes             |                   0 |
| Processed Packets           |                   0 |
| Processed Binary Messages   |                   0 |
| Processed FIX Messages      |                   0 |
| Tx Order Book Operations    |                   0 |
+-----------------------------+---------------------+
| Clock Tick Events           |                 121 |
+-----------------------------+---------------------+
>>
```

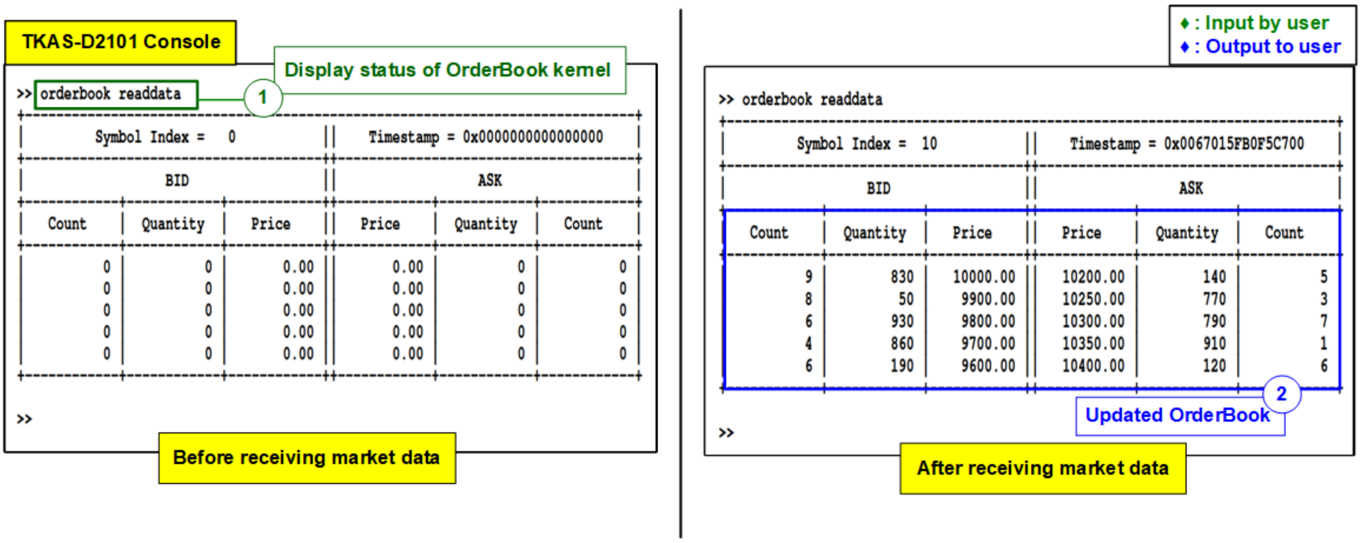
```
>> feedhandler getstatus
+-----------------------+-----------------------+
| CU Address            |    0x0000000001430000 |
| CU Index              |                     2 |
| Is Running            |                  true |
+-----------------------+-----------------------+


+-----------------------------+---------------------+
| Counter Name                |               Value |
+-----------------------------+---------------------+
| Processed Bytes             |                 848 |
| Processed Packets           |                  53 |
| Processed Binary Messages   |                  53 |
| Processed FIX Messages      |                  53 |
| Tx Order Book Operations    |                  53 |
+-----------------------------+---------------------+
| Clock Tick Events           |                 319 |
+-----------------------------+---------------------+
>>
```

Figure 4-9 Feed handler kernel status

As shown in Figure 4-4, there are 104 packets of sample market data sent by Test PC. However, there are 53 packets that are valid for Feed handler processing. Other packets are rejected by Line handler.

4.3.4 OrderBook Kernel

1) User inputs the following command to display the order book, the output from Order Book kernel.
   >> orderbook readdata
2) The console displays the current value of order book. As shown in Figure 4-10, the left window shows clean status of order book before the system transmits sample market data. While the right window shows the updated order book after transferring all sample market data. The bid/ask quantity and the bid/ask price are updated by OrderBook kernel.


TKAS-D2101 Console
Display status of OrderBook kernel
1
Before receiving market data
♦ : Input by user
♦ : Output to user
2
Updated OrderBook
After receiving market data

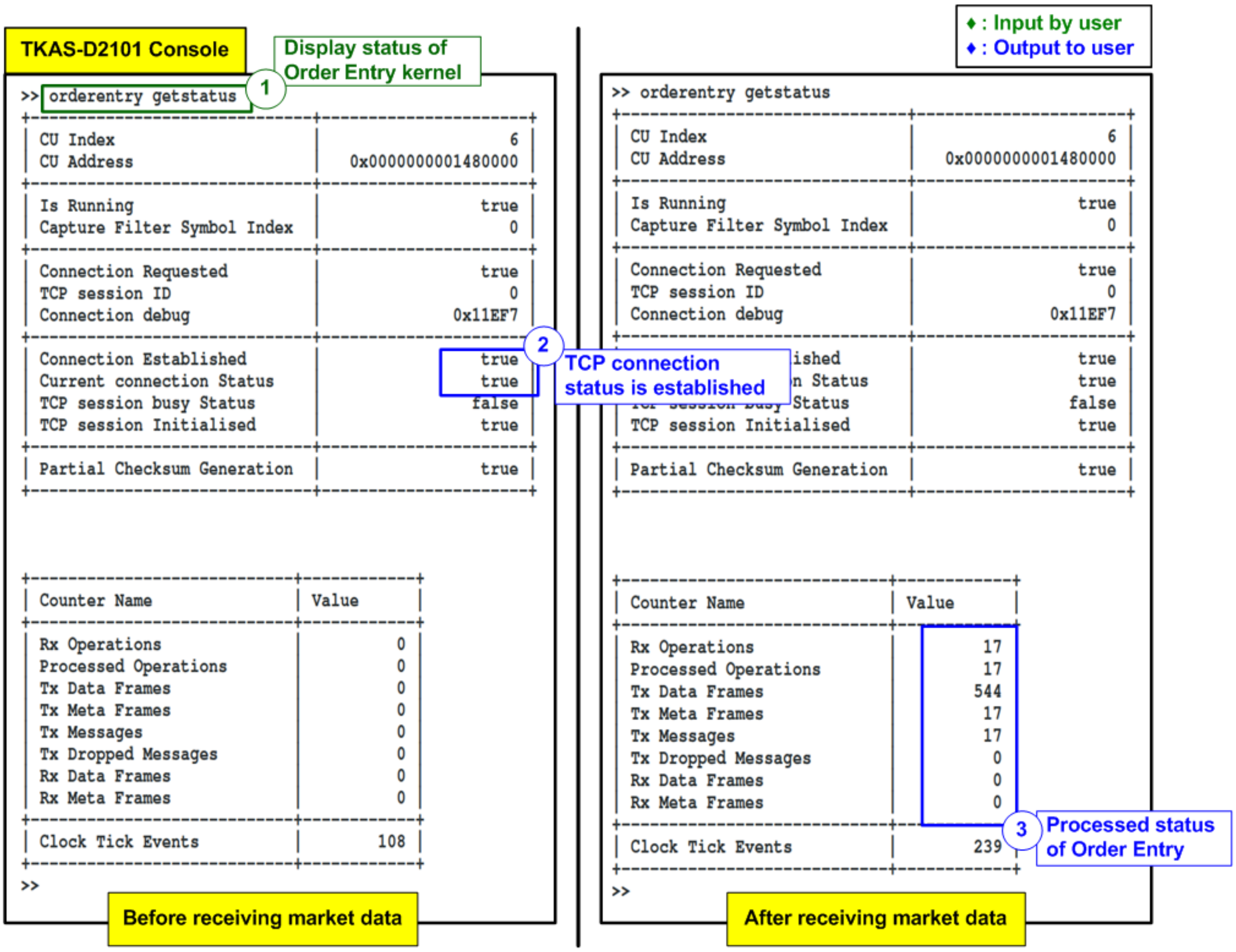
```
>> orderbook readdata
+--------------------------------------++--------------------------------------+
|         Symbol Index =    0          ||     Timestamp = 0x0000000000000000     |
+--------------------------------------++--------------------------------------+
|                 BID                  ||                 ASK                  |
+------------+------------+------------++------------+------------+------------+
|   Count    |  Quantity  |   Price    ||   Price    |  Quantity  |   Count    |
+------------+------------+------------++------------+------------+------------+
|          0 |          0 |       0.00 ||       0.00 |          0 |          0 |
|          0 |          0 |       0.00 ||       0.00 |          0 |          0 |
|          0 |          0 |       0.00 ||       0.00 |          0 |          0 |
|          0 |          0 |       0.00 ||       0.00 |          0 |          0 |
|          0 |          0 |       0.00 ||       0.00 |          0 |          0 |
+------------+------------+------------++------------+------------+------------+

>>
```

```
>> orderbook readdata
+--------------------------------------++--------------------------------------+
|         Symbol Index =   10          ||     Timestamp = 0x0067015FB0F5C700     |
+--------------------------------------++--------------------------------------+
|                 BID                  ||                 ASK                  |
+------------+------------+------------++------------+------------+------------+
|   Count    |  Quantity  |   Price    ||   Price    |  Quantity  |   Count    |
+------------+------------+------------++------------+------------+------------+
|          9 |        830 |   10000.00 ||   10200.00 |        140 |          5 |
|          8 |         50 |    9900.00 ||   10250.00 |        770 |          3 |
|          6 |        930 |    9800.00 ||   10300.00 |        790 |          7 |
|          4 |        860 |    9700.00 ||   10350.00 |        910 |          1 |
|          6 |        190 |    9600.00 ||   10400.00 |        120 |          6 |
+------------+------------+------------++------------+------------+------------+

>>
```

Figure 4-10 Updated OrderBook after finishing processing

4.3.5 Order Entry Kernel

1) User inputs the following command to display the current status of Order Entry kernel.
   >> orderentry getstatus
2) After starting AAT demo and before starting sending the sample market data, the user can check the status of Network channel#0 (TCP/IP) from Order Entry kernel status.
   i) Connection Established : true
   ii) Current Connection Status : true


TKAS-D2101 Console
Display status of Order Entry kernel
1
♦ : Input by user
♦ : Output to user
2
TCP connection status is established
3
Processed status of Order Entry
Before receiving market data
After receiving market data

```
>> orderentry getstatus
+------------------------------+------------------------+
| CU Index                     |                      6 |
| CU Address                   |     0x0000000001480000 |
+------------------------------+------------------------+
| Is Running                   |                   true |
| Capture Filter Symbol Index  |                      0 |
+------------------------------+------------------------+
| Connection Requested         |                   true |
| TCP session ID               |                      0 |
| Connection debug             |                0x11EF7 |
+------------------------------+------------------------+
| Connection Established       |                   true |
| Current connection Status    |                   true |
| TCP session busy Status      |                  false |
| TCP session Initialised      |                   true |
+------------------------------+------------------------+
| Partial Checksum Generation  |                   true |
+------------------------------+------------------------+


+----------------------------+------------+
| Counter Name               | Value      |
+----------------------------+------------+
| Rx Operations              |          0 |
| Processed Operations       |          0 |
| Tx Data Frames             |          0 |
| Tx Meta Frames             |          0 |
| Tx Messages                |          0 |
| Tx Dropped Messages        |          0 |
| Rx Data Frames             |          0 |
| Rx Meta Frames             |          0 |
+----------------------------+------------+
| Clock Tick Events          |        108 |
+----------------------------+------------+
>>
```

```
>> orderentry getstatus
+------------------------------+------------------------+
| CU Index                     |                      6 |
| CU Address                   |     0x0000000001480000 |
+------------------------------+------------------------+
| Is Running                   |                   true |
| Capture Filter Symbol Index  |                      0 |
+------------------------------+------------------------+
| Connection Requested         |                   true |
| TCP session ID               |                      0 |
| Connection debug             |                0x11EF7 |
+------------------------------+------------------------+
|                   ished      |                   true |
|                   n Status   |                   true |
|                   Status     |                  false |
| TCP session Initialised      |                   true |
+------------------------------+------------------------+
| Partial Checksum Generation  |                   true |
+------------------------------+------------------------+


+----------------------------+------------+
| Counter Name               | Value      |
+----------------------------+------------+
| Rx Operations              |         17 |
| Processed Operations       |         17 |
| Tx Data Frames             |        544 |
| Tx Meta Frames             |         17 |
| Tx Messages                |         17 |
| Tx Dropped Messages        |          0 |
| Rx Data Frames             |          0 |
| Rx Meta Frames             |          0 |
+----------------------------+------------+
| Clock Tick Events          |        239 |
+----------------------------+------------+
>>
```

Figure 4-11 Order Entry kernel status

3) After transmitting the sample market data completely, the packet count of Order Entry is updated from 0 to the new value to show total numbers of message/frames that is processed by Order Entry kernel.

5 Revision History

Revision	Date	Description
1.1	7-Sep-22	Add TOE10GLL-IP
1.0	4-Jan-22	Initial version release