DCSG S9500-22XST Test Plan

R1.0

The information contained in this document is considered confidential information and should not be disclosed without prior consent of UfiSpace. This information is considered preliminary and may change in substance or form prior to final product release.

May. 30, 2021

UfiSpace Technology Corporation

Copyright © 2021 by Ufi Space Co., Ltd. All rights reserved

HISTORY

Date	Rev	Author	Summary of Change
2021/05/30	1.0	Roye Shao	Initial version

Table of Terminology

1 INTRODUCTION 6
1.1 Purpose 6
1.2 Scope 6
2 Test items 7
2.1 BMC IPMI function verification 7
2.1.1 IPMI test setup 7
2.1.2 BMC IPMI functions 7
2.1.3 Reboot DUT 7
2.1.4 PSU status function 8
2.1.5 FAN status function 8
2.1.6 System Thermal Temperature Test 9
2.2 SSD, DIMM verification 9
2.2.1 SSD, DIMM access verification 9
2.2.2 SSD, DIMM basic performance test 10
2.2.3 SSD, DIMM specification verification 10
2.3 Switch ports link status verification 11
2.3.1 Port link test setup with single loopback module 11
2.3.2 Port linkup status verification 11
2.3.3 Port present status verification 12
2.4 Switch traffic function verification 12
2.4.1 Line speed test setup with DAC cable 12
2.4.2 Line speed test 13
2.4.3 Snake test setup with DAC cable and traffic generator 13
2.4.4 Snake traffic test 14
2.5 Timing interface protocol verification 15
2.5.1 DPLL output Reference Lock test 16
2.5.2 DPLL input Reference Lock test 17
2.5.3 FPGA input/output Reference Lock test 17
2.5.4 Timestamp verification 18
2.6 LED verification 18
2.6.1 System LED test 19
2.6.2 RJ45 OOB LED test 19
2.6.3 SFP+ LED test 20
2.6.4 QSFP LED test 20
2.6.5 Fabric LED test 21
2.6.6 Beacon (Digital) LED test 21
2.6.7 FAN LED test 22

2.6.8 PSU LED test 22
2.7 USB port verification 23
2.7.1 USB port test 23
2.8 Micro USB and RJ45 serial console verification 23
2.8.1 Micro USB and RJ45 serial console test 23
2.9 OOB management port verification 24
2.9.1 OOB management port test 24
2.10 Reset button verification 25
2.10.1 Reset button test 25
2.11 Internal I2C component verification 25
2.11.1 Internal I2C component test 25
3 TEST MATRIX AND SUMMARY 26
3.1 X86 test coverage 26
3.2 BMC test coverage 27

Table of Terminology

Term	Description
CLI	Command Line Interface
DAC	Direct Attached Cable
DUT	Device Under Test
FRU	Field Replaceable Unit
GPIO	General Purpose Input Output
HW	Hardware
I2C, IIC, I2C	Inter-Integrated Circuit
PSU	Power Supply Unit
QSFP	Quad Small Form-factor Pluggable
QSFPDD	Quad Small Form-factor Pluggable Double Density
SFP+	Small Form-factor Pluggable
SW	Software
UART	Universal Asynchronous Receive and Transmit

1 INTRODUCTION

1.1 Purpose

This document describes the test cases for S9500-22XST product. The software components of S9500-22XST are applications installed in Ubuntu for x86 and applications in BMC software for BMC. The test cases description in this document includes the Test Name of the Test Cases, Test Purpose, Test Configuration, Test Procedure and Expected Result. All test procedures are provided here as guideline for performing software design validation test.

1.2 Scope

The test includes follow parts:

a. BMC IPMI function verification
b. SSD, DIMM verification
c. Switch ports link status verification
d. Switch traffic function verification
e. Timing interface protocol verification
f. LED verification
g. USB port verification
h. Micro USB and RJ45 serial console verification
i. OOB management port verification
j. Reset button verification
k. Internal I2C component verification

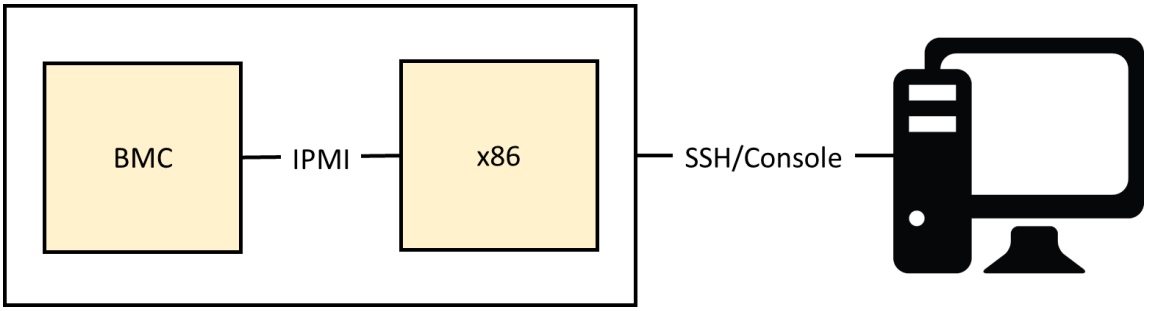
2 Test items

The test cases are designed to verify the major functions which the DUT can work normally as the production specification. These tests are designed to cover as much as possible for the software design and implementation verification.

2.1 BMC IPMI function verification

2.1.1 IPMI test setup

BMC IPMI function is enabled for x86 CPU use. The test scenario is test PC connects to x86 Linux in DUT via console or SSH connection and then use IPMI command to test BMC IPMI functions.



IPMI tools in x86 can be installed with Linux apt utility and BMC IPMI functions is enabled by default.

2.1.2 BMC IPMI functions

BMC supports the below major IPMI functions:

- HW Reboot
- PSU Test PSU Module #1 and #2
- FAN Test in PSU Module #1 and #2
- FAN Test in FAN Module #1, #2, #3 and #4
- System Thermal Temperature Test

2.1.3 Reboot DUT

A. Test Purpose

To verify that the DUT can be reboot via IPMI command.

B. Test Configuration

Install IPMI tool in x86 Linux in DUT. To configure the testing environment as IPMI test setup.

C. Test Procedure and Expected Result

Use "ipmitool power cycle" to reset DUT. Expect that x86 Linux do gracefully shutdown and then reboot normally.

D. Test Result

Pass / NG

2.1.4 PSU status function

A. Test Purpose

To verify that the specified PSU#n of the DUT. BMC software will reply the correct status information: PSU#n PG status / PSU#n presence status

B. Test Configuration

Install IPMI tool in x86 Linux in DUT. To configure the testing environment as IPMI test setup.

C. Test Procedure and Expected Result

1. Make sure the PSU#1 and PSU#2 are attached with power Cable to DUT.

1.1 Execute "ipmitool sdr elist" to check the status shall show PSU#n_Presence="Device Present", PSU#n_POWEROK="Device Enabled"

2. Remove the power cable from PSU#n,

2.1 Execute "ipmitool sdr elist" to check the status shall show PSU#n_Presence="Device Present", PSU#n_POWEROK="Device Disabled"

3. Re-insert the power cable into PSU#n,

3.1 Execute "ipmitool sdr elist" to check the status shall show PSU#n_Presence="Device Present", PSU#n_POWEROK="Device Enabled"

4. Remove the PSU#1 from DUT,

4.1 Execute "ipmitool sdr elist" to check the status shall show PSU1_Presence="Device Absent", PSU1_POWEROK="Device Disabled"

5. Re-insert the power cable into PSU#n,

5.1 Execute "ipmitool sdr elist" to check the status shall show PSU#n_Presence="Device Present", PSU#n_POWEROK="Device Enabled"

D. Test Result

Pass / NG

2.1.5 FAN status function

A. Test Purpose

To verify that the specified FAN#n or FAN/PSU#n of the DUT. BMC software will reply the correct status information.

B. Test Configuration

Install IPMI tool in x86 Linux in DUT. To configure the testing environment as IPMI test setup.

C. Test Procedure and Expected Result

1. Make sure the FAN#1 ~ FAN#6 FAN/PSU#1 and FAN/PSU#2 are attached to DUT.

1.1 Execute "ipmitool sdr elist" and check the FAN_#n and FAN_PSU#n status shall show "ok" status and has RPM rate. FAN#0_Presence shall show "Device Present".

2. Remove FAN#n and FAN/PSU#n from the DUT,

2.1 Execute "ipmitool sdr elist" and check the FAN_#n status shall show "lnr" status, and FAN_PSU#n shall show "ns" status. FAN#0_Presence shall show "Device Absent".

3. Re-insert the FAN#n into DUT,

3.1 Execute "ipmitool sdr elist" and check the FAN_#n and FAN_PSU#n status shall show "ok" status and has RPM rate. FAN#0_Presence shall show "Device Present".

D. Test Result

Pass / NG

2.1.6 System Thermal Temperature Test

A. Test Purpose

To verify that System Thermal Temperature can be detected when fan is removed

B. Test Configuration

Install IPMI tool in x86 Linux in DUT. To configure the testing environment as IPMI test setup.

C. Test Procedure and Expected Result

1. Make sure all fans are attached to DUT.

1.1 Read the System Thermal Temperature with "ipmitool sensor" command

2. Remove all fans from DUT for 30s

2.1 Read the System Thermal Temperature with "ipmitool sensor" command

3. Attach all fans to DUT to stop the case.

D. Test Result

If observe the Temperature at 2.1 is higher than 1.1 which is Pass

2.2 SSD, DIMM verification

For SSD and DIMM, the diagnostics utility is an application running on x86 Linux. It will run the component access testing via the reading and writing process. Meanwhile, will run the basic performance, quality and stress test on it.

- Access test
- Basic performance test: Sysbench (Benchmark Tool)
- Specification verification

2.2.1 SSD, DIMM access verification

A. Test Purpose

To verify the component is accessible normally

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. “flashtest small/large” and “memtest” test command will do the read and write with the small file size and large file size on SSD.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux

2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “flashtest small” test command

DIAG> flashtest small

4. Check the result is pass or not

5. Execute “flashtest large” test command

DIAG> flashtest large

6. Check the result is pass or not

7. Execute “memtest” test command

DIAG> memtest

8. Check the result is pass or not

D. Test Result

Pass / NG

2.2.2 SSD, DIMM basic performance test

A. Test Purpose

To verify the performance is match with specification

B. Test Configuration

Install Sysbench utility in x86 Linux in DUT. Run parameter CPU to verify the CPU performance, run parameter MEMORY to verify memory function test, and run FILEIO to verify the I/O test quality.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux

2. Execute “sysbench --test=cpu run --num-threads=8” in x86 Linux

root@ubuntu~# sysbench --test=cpu run --num-threads=8

3. Execute “sysbench --test=memory --memory-block-size=8K --memory-total-size=10G run” in x86 Linux

root@ubuntu~# sysbench --test=memory --memory-block-size=8K --memory-total-size=10G run

4. Execute “sysbench --test=fileio --num-threads=8 --file-total-size=100G --file-io-mode=sync --file-test-mode=rndrw run” in x86 Linux

root@ubuntu~# sysbench --test=fileio --num-threads=8 --file-total-size=100G --file-io-mode=sync --file-test-mode=rndrw run

5. Check the result is pass or not

D. Test Result

Pass / NG

2.2.3 SSD, DIMM specification verification

A. Test Purpose

To verify the component information and specification

B. Test Configuration

Install “Smartctl” and “dmidecode” utility in x86 Linux in DUT which are the open source tool for read the SSD/DIMM information.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute “smartctl -a /dev/sda” in x86 Linux

root@ubuntu~# smartctl -a /dev/sda

3. Verify the SSD’s vendor, FW version, storage size, and Vendor specific SMART attributes
4. Execute “dmidecode -t 17” in x86 Linux

root@ubuntu~# dmidecode -t 17

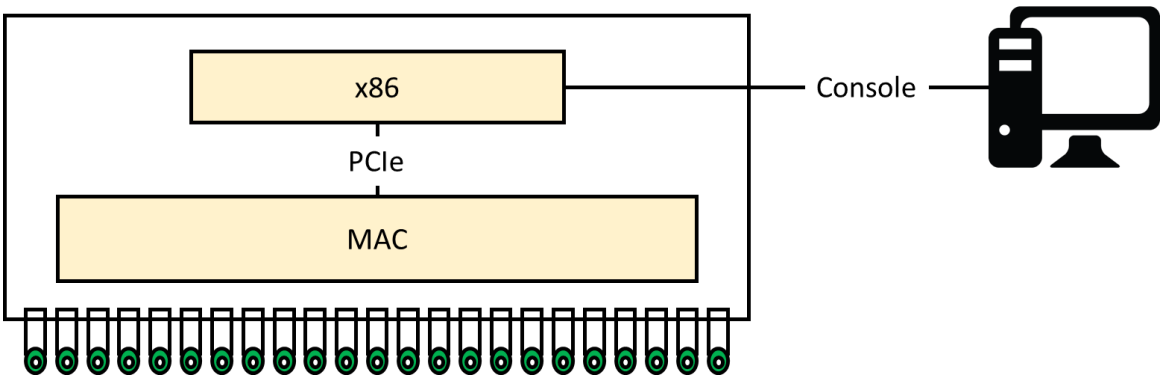
5. Verify the DIMM’s vendor, size, and runtime speed
6. Check the result is pass or not

D. Test Result

Pass / NG

2.3 Switch ports link status verification

2.3.1 Port link test setup with single loopback module



The diagnostics utility is an application running on x86 Linux. When utility runs up, it will initialize MAC and related SDK configuration. The port link test will test port linkup status and port present status.

2.3.2 Port linkup status verification

A. Test Purpose

To verify the port linkup verification with the single loopback module

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. To initialize MAC and related SDK configuration. To plug the single loopback module into every port of the DUT. The “port_status” test command will get the port status from MAC’s register.

C. Test Procedure and Expected Result

1. Make sure all single loopback modules connects well for each port
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “port_status” test commands

DIAG> port_status

4.Check test result if all port links up or not

D. Test Result

Pass / NG

2.3.3 Port present status verification

A. Test Purpose

To verify the port present status verification with the single loopback module

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. To plug the single loopback module into every port of the DUT. The “port_status present 1” test command will get the port status from MAC’s register.

C. Test Procedure and Expected Result

1. Make sure all single loopback modules connects well for each port

2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “port_status present 1” test command

DIAG> port_status present 1

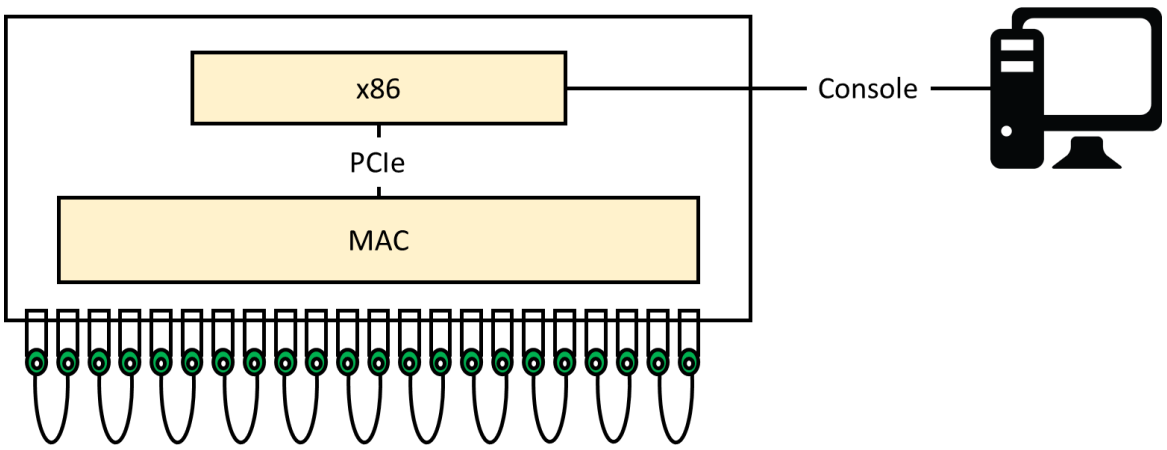
4.Check test result if all port present or not

D. Test Result

Pass / NG

2.4 Switch traffic function verification

2.4.1 Line speed test setup with DAC cable



Line speed test utility is an application running on x86 Linux. When utility runs up, it will initialize MAC and related SDK configuration.

2.4.2 Line speed test

A. Test Purpose

To verify traffic test with DAC cable

B. Test Configuration

Install Linespeed utility in x86 Linux in DUT. To configure the testing environment as Line speed test setup. Linespeed test utility will send fixed number of packets to each port pair and then make traffic loop in the port pair to approach line rate traffic. When test done, it will calculate error packets number and related traffic rate.

C. Test Procedure and Expected Result

1. Make sure all DAC cable connects well for each top and bottom port pair.
2. SSH or console connects to x86 Linux
3. Execute linespeed utility in x86 Lin

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

4. Execute linespeed test command

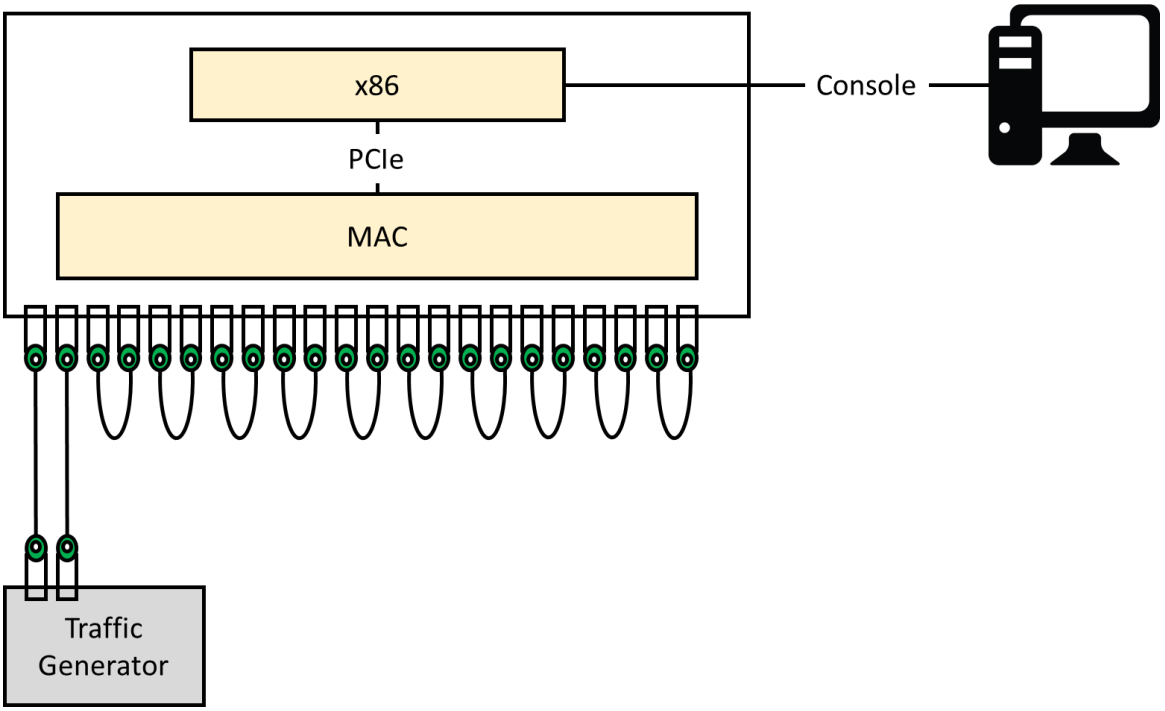
DIAG> linespeed extp 0 29 0x55 1500 30 30

5.Check test result if all pass or CRC error

D. Test Result

If there is no CRC packet and traffic rate can reach 90% port speed, result is Pass. Otherwise, result is NG.

2.4.3 Snake test setup with DAC cable and traffic generator



The snake test utility is an application running on x86 Linux. When utility runs up, it will initialize MAC and related SDK configuration.

2.4.4 Snake traffic test

A. Test Purpose

To verify traffic test with DAC cable

B. Test Configuration

Install snake test utility in x86 Linux in DUT. To configure the testing environment as snake test setup. Snake test utility will make all front switch ports as a chain.

C. Test Procedure and Expected Result

1. Make sure all DAC cable connects well for each top and bottom port pair

And connect one port pair with two ports of traffic generator

2. SSH or console connects to x86 Linux

3. Execute snake test utility in x86 Linux

root@ubuntu~# cd /root/DIAG/DNX_QAX_DIAG

root@ubuntu~# ./runbcm.sh ufi

DIAG>

4. Execute snake test command

DIAG> snakesetup 0 29

5.Send traffic from traffic generator and shall also receive traffic in generator. Check if traffic received by traffic generator has CRC packets or not

D. Test Result.

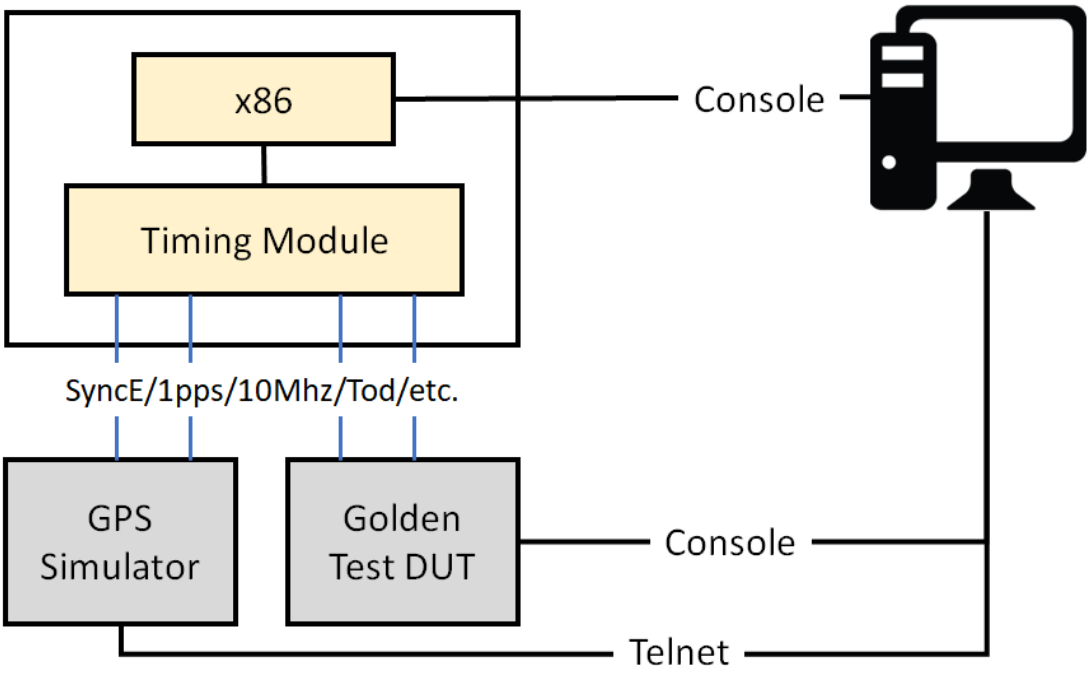
If there is no any CRC or drop packets, result is Pass. Otherwise, result is NG.

2.5 Timing interface protocol verification

The Timing interface protocol verification will verify the DPLL/FPGA/etc. module frequency accuracy, Jitter/Wander generation and tolerance. And the Short/Long term transient response (holdover). The part of the timing test as follow table:

Module/Protocol	Inspection	Expected Result
OCXO (Clock Source)	DPLL Lock test	1) DPLL Locked
SyncE Verification	Main Chip (BCM88270)	1) DPLL Locked
	10G PHY (BCM82759)	1) DPLL Locked
	1G PHY (BCM54140)	1) DPLL Locked
1PPS Verification	SMB (Media)	1) DPLL Locked
	RJ45 (TOD Media)	1) DPLL Locked
	GNSS (GPS)	1) DPLL Locked
10Mhz Verification	SMB (Media)	1) Read/Write data test
	RJ45 (TOD Media)	1) DPLL Locked
	GNSS (GPS)	1) DPLL Locked
ToD Verification	RJ45 (TOD Media)	1) DPLL Locked
	GNSS (GPS)	1) FPGA Locked
	PKT-TOD (From CPU)	1) FPGA Locked
T1/E1 Verification	RJ45 (BITS Media)	1) DPLL Locked
Timestamp	1G PHY (BCM54140)	1) Pattern Match

Verify the Timing related interfaces, need the GPS simulator and the Golden DUT which can verify the signal with the DUT, the topology as the follow picture.



2.5.1 DPLL output Reference Lock test

A. Test Purpose

To verify the DPLL output reference lock from the Golden DUT with SMB-1pps, SMB-10MHz, TOD and etc.

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. After initialize DPLL and FPGA, for example, execute “idttest lock SMA-1PPS” test command to verify the reference lock.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “idttest lock SMA-1PPS” test command

DIAG> idttest lock SMA-1PPS

4. Execute “idttest lock SMA-10Mhz” test command

DIAG> timing/ntm dpll lock SMA-10Mhz

5. Execute “timing/ntm dpll lock output-tod-1pps” test command

DIAG> timing/ntm dpll lock output-tod-1pps

(and so on)

6. Check the reference lock result

D. Test Result

Pass / NG

2.5.2 DPLL input Reference Lock test

A. Test Purpose

To verify the DPLL input reference lock with OCXO, GNSS-1PPS, GNSS-10MHz, GNSS-TOD, BITS, SMB-10MHz, SMB-1pps, TOD, SyncE and etc.

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. After initialize DPLL and FPGA, for example, execute “timing/ntm dpll lock OCXO” test command to verify the reference lock.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “timing/ntm dpll lock OCXO” test command

DIAG> timing/ntm dpll lock OCXO

4. Execute “timing/ntm dpll lock GNSS-1PPS” test command

DIAG> timing/ntm dpll lock GNSS-1PPS

5. Execute “timing/ntm dpll lock GNSS-10MHz” test command

DIAG> timing/ntm dpll lock GNSS-10MHz

(and so on)

6. Check the reference lock result

D. Test Result

Pass / NG

2.5.3 FPGA input/output Reference Lock test

A. Test Purpose

To verify the FPGA input/output reference lock with RJ45-TOD, GNSS-TOD and etc.

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. After initialize DPLL and FPGA, for example, execute “timing/ntm fpga lock RJ45-TOD” test command to verify the reference lock.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “timing/ntm fpga lock RJ45-TOD” test command

DIAG> timing/ntm fpga lock RJ45-TOD

4. Execute “timing/ntm fpga lock GNSS-TOD” test command

DIAG> timing/ntm fpga lock GNSS-TOD

(and so on)

5. Check the reference lock result

D. Test Result

Pass / NG

2.5.4 Timestamp verification

A. Test Purpose

To verify the timestamp protocol with CHIP behavior.

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. After initialize DPLL and FPGA, for example, execute "timing/cint/bcm_ts.c" test command to verify the reference lock.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute "timing/cint/bcm_ts.c" test command

DIAG> timing/cint/bcm_ts.c (and so on)

4. Check the reference lock result

D. Test Result

Pass / NG

2.6 LED verification

The LED verification has the System LED, RJ45 OOB port LED, SPF+ LED, QSFP LED, Fabric LED, digital beacon LED, FAN module LED and PSU module LED. The LED behavior of the color as follow table:

LED	State	Expected Result
System	Normal	Green
	Fault	Amber
RJ45 OOB	Link up	Green
	Transmitting/Receiving data	Blinking
	No connection	Off
SPF+	Link up	Green
	Transmitting/Receiving data	Blinking
	No connection	Off
QSFP	Link up	Green

	Transmitting/Receiving data	Blinking
	No connection	Off
Fabric	Link up	Green
	Link down	Red/Blue
	Transmitting/Receiving data	Blinking
	No connection	Off
Beacon (Digital)	Display Msg	Blue number
FAN Module	Normal operation	Green
	Fault	Amber
PSU Module	Normal operation	Green
	Fault	Red
	Voltage Fail	Blinking Red

2.6.1 System LED test

E. Test Purpose

To verify system LED display which has PSU status, Fan status and system power good status.

F. Test Configuration

Install diagnostics utility in x86 Linux in DUT. The “ledtest sys” test command will do the system LED test. First test will force to display green of the system LED, second will force to display amber color and final will force to turn off the LED.

G. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “ledtest sys” test command

DIAG> ledtest sys

4. Check the first test result: All green color
5. Check the second test result: All amber color
6. Check the third test result: All off

H. Test Result

Pass / NG

2.6.2 RJ45 OOB LED test

A. Test Purpose

To verify RJ45 OOB management port LED display via plug the network cable with traffic

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. To plug the network cable connect between the OOB management

port and the network switch which will get the IP address. The “ifconfig” command will check the network connection of the OOB management port. And the “ping” test command will creat the traffic with the packets.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute “ifconfig” utility in x86 Linux

root@ubuntu~# ifconfig

3. Check the OOB management network connection
4. Execute “ping” test command

root@ubuntu~# ping xx.xx.xx.xx -c 5 -s 1024

5. Check the “ping” result and the OOB LED behavior

D. Test Result

Pass / NG

2.6.3 SFP+ LED test

A. Test Purpose

To verify switch SFP+ LED display

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. The “kr_test” test command will do the SFP+ traffic test which will deliver the packets.

C. Test Procedure and Expected Result

1. Make sure SFP+ DAC cable connects well
2. SSH or console connects to x86 Linux
3. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

4. Execute “kr_test” test command

DIAG> kr_test

5. Check the OOB management LED behavior while traffic test

D. Test Result

Pass / NG

2.6.4 QSFP LED test

A. Test Purpose

To verify switch QSFP LED display

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. The “ledtest port” test command will do the QSFP LED test. First test will force to display green of all the port LED, second will force to display amber color and final will force to blink the LED with green and amber.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “ledtest port” test command

DIAG> ledtest port

4. Check the first test result: All green color
5. Check the second test result: All amber color
6. Check the third test result: All blink with green and amber

D. Test Result

Pass / NG

2.6.5 Fabric LED test

A. Test Purpose

To verify switch Fabric LED display

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. The “ledtest fabric” test command will do the QSFP LED test. First test will force to display green of all the port LED, second will force to display red color, third will force to display blue color and final will force to blink the LED with green, red and blue.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “ledtest fabric” test command

DIAG> ledtest fabric

4. Check the first test result: All green color
5. Check the second test result: All red color
6. Check the second test result: All blue color
7. Check the third test result: All blink with green, red and blue

D. Test Result

Pass / NG

2.6.6 Beacon (Digital) LED test

A. Test Purpose

To verify beacon (digital) 8-bit LED display which can display the alphabet or number.

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. The “ledtest beacon” test command will do the beacon LED test. Will force to display “88” to make sure all the tubes’ behavior.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

DIAG>

3. Execute “ledtest beacon” test command

DIAG> ledtest beacon

4. Check the beacon LED behavior

D. Test Result

Pass / NG

2.6.7 FAN LED test

A. Test Purpose

To verify FAN#n LED behavior

B. Test Configuration

To configure the testing environment as FAN port LED test setup.

C. Test Procedure and Expected Result

1. Power up device.
2. Check all FAN LED is green
3. Plug out FAN#n, Check FAN#n LED is off
4. Re-plug in FAN#n, Check FAN#n LED is green
5. Use fan stopper to stop FAN#n, Check FAN#n LED is blinking amber
6. Remove fan stopper, Check FAN#n LED is green

D. Test Result

Pass / NG

2.6.8 PSU LED test

A. Test Purpose

To do PSU#n LED behavior check

B. Test Configuration

To configure the testing environment as PSU port LED test setup.

C. Test Procedure and Expected Result

1. Power up device.
2. Check all PSU LED is green
3. Plug out PSU#n, Check PSU#n LED is off

4. Re-plug in PSU#n, Check PSU#n LED is green
5. Plug out the power cord of PSU#n, Check PSU#n LED is blinking red
6. Re-plug the power cord of PSU#n, Check PSU#n LED is green

D. Test Result

Pass / NG

2.7 USB port verification

2.7.1 USB port test

A. Test Purpose

To verify the USB flash drive can be detected and read/write works properly

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. The “usbtest” test command will do the USB access read/write test.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

3. Make sure USB dongle connects well
4. Execute “usbtest” test command

DIAG> usbtest

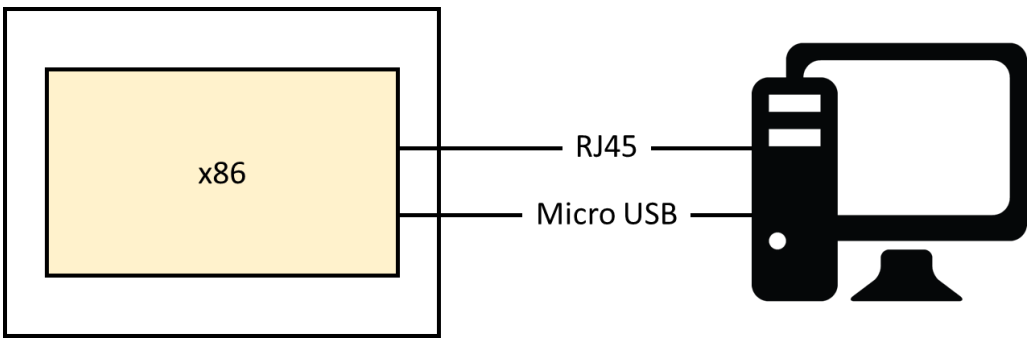
5. Check the USB test result

D. Test Result

Pass / NG

2.8 Micro USB and RJ45 serial console verification

2.8.1 Micro USB and RJ45 serial console test



A. Test Purpose

To verify the Micro USB drive and RJ45 can access the DUT with correct baud rate

B. Test Configuration

First, plug the RJ45 LAN cable between the DUT and the PC and execute terminal application to access the DUT. Second, replace the RJ45 LAN cable with the micro USB cable.

C. Test Procedure and Expected Result

1. Plug RJ45 LAN cable
2. Do console connects to x86 Linux
3. Execute CLI interactive command in x86 Linux
4. Make sure console works well
5. Plug out RJ45 cable and plug Micro USB cable
6. Do console connects to x86 Linux
7. Execute CLI interactive command in x86 Linux
8. Make sure console works well

D. Test Result

Pass / NG

2.9 OOB management port verification

2.9.1 OOB management port test

A. Test Purpose

To verify the OOB management port can work properly via packet traffic with 10Mbps, 100Mbps and 1Gbps

B. Test Configuration



First, Install “ethtool” utility in x86 Linux in DUT. Second, plug the network cable between the DUT and the PC then configurate PC’s network setting to 10Mbps/100Mbps/1Gbps. Third, verify the DUT network information via command “ethtool”.

C. Test Procedure and Expected Result

1. Plug network LAN cable
2. SSH or console connects to x86 Linux

3. Configurate the PC's network interface speed to be 10Mbps
4. Execute diagnostics utility in x86 Linux

root@ubuntu~# ethtool eth0

5. Check the "Speed" and "Duplex" result of the output is 10Mbps/Full or not
6. Configurate the PC's network interface speed to be 100Mbps
7. Execute diagnostics utility in x86 Linux

root@ubuntu~# ethtool eth0

8. Check the "Speed" and "Duplex" result of the output is 100Mbps/Full or not

Configurate the PC's network interface speed to be 1Gbps

9. Execute diagnostics utility in x86 Linux

root@ubuntu~# ethtool eth0

10. Check the "Speed" and "Duplex" result of the output is 1Gbps/Full or not

D. Test Result

Pass / NG

2.10 Reset button verification

2.10.1 Reset button test

A. Test Purpose

To verify the "Reset" button of the DUT can press and reset the system properly

B. Test Configuration

Login the x86 Linux and use the fixture to press the "Reset" button

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Press the "Reset" button
3. Check the system is reboot/reset or not

D. Test Result

Pass / NG

2.11 Internal I2C component verification

2.11.1 Internal I2C component test

A. Test Purpose

To verify the I2C component can be read/write via i2c bus interface in the DUT. I2c component including EEPROM, MUX, IOEXP, CPLD and Sensors.

B. Test Configuration

Install diagnostics utility in x86 Linux in DUT. The "i2ctest all all" test command will do the i2c component access read/write test.

C. Test Procedure and Expected Result

1. SSH or console connects to x86 Linux
2. Execute diagnostics utility in x86 Linux

root@ubuntu~# cd /root/DIAG/

root@ubuntu~# ./runbcm.sh ufi

3. Execute “i2ctest all all” test command

DIAG> i2ctest all all

5. Check the test result

D. Test Result

Pass / NG

3 TEST MATRIX AND SUMMARY

Test matrix has two parts to show the x86/BMC domain test coverage

3.1 X86 test coverage

		CPU & BMC				BRCM Ethernet Switch					
	Test Scope	BMC (AST2400)	Main Processor	DDR SDRAM	flash	BRCM Ethernet Switch	Ethernet Switch LED	QSFP28 / QSFP-DD Interface	BRCM (Gearbox)	BRCM (Retimer)	BRCM (OP2)
x86	DDR Test		●	●							
	Flash Test		●		●						
	CPLD Test		●								
	EEPROM Test		●								
	I2C Test / IOexp test		●					●			
	LED Test		●			●	●				
	USB Test		●								
	Voltage Test		●								
	Fan Test		●								
	Temperature Test		●								
	Power Supply Test		●								
	Mgmt. Port Test		●								
	Interrupt Test	●	●			●		●			
	Present Test		●					●			
	PCIe Test		●			●					●
	Reset Test		●			●		●			
	SW Traffic Loopback Test		●			●		●	●	●	
	SW Line Speed Test		●			●		●	●	●	●
	TPM Test		●								
	MicroUsb console test		●								

		Peripheral Devices & Interfaces													
	Test Scope	RJ45 Mgmt. Port (I210)	SFP+ interface	CPLD	PSU	Power Management IC	Temperature Sensors	UCD90124A	FAN Trays	I2C MUX / IO expander	EEPROM(Sys, Fan, PSU, SFP, QSFP, QSFPDD)	TPM (SLB9670)	USB / Micro USB	RJ45 Console	LEDs
x86	DDR Test													●	
	Flash Test													●	
	CPLD Test			●										●	
	EEPROM Test										●			●	
	I2C Test / IOexp test			●	●	●	●	●	●	●	●			●	
	LED Test	●												●	●
	USB Test												●	●	
	Voltage Test				●	●		●		●				●	
	Fan Test							●	●	●				●	
	Temperature Test						●			●				●	
	Power Supply Test			●	●					●				●	
	Mgmt. Port Test	●												●	
	Interrupt Test		●		●			●	●	●		●		●	
	Present Test		●	●	●				●	●				●	
	PCIe Test	●												●	
	Reset Test									●		●		●	
	SW Traffic Loopback Test													●	
	SW Line Speed Test													●	
	TPM Test											●		●	
	MicroUsb console test												●		

3.2 BMC test coverage

		CPU & BMC				BRCM Ethernet Switch					
	Test Scope	BMC (AST2400)	Main Processor	DDR SDRAM	flash	BRCM Ethernet Switch	Ethernet Switch LED	QSFP28 / QSFP-DD Interface	BRCM (Gearbox)	BRCM (Retimer)	BRCM (OP2)
BMC	DDR Test	●		●							
	Flash Test	●			●						
	I2C Test / IOexp test	●	●								
	Fan Test	●									
	Mgmt. Port Test	●									
	LPC Test	●	●								
	Power Supply Test	●									
	Temperature Test	●	●								
	Watchdog Test	●	●								
	Interrupt Test	●									
	FRU info Test	●									
	Reset Test	●	●								

		Peripheral Devices & Interfaces													
	Test Scope	RJ45 Mgmt. Port (I210)	SFP+ interface	CPLD	PSU	Power Management IC	Temperature Sensors	UCD90124A	FAN Trays	I2C MUX / IO expander	EEPROM(Sys, Fan, PSU, SFP, QSFP, QSFPDD)	TPM (SLB9670)	USB / Micro USB	RJ45 Console	LEDs
BMC	DDR Test													●	
	Flash Test												●	●	
	I2C Test / IOexp test			●	●	●	●	●	●	●	●			●	
	Fan Test								●	●				●	●
	Mgmt. Port Test	●												●	
	LPC Test													●	
	Power Supply Test				●					●				●	
	Temperature Test				●		●			●				●	
	Watchdog Test													●	
	Interrupt Test				●		●	●	●					●	
	FRU info Test										●			●	
	Reset Test			●				●						●	