Broadcom NetXtreme Controller – Whitney/Stratus/Thor

Bnxtmt (Lite) User's Guide

Document Version 221.1

Broadcom, Inc.,
15191 Alton Pkwy,

Irvine, CA 92618

www.broadcom.com

Revision History

Rev.	Date	By	Notes
1.0	06/05/2021	Richard L.	Initial release. Taken from bnxtmt manual.
1.1	06/29/2021	Stephen K.	Added provisioning check test. Added "-pvsn_prg" and "-cert_tool" CLI options to support provisioning.
221.0	08/25/2021	Richard L.	Changed versioning to match SIT release branch Added UEFI description Added supported Linux Kernels
221.1	11/23/2021	Richard L.	Updated/Corrected descriptions of command line options

TABLE OF CONTENTS

Contents

Introduction

This document provides the information on how to use Broadcom's Bnxtmt (Lite) utility for manufacturing (which is a lite version of bnxtmt that also provides dignostics utilities for Broadcom internal developers) for the following Broadcom NetXtreme controllers and their derivatives.

This manual is intended for manufacturing of Broadcom LOM customers.

This manual describes the Bnxtmt manufacturing mode commands and usage. Some tests may not apply, depending upon the controllers installed in the system. Some tests may be deprecated or removed from the bnxtmt tool.

1. Operating Environment

1.1 Linux environment

Linux Bnxtmt (Lite) currently supports the following deliveries to Linux.

Key Software:

bnxtmtdrv.ko: kernel mode diagnostic driver

bnxtmt: user mode application

load.sh: shell script used to load bnxtmtdrv.ko and launch bnxtmt

Deployment:

Grab the bnxtmt (lite) tarball from SIT release. SIT release builds and generates bnxtmt (lite) tarball for each chipset. The supported Linux Kernels are: Kernel 3.10, 4.6 for X86 architecture; and Kernel 4.18 for ARM64 architecture. So we should see the following bnxtmt tarball files under each CHIP_CFG folder(e.g. TH_A for Thor, CMB_A for Wh+, and SR_A for Stratus).

File Name	Architecture	OS	Kernel
bnxtmt-lite-xx.yy.zz.ww-x86_64.tar.gz	X86	CentOS	3.10
bnxtmt-lite-xx.yy.zz.ww-x86_64.k406.tar.gz	X86	CentOS	4.6
bnxtmt-lite-xx.yy.zz.ww-arm64.k418.tar.gz	ARM	CentOS	4.18

Untar the bnxtmt (lite) release tarball.

Issue "make" under the root of untarred bnxtmt directory

1.2 UEFI environment

UEFI Bnxtmt (Lite) currently supports the following deliveries to UEFI.

Key Software:

bnxtmt.efi: user mode application

Deployment:

Grab the bnxtmt (lite) tarball from SIT UEFI release. SIT release builds and generates bnxtmt (lite) tarball for each chipset. The supported UEFI Development Kit is UDK 2017 for X86 architecture. We should see the bnxtmt tarball file (bnxtmt-lite-uefi-xx.yy.zz.ww-x64.tar.gz) under each CHIP_CFG folder(e.g. TH_A for Thor, CMB_A for Wh+, and SR_A for Stratus).

Untar bxntmt (lite) UEFI releases tarball and copy all the files to the UEFI boot drive (e.g. USB drive).

2. Support Matrix

Bnxtmt (Lite)	**OS Support**	**Platform**	**Chip Support**
220.0 and above	Linux/UEFI	X86, ARM	Whitney+, Stratus, Thor

3. Modes of Operation

The Bnxtmt (Lite) utility has one mode of operation: manufacturing mode.

The bnxtmt utility supports one NIC card on the same system. Using bnxtmt with multiple cards on one system can run into uncertain errors.

3.1 Manufacturing Mode

Manufacturing mode is the command-line mode that should be run from Linux shell or by upper-layer manufacturing tool/scripts. Manufacturing mode command will always return with either a PASS or FAIL result code. The following command is an example of manufacturing mode testing which doing nothing:

```
[hostname] # ./load.sh(or bnxtmt.efi) -sysop -no_swap -none
```

Warning: The engineering mode is not supported by Bnxtmt (Lite). The user will see an error message when issuing "./load.sh(or load.nsh or bnxtmt.efi -eng under UEFI)" to launch Bnxtmt (Lite)

4. bnxtmt Command Line Options

4.1 Options

The followings are the available command line options in bnxtmt diagnostic tool. "./load.sh(or bnxtmt.efi) -help" will always display the supported command line options by the bnxtmt is used.

OPTIONS	DESCRIPTION
-I <iteration#>	Specify how many iterations tests need to run (Default is 1).
-ver	Display information on bnxtmt version and exits.
-log <logfile>	Log the tests' execution results into the specified file.
-stride	Obsolete. Option used for MAC programming
-no_pci	Do not use the PCI. In this mode, the user can enable the Debug UART for GRC access instead of PCI.
-t <grps/tests>	Disable certain groups/tests (e.g. -t A5).
-T <grps/tests>	Enable certain tests/groups (e.g. -T A5), in addition to the default test matrix. Can be used in conjunction with –none flag to only enable the specified tests/groups.
-cof	Allow tests to continue tests on failure.
-none	Disable all tests.
-help	Display all available options and exit.
-m	Obsolete. Used for manufacturing: prompt users to enter MAC addresses. The option "-autom" can be used to eliminate the need to prompt the user for address for subsequent ports. Options "-m", "-mac", and "-fmac" are mutually exclusive, only one of them can be used at a time.
-mac <MAC>	Obsolete. Used for manufacturing: Use <MAC> as the MAC address to program boards. It's suggested to use "-autom" in conjunction to this to generate MAC address for subsequent port(s). Options "-m", "-mac", and "-fmac" are mutually exclusive, only one of them can be used at a time.
-fmac <MAC file>	Used for manufacturing: Extract MAC address from <MAC file> to program boards. Do not use "-autom" with this. Options "-m", "-mac", and "-fmac" are mutually exclusive, only one of them can be used at a time.

-autom	Obsolete. Used for manufacturing: automatically generate MAC addresses. It cannot be used with "-mac_prof".
-mac_prof <prof_file>	Used for MAC address assignment for a given device. Currently, only "primary" and "BMC" are supported. This option can be used in combination with "-m", "-mac", or "fmac". But it cannot be used with "-autom".
-fnvm <pkgfile>	Used for manufacturing: program various NVRAM components from <pkgfile>.
-cfgchk <chkfile>	Used for manufacturing: compare FW version or CRC value from <chkfile>.
-noidcheck	Disable matching of VID,DID,SVID,and SSID, used with –fnvm option only.
-sn [file name]	Get the serial number of board from file or prompt user
-sn2 [file name]	Get the serial number (second format) from file or prompt user
-sn3 [file name]	Get the serial number (third format) from file or prompt user
-sn4 [file name]	Get the serial number (fourth format) from file or prompt user
-devnrev [file name]	Get the VPD VB field from the file
-vpd	Write VPD fields according to user inputs
-promote	Issue a promote crid command (Only for Thor)
-fru <lock/unlock>	Lock/Unlock FRU for programming
-fru validate <binfile> <specfile>	Program the FRU with the updated binfile and validate the dumped file from FRU
-promote_otp <crid_val> <srt_revid_val> <crt_revid_val>	Obsolete. Promote one or more OTP values (CRID, srt_rev, crt_rev) (Only for Thor)
-get_otp	Obsolete. Read OTP values (CRID, srt_revid, crt_revid)
-gen1	Force to PCIE Gen1 link (with "-fnvm" only)
-blank	Blank the board with the recommended way for the specific device

-lldp_chk	Obsolete. Check LLDP parameters initialized in NVM
-bus <bus>	Only recognize devices from a specified PCIE domain/bus. <bus> is the value of the second part of the PCI bus string. e.g. for PCI bus "00:AF:00:00", the syntax is "-bus 0xAF"
-pvsn_prg [prod]	Provision the board; the "prod" keyword must be provided for manufacturing application (Thor device only, v220 and above)
-cert_tool <tool>	Specify the certificate signing tool and its location; if not specified, the default is "../cert_tool/secureSignSPDM.linux.bin" (Thor device only)

Notes:

Command lines are performed by default on all devices installed on the machine. In the event the devices were selected by –dev or –c, the command line will be performed on the selected devices only.

4.2 Input Files

The following describes files used by some of the command line options.

4.2.1 MAC address range file (-fmac)

This file is used by "-fmac" option. Below is an example content of the file.

```
mac_addr_pref  =  000AF7
mac_addr_start =  8C7F10
mac_addr_end   =  8C7FF0
```

The first entry specifies the first three bytes of the MAC address range. The second entry specifies the last three bytes of the starting MAC address. The third entry specifies the last three bytes of the ending MAC address. The second entry of the file will be updated by Bnxtmt as MAC addresses are consumed. Once the MAC address assignment hits the ending value (third entry), the MAC address generation will stop with an error.

4.2.2 MAC address profile (-mac_prof)

This file is used by "-fmac" option. Below is an example content of the file.

```
# This line is a comment. Make sure it starts with '#'.
# This is a MAC assignment profile file. It defines how

# Must start with version. Only 1.0 is supported for now.
# version = 1.0

# Total number of entries below.
# entries = 2
# ordinal   : the ordinal value used in NVRAM.
# mac_type : either "primary" or "BMC", more can be defined
#            as needed later.
# increment: MAC address value from the base one.

# ordinal = 0
# mac_type = primary
# increment = 0
#
# ordinal = 1
# mac_type = primary
# increment = 1
#
# MAC address value to be advanced for the next chip.
# stride = 2
```

Description…

The following keywords are required.

version	This must be the first entry of the keywords. Only "1.0" is supported.
entries	This specifies the number of MAC address entries (each composed of keywords "ordinal", "mac_type", and "increment". This must be available before starting any MAC entry. bnxtmt needs to know how much memory to reserve to build the internal table.
ordinal	This value corresponds to the ordinal value seen in the "nvm dir" output. This value is loosely related to PCIE function and/or physical network port. The "ordinal", "mac_type", and "increment" are all required for each set of MAC assignment entry.
mac_type	Only "primary" (for L2) and "BMC" are supported. The "ordinal", "mac_type", and "increment" are all required for each set of MAC assignment entry.
increment	The amount of MAC address value to bump from the based one. For example, if the L2 MAC of the first function (strictly speaking, ordinal) is 00:10:18:aa:bb:c0, and the increment value is 5, the MAC address value will be 00:10:18:aa:bb:c5. The "ordinal", "mac_type", and "increment" are all required for each set of MAC assignment entry.
stride	This value specifies the value of MAC address for the next device/board. This usually has the same value as that of "entries". A possible exception is when the number requirement MAC addresses

per device/board is not a power of two, or some MAC addresses need to be reserved but not yet assigned per device/board.

4.2.3 FRU binary template file

This file is used by "-fru validate" option. This file contains the FRU data in binary format. The template file is used to generate a new file with the updated FRU data. This updated file is written to the FRU. The binary template is board specific. Currently only Broadcom generic template is supported.

4.2.4 FRU spec file

This file is used by "-fru validate" option. This file defines the FRU data format in the binary template file. The spec file is board specific. This may be a text file or a csv (Comma Separated Values) file. Currently only Broadcom generic spec file (text format) is supported.

5. Manufacturing – Program Boards

The boards need to be programmed with the correct FW images, nvram configurations, assigned MAC addresses, serial number and etc. The commands listed in this section are for typical usage and there might be some differences from one board SKU to another.

5.1 Blank a board

Usually a board should be blanked in case it has been programmed. The following command is used to blank a board.

Command: ./load.sh(or bnxtmt.efi) -sysop -none -no_swap -blank

5.2 Promote a board

For a controller card(e.g. THOR) that has CRID# seurity option, the boards will come with CRID0 and need to be promoted to CRID1 before production FW is programmed.

Command: ./load.sh(or bnxtmt.efi) -none -promote

5.3 Program a board

The following command will program the nvram image that includes FW images and related nvram configurations. It will also program the MAC addresses, Serial number, and DEVNREV. The required input files are nvram.bin, sample.mac, sample.mpf, sample.sn4, sample.devnrev. The sample input files can be found in subfolder samples_input of bnxtmt package.

Command:

```
./load.sh(or bnxtmt.efi) -sysop -none -no_swap -fnvm nvram.bin -fmac samples_input/sample.mac -mac_prof samples_input/sample.mpf -noreset -sn4 samples_input/sample.sn4 -vpd -noidcheck -gen1 -devnrev samples_input/sample.devnrev
```

5.4 Validate FRU device

Some OCP 3.0 boards have a FRU EEPROM device and bnxtmt. Bnxtmt is not required to program the FRU but it provides a command to validate the FRU EEPROM can be accessed(read and write) correctly. The fru.bin and fru.spec should be specified in BC.

Commands: ./load.sh(or bnxtmt.efi) -none -fru validate fru.bin fru.spec

6. Manufacturing - SysOp Test List

SysOp tests are a separate suite of tests with the emphasis on testing board level component inter-connect. SysOp tests should be done in manufacturing to make sure the boards are manufacutred without connectivity defects.

This suite of tests are divided to different groups. Each test will have a test ID that contains the Group Name and Test Number. For example, the test ID for PCIE test is A1 that me ans the Test 1 in Group A.

What tests will be required to run will be depending on the board design. This should be specified by the BC.

To invoke this suite of tests, the option "-sysop" needs to be specified in the command line.

The table below describes what each testing group includes:

Group Name	Group Description
Group A	Basic tests: This includes PCIE and general hardware register access tests. The test also verifies the multi-host configuration and the clock crystal.
Group B	Memory tests: This group is not executed since it is mainly to exercise various memory blocks, which is a chip level test.
Group C	Traffic tests: This exercises the basic Tx and Rx packet flows via external loopback or PRBS test.
Group D	ChiMP/Primate and misc. tests: This exercises the ChiMP/Primate processor to perform NVRAM accesses and DMA to/from host, etc. Other misc. tests such as checking for certificate provision on Thor devices also fall into this group.
Group E	Wrap up tests: Unlike the previous groups of tests, which are executed together on a single host device interface (groups A-D, one device at a time), this group is invoked only on the last device interface, but the tests in this group exercise all device interfaces together. Currently, only LED test is available in this group. It is arranged this way to facilitate optional visual inspection by test operators.

6.1 Group A – Basic Tests

6.1.1 PCIE Configuration Test (Gen3)

The test ensures the controller is operating at the maximum PCIE speed (currently Gen3) and the maximum PCIE lane capable (currently 8 lanes).

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T A1

6.1.2 PCIE Configuration Test (Gen4)

The test ensures the controller is operating at the maximum PCIE speed (currently Gen4) and the maximum PCIE lane capable.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T A2

6.1.3 Hardware Register Test

The test exercises read and write accesses to some of the internal hardware registers. The hardware block of choice happens to be BMTBM, while any other blocks will work just as well.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T A3

6.1.4 Clock Crystal Test

The test measures the general accuracy of the clock crystal on the board. This is achieved by comparing the clock ticks reported by the chip (which is run by the crystal) and the system time. It is considered passed if the clock ticks fall within an acceptable range (+/- 1%) from the reference tick count.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T A4

6.1.5 Pci Multi-host Tests

The test checks whether all the functions in a Multi Host environment are enumerated properly and are accessible. There are multiple entries in the test matrix for different combination of number of host, number of lanes and PCI generations.

Note: This test should be run on a system which is capable of PCIe segmentation/bifurcation. The PCIe bifurcation must be enabled on the PCIe slot where the card has been installed. This enables the enumeration of all the multi-host devices on the same system with different bus numbers.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T A7

6.2 Group B – Memory Tests

Memory tests are mainly used for chip level diagnostics. Thus, this group does not apply for SysOps usage.

6.3 Group C – Traffic Tests

6.3.1 External loopback Test (built-in)

This test is an External Loopback Test that requires a special built-in FW to be fast booted. This test verifies the link and traffic operations. The test sends a random sequence of data and receives it back. An external loopback connection is required for this test. For a Serdes port, use the external loopback connector. For 10GBase-T ports, use a compliant UTP cable to connect two ports because auto-negotiation is required to establish a link.

This test requires a special testing FW fastboot image that is only supported by Wh+ or Stratus.

This test doesn't require the production Linux driver (bnxt_en) to be loaded and so it can be used in both Linux and UEFI.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T C1

6.3.2 PRBS (Pseudo Random Binary Sequence) Test

This test is applicable to Serdes port only. This test is an alternative to the External Loopback Test. The test sends a random sequence of data and receives it back like the loopback test. The external loopback connector is required for this test. This test verifies the link and traffic operations.

This test doesn't require the production driver (bnxt_en) to be loaded and so it can be used in both Linux and UEFI.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T C2

6.3.3 External Loopback Test (socket-based)

This test is designed for dual port 10GBase-T where the PRBS test is not applied. A compliant UTP cable is needed to connect between the two ports of the controller because auto-negotiation is required to establish a link. Traffic is initiated from one port. The PHY of the other port is put into a remote loopback mode, essentially serving as a loopback plug, to bounce packets back to the initiating port.

This test requires the production Linux driver (bnxt_en) to be loaded in order to execute. Packets are sent and received through the Linux OS network stack.

This test is not supported by bnxtmt lite UEFI.

Command: ./load.sh -sysop -no_swap -none -T C3

6.3.4 Optical Module I2C

The test exercises the I2C interface to the optical cages. A loopback plug or a DAC needs to be inserted, as the test attempts to read content from the EEPROM of the plug/DAC. The test is not applicable on 10GBase-T designs and reported as "skipped".

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T C4

6.3.5 NCSI Loopback Test

The test exercises the NCSI interface. A loopback cable is required to plug into the RMII connector on a PCIe stand-up card or plug into the RJ-45 ports of a Jig card .

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T C5

6.4 Group D – ChiMP/Primate Tests

6.4.1 NVRAM Test

In this test, a piece of diagnostic firmware is written to the ChiMP processor, and that firmware will be invoked to perform access to NVRAM. The accesses are read only since we do not want to shorten the NVRAM life span by erasing pages repeatedly. The test ensures connections between the chip and the NVRAM part are functioning properly.

Since the ChiMP processor and the NVRAM interface are the common blocks of the chip, it is not necessary to execute the same test through multiple host device interfaces. Thus, the test is only performed on device 1 in the SysOps test.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T D1

6.4.2 Host DMA Test

In this test, a piece of diagnostic firmware is written to the ChiMP or Primate processor, and that firmware will be invoked to perform DMA read/write operations, transferring data in/out of the chip from/to the host memory. This creates stress conditions on the PCIE connections to the host. In each operation, data integrity check is performed to ensure no data corruption occurs.

Since the PCIE connections between host memory and the controller are for all PCIE functions, it is not necessary to execute the same test through multiple host device interfaces. Thus, the test is only performed on device 1 in the SysOps test.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T D2 (ChiMP – Wh+ and Stratus)

./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T D3 (Primate - Thor)

6.4.3 Certificate Provisioning Check (Thor only)

This test is primarily a query to the production firmware on whether the board has been provisioned for security applications (specifically SPDM in the current implementation).

The provisioning feature is currently available only on Thor devices. For the same reason, this test is applicable only to Thor devices as well.

Since the provisioning is applicable for the entire device, it is not necessary to execute the same test through multiple host device interfaces. Thus, the test is only performed on device 1 in the SysOps test.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T D5

6.5 Group E – Wrap Up Tests

6.5.1 Crash Dump Check Test

This test ensures that there is no firmware crash during the test procedure. The test looks for valid crash dump information. The existence of a valid crash dump implies a firmware crash during the tests. If the firmware crash is observed during the tests, this test raises a flag. For a successful manufacturing test cycle, this test should not detect any crash dump data.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T E1

6.5.2 LED Test

The test drives each of the LED pins for a fixed amount of time. This is done for all available network ports. Currently, visual inspection is required to verify the proper connection to all the LEDs (and perhaps colors). The visual inspection can be done either manually by human eye or by some optical device looking for specific colors and location.

As the group name implied, such a test is performed toward the end of the entire diagnostic test suite.

The following command will blink the LED one by one.

Command: ./load.sh(or bnxtmt.efi) -sysop -no_swap -none -T E2