



Validation Report

Project Title:

Shikhara SoC Firmware

Validation

Prepared by:

MosChip Semiconductor Technology Ltd
Plot No. 83 & 84, 2nd Floor, Punnaiah plaza
Road No. 2, Banjara Hills,
Hyderabad - 500034, India.
Tel: +91-40-2337-9440, 6622-9292
Fax: +91-40-2337-9439, 6622-9393



Revision History (Revision Changes Date By)

Revision	Changes	Date	Prepared by
1.0	First version	26 th Oct 2017	Moschip Semiconductors

Table of Contents

1.	ABBREVIATIONS	5
2.	INTRODUCTION	6
2.1	Shikhara SoC	6
2.1.1	Shikhara SoC Architecture Diagram	7
2.2	Shikhara_SoC Address Map	7
3.	BUILDING U-BOOT FOR SHIKHARA	11
3.1	Building U-boot For MALI	11
4.	Shikhara SoC (ASIC) Boot modes and bootstraps:	13
4.1	Bootting from JTAG	13
4.1.1	Connecting DSTREAM to Shikhara Target Board	13
4.1.2	Creating a new platform configuration	14
4.1.3	Creating Project in Eclipse	18
4.2	Bootting from NOR	22
4.2.1	Bootting from NOR Production mode:	22
	File Transfer from Host PC using minicom xmodem.....	24
4.2.2	Bootting from NOR BootRom Mode	26
	File Transfer from Host PC using minicom xmodem.....	27
4.3	Memory Arrangement/Allotment for the IP's	30
5.	U-BOOT BASED VALIDATION TEST REPORT	32
5.1	TEST CASES FOR MEMORY VALIDATION	32
5.1.1	Ex-SRAM Test: Write & Read Operations On External SRAM.....	32
5.1.2	NOR Test: Write & Read Operations On NOR	32
5.1.3	NAND Validation	33
5.1.4	DDR Test: Write & Read Operations On DDR	33
5.2	TEST CASES FOR HIGH SPEED PERIPHERALS	35
5.2.1	SD / MMC Validation.....	35
5.2.2	USB Host Validation	38
5.2.3	USB Device Validation.....	43
5.2.4	DMA Validation.....	46
5.3	TEST CASES FOR LOW-SPEED PERIPHERALS	48
5.3.1	UART Validation	48
5.3.2	I2C Validation.....	49
5.3.3	RTC Validation	53
5.3.4	I2S Validation	53
5.3.5	CAN Validation	58
5.3.6	KMI (PS2) Validation.....	60
5.3.7	SPI Validation	62
5.4	TEST CASES FOR DISPLAY PERIPHERALS	65
5.4.1	HDMI Validation	65
5.4.2	CSI Validation	71
5.4.3	DSI Validation	75
5.4.4	LCD Validation.....	77
5.5	TEST CASES FOR H/W ACCELERATED PERIPHERALS	79

5.5.1	GPU Validation.....	79
5.5.2	AV417 Video Subsystem Validation.....	109
5.6	TEST CASES FOR MISCELLANEOUS PERIPHERALS	117
5.6.1	GPIO Validation	117
5.6.2	Timer Validation	119
5.6.3	GNSS Validation	121
5.6.4	TZASC Validation	125
6.	U-BOOT VALIDATION COMMANDS SUMMARY	126
APPENDIX - I		130
➤	ENVIRONMENT VARIABLE SETTING IN U-BOOT	130
APPENDIX - II.....		131
➤	U-Boot Modifications:	131
APPENDIX – III.....		138
➤	Porting U-Boot to Shikhara board	138

1. ABBREVIATIONS

AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
BT	Bluetooth
CLCD	Colour Liquid Crystal Display
CPU	Central Processing Unit
CSI	Camera Serial Interface
DDR	Double Data Rate
DMA	Direct Memory Access
DRD	Dual-Role-Device
DSI	Display Serial Interface
DT	Dual Timer
GIC	Generic Interrupt Controller
GNSS	Global Navigation Satellite System
GP	Geometry Processor
GPIO	General Purpose Input Output
GPU	Graphics Processing Unit
HDMI	High Definition Multimedia Interface
I2C	Inter Integrated Circuit
I2S	Integrated Interchip Sound (Inter- IC Sound)
IP	Intellectual Property
KMI	Keyboard Mouse Interface
L2CC	L2 Cache Controller
LPDDR	Low Power Double Data Rate
MIPI	Mobile Industry Processor Interface
PP	Pixel Processor
RTC	Real Time Clock
SD-MMC	Secure Digital – Multi Media Card
SMC	Static Memory Controller
SoC	System on chip
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver and Transmitter
USB	Universal Serial Bus
WDT	Watch Dog Timer

2. INTRODUCTION

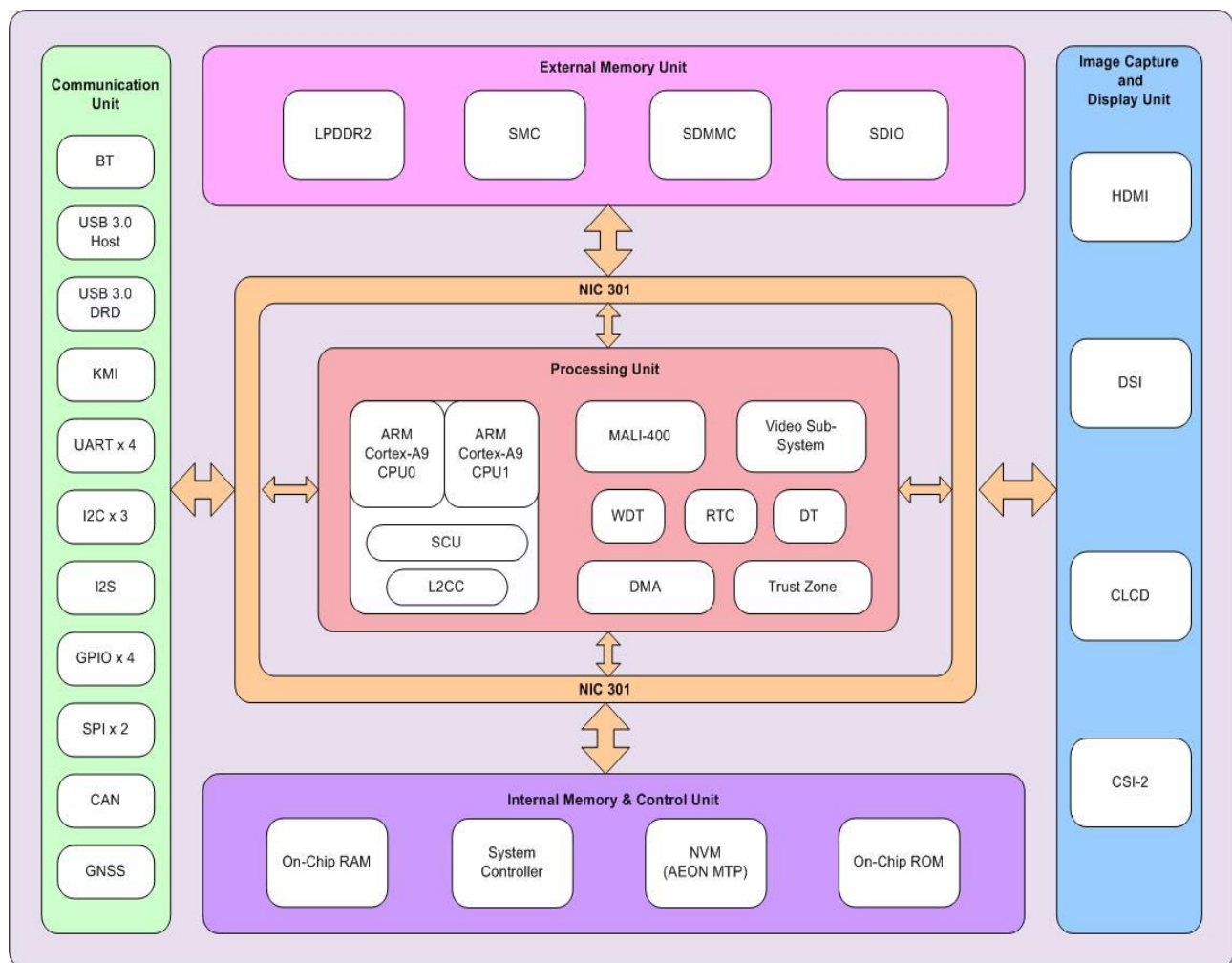
This document defines and presents reports of firmware related test cases used for SHIKHARA SoC Validation.

2.1 Shikhara SoC

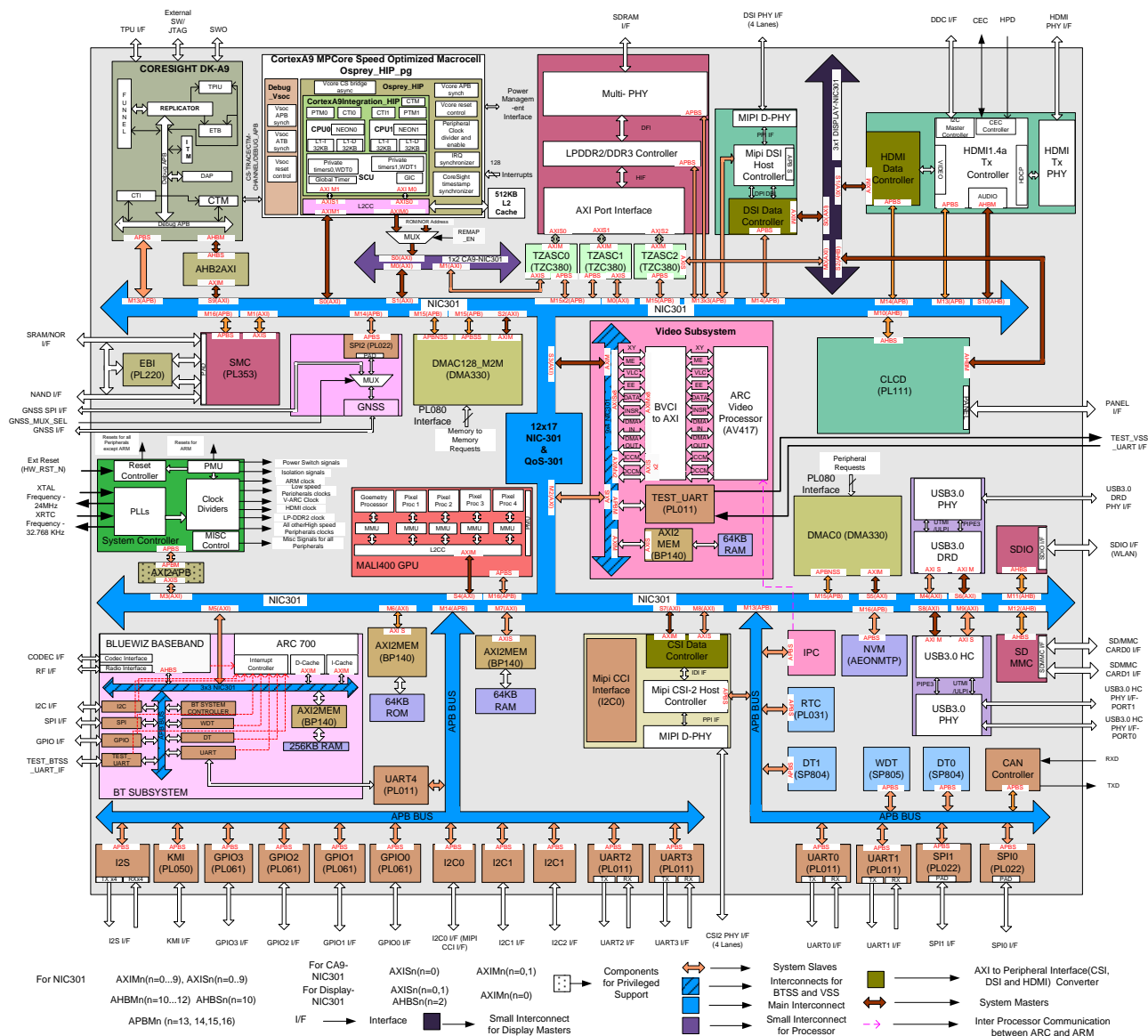
Shikhara SoC architecture is developed for ANURAG's using ARM Cortex A9 MPcore, MALI-400 GPU and AV417 video processor cores along with various system peripherals, connectivity IP's, Memory Sub System.

Shikhara SoC consists of 64KB ROM (Read only memory) and 64KB SRAM (Static RAM) as internal memory.

The following SoC Block diagram illustrates the various interfaces supported.



2.1.1 Shikhara SoC Architecture Diagram



2.2 Shikhara_SoC Address Map

The following Table presents Shikhara_SoC Address Map.

SL No	Sub-System	IP Name	Size	Start Address	End Address
1	LPDDR2/DDR3	LPDDR2/DDR3	2GB	0x0000_0000	0x7FFF_FFFF
2	RESERVED	RESERVED	1GB	0x8000_0000	0xBFFF_FFFF
3	Static Memory ⁽⁴⁾	NOR_MEM_0	64MB	0xC000_0000	0xC3FF_FFFF
4		NOR_MEM_1	64MB	0xC400_0000	0xC7FF_FFFF
5		NOR_MEM_2	64MB	0xC800_0000	0xCBFF_FFFF
6		NOR_MEM_3	64MB	0xCC00_0000	0xCFFF_FFFF
7		NAND_MEM_0	16MB	0xD000_0000	0xD0FF_FFFF
8		NAND_MEM_1	16MB	0xD100_0000	0xD1FF_FFFF
9		NAND_MEM_2	16MB	0xD200_0000	0xD2FF_FFFF
10		NAND_MEM_3	16MB	0xD300_0000	0xD3FF_FFFF
11	ARC-VIDEO SUB SYS	ICCM	16KB	0xD400_0000	0xD400_3FFF
12		DCCM	16KB	0xD400_4000	0xD400_7FFF
13		TEST_UART	4KB	0xD400_8000	0xD400_8FFF
14		RESERVED	1948KB	0xD400_9000	0xD41E_FFFF
15		AV-417 ON-CHIP RAM	64KB	0xD41F_0000	0xD41F_FFFF
16	AXI SUB SYS	NIC-301	1MB	0xD420_0000	0xD42F_FFFF
17		USB 3.0 Host controller	1MB	0xD430_0000	0xD43F_FFFF
18		USB 3.0 DRD controller	1MB	0xD440_0000	0xD44F_FFFF
19	ARC-BT SYS	Bluetooth ON-CHIP RAM	256KB	0xD450_0000	0xD453_FFFF
20		RESERVED	4KB	0xD454_0000	0xD454_0FFF
21		BASE-BAND - CONTROLLER	4KB	0xD454_1000	0xD454_1FFF
22		SYS CONTROLLER	4KB	0xD454_2000	0xD454_2FFF
23		TIMER	4KB	0xD454_3000	0xD454_3FFF
24		WDT	4KB	0xD454_4000	0xD454_4FFF
25		RESERVED	4KB	0xD454_5000	0xD454_5FFF
26		I2C	4KB	0xD454_6000	0xD454_6FFF
27		SPI	4KB	0xD454_7000	0xD454_7FFF
28		GPIO	4KB	0xD454_8000	0xD454_8FFF
29		UART	4KB	0xD454_9000	0xD454_9FFF
30		TEST_UART	4KB	0xD454_A000	0xD454_AFFF
31	RESERVED	RESERVED	32KB	0xD454_B000	0xD455_2FFF
32	AHB SUB SYS	RESERVED	4KB	0xD455_3000	0xD455_3FFF
33		CLCD	4KB	0xD455_4000	0xD455_4FFF
34		SD/MMC	4KB	0xD455_5000	0xD455_5FFF
35		SDIO	4KB	0xD455_6000	0xD455_6FFF
36		RESERVED	4KB	0xD455_7000	0xD455_7FFF
37	RESERVED	RESERVED	32KB	0xD455_8000	0xD455_FFFF
38	APB SUB SYS	GNSS	4KB	0xD456_0000	0xD456_0FFF

39		Sys Controller	4KB	0xD456_1000	0xD456_1FFF
40		MIPI_DSI_CTRL	4KB	0xD456_2000	0xD456_2FFF
41		MIPI_CSI_CTRL	4KB	0xD456_3000	0xD456_3FFF
42		RESERVED	4KB	0xD456_4000	0xD456_4FFF
43		DMAC-0_NS	4KB	0xD456_5000	0xD456_5FFF
44		RESERVED	8KB	0xD456_6000	0xD456_7FFF
45		DMAC128_M2M_S	4KB	0xD456_8000	0xD456_8FFF
46		DMAC128_M2M_NS	4KB	0xD456_9000	0xD456_9FFF
47		RESERVED	4KB	0xD456_A000	0xD456_AFFF
48		TZASC-0	4KB	0xD456_B000	0xD456_BFFF
49		TZASC-1	4KB	0xD456_C000	0xD456_CFFF
50		TZASC-2	4KB	0xD456_D000	0xD456_DFFF
51		DT-0	4KB	0xD456_E000	0xD456_EFFF
52		DT-1	4KB	0xD456_F000	0xD456_FFFF
53		RESERVED	4KB	0xD457_0000	0xD457_0FFF
54		WDT	4KB	0xD457_1000	0xD457_1FFF
55		SPI-0	4KB	0xD457_2000	0xD457_2FFF
56		SPI-1	4KB	0xD457_3000	0xD457_3FFF
57		RESERVED	4KB	0xD457_4000	0xD457_4FFF
58		KMI	4KB	0xD457_5000	0xD457_5FFF
59		GPIO-0	4KB	0xD457_6000	0xD457_6FFF
60		GPIO-1	4KB	0xD457_7000	0xD457_7FFF
61		GPIO-2	4KB	0xD457_8000	0xD457_8FFF
62		GPIO-3	4KB	0xD457_9000	0xD457_9FFF
63		RESERVED	4KB	0xD457_A000	0xD457_AFFF
64		UART-0	4KB	0xD457_B000	0xD457_BFFF
65		UART-1	4KB	0xD457_C000	0xD457_CFFF
66		UART-2	4KB	0xD457_D000	0xD457_DFFF
67		UART-3	4KB	0xD457_E000	0xD457_EFFF
68		UART-4	4KB	0xD457_F000	0xD457_FFFF
69		RESERVED	4KB	0xD458_0000	0xD458_0FFF
70		I2S	4KB	0xD458_1000	0xD458_1FFF
71		I2C-0	4KB	0xD458_2000	0xD458_2FFF
72		I2C-1	4KB	0xD458_3000	0xD458_3FFF
73		I2C-2	4KB	0xD458_4000	0xD458_4FFF
74		RESERVED	8KB	0xD458_5000	0xD458_6FFF
75		CAN	4KB	0xD458_7000	0xD458_7FFF
76		HDMI_CTRL	32KB	0xD458_8000	0xD458_FFFF
77		MALI 400	64KB	0xD459_0000	0xD459_FFFF
78		IPC	4KB	0xD45A_0000	0xD45A_0FFF
79		RESERVED	4KB	0xD45A_1000	0xD45A_1FFF
80		SMC	4KB	0xD45A_2000	0xD45A_2FFF
81		RTC	4KB	0xD45A_3000	0xD45A_3FFF
82		LPDDR2/DDR3-CTRL	4KB	0xD45A_4000	0xD45A_4FFF
83		LPDDR2/DDR3-PHY	4KB	0xD45A_5000	0xD45A_5FFF

84		HDMI_Bridge	4KB	0xD45A_6000	0xD45A_6FFF
85		DSI_Bridge	4KB	0xD45A_7000	0xD45A_7FFF
86		CSI_Bridge	4KB	0xD45A_8000	0xD45A_8FFF
87		NVM_CTRL	4KB	0xD45A_9000	0xD45A_9FFF
88	RESERVED	RESERVED	24KB	0xD45A_A000	0xD45A_FFFF
89	Cortex A9 On-Chip RAM	Cortex A9 On-Chip RAM	64KB	0xD45B_0000	0xD45B_FFFF
90	Core-Sight	DAP Table	4KB	0xD45C_0000	0xD45C_0FFF
91		ETB	4KB	0xD45C_1000	0xD45C_1FFF
92		CTI	4KB	0xD45C_2000	0xD45C_2FFF
93		TPIU	4KB	0xD45C_3000	0xD45C_3FFF
94		Funnel	4KB	0xD45C_4000	0xD45C_4FFF
95		ITM	4KB	0xD45C_5000	0xD45C_5FFF
96		SWO ⁽¹⁾	4KB	0xD45C_6000	0xD45C_6FFF
97		RESERVED	996KB	0xD45C_7000	0xD46B_FFFF
98		Cortex A9 ROM Table	4KB	0xD46C_0000	0xD46C_0FFF
99		RESERVED	60KB	0xD46C_1000	0xD46C_FFFF
100		CPU0 DBG	4KB	0xD46D_0000	0xD46D_0FFF
101		CPU0 PMU	4KB	0xD46D_1000	0xD46D_1FFF
102		CPU1 DBG	4KB	0xD46D_2000	0xD46D_2FFF
103		CPU1 PMU	4KB	0xD46D_3000	0xD46D_3FFF
104		RESERVED	16KB	0xD46D_4000	0xD46D_7FFF
105		CPU0 CTI	4KB	0xD46D_8000	0xD46D_8FFF
106		CPU1 CTI	4KB	0xD46D_9000	0xD46D_9FFF
107		RESERVED	8KB	0xD46D_A000	0xD46D_BFFF
108		CPU0 PTM	4KB	0xD46D_C000	0xD46D_CFFF
109		CPU1 PTM	4KB	0xD46D_D000	0xD46D_DFFF
110		RESERVED	8KB	0xD46D_E000	0xD46D_FFFF
111		RESERVED	16KB	0xD46E_0000	0xD46E_3FFF
112		RESERVED	16KB	0xD46E_4000	0xD46E_7FFF
113		RESERVED	4KB	0xD46E_8000	0xD46E_8FFF
114		RESERVED	4KB	0xD46E_9000	0xD46E_9FFF
115	RESERVED	RESERVED	32KB	0xD46E_A000	0xD46F_1FFF
116	Cortex A9 Dual core Processor	SCU	256B	0xD46F_2000	0xD46F_20FF
117		GIC-Interface	256B	0xD46F_2100	0xD46F_21FF
118		GT	256B	0xD46F_2200	0xD46F_22FF
119		RESERVED	768B	0xD46F_2300	0xD46F_25FF
120		PT and WDT	256B	0xD46F_2600	0xD46F_26FF
121		RESERVED	2304B	0xD46F_2700	0xD46F_2FFF
122		GIC-Distributor	4KB	0xD46F_3000	0xD46F_3FFF
123	L2CC	L2CC	4KB	0xD46F_4000	0xD46F_4FFF
124	RESERVED	RESERVED	44KB	0xD46F_5000	0xD46F_FFFF
125	ROM	ROM	64KB	0xD470_0000	0xD470_FFFF
126	RESERVED	RESERVED	~696MB	0xD471_0000	0xFFFF_FFFF

3. BUILDING U-BOOT FOR SHIKHARA

U-boot which is built by default will support following peripherals. DDR3 Controller, SMC, RTC, WDT, DT, SDMMC, UART (0-3), I2C (0-2), CAN, SPI (0-1), GPIO (0-3), I2S, KMI, USB3.0 Host, DRD Device, System Controller, GIC, DSI, HDMI and LCD.

The default Configuration Support:

1. Relocation off & Load Address 0x3D000000
2. GPU is disabled (Need to enable explicitly in include/configs/anusoc.h) when needed.
3. HDMI is configured for 640X480 Resolution to work with OV5640 Camera Preview (default option)
4. AV& GPU can work with HDMI 720X480 Resolution (Need to enable explicitly in the configuration when needed)

Following are setup and compilation steps to build the U-boot

1. Copy the ARM toolchain **gcc-arm-none-eabi-4_7-2012q4-20121208-linux.tar.bz2** to local folder on PC. Find the path for the tool-chain under the firmware release folder, the path is , -> **toolchain**
2. Extract the tool chain tar file.
3. Export the tool chain path by adding the export command in .bashrc file
-> **export PATH=\$PATH:/.../gcc-arm-none-eabi-4_7-2012q4/bin**
4. Use the following commands to compile U-boot source code.
-> **make ARCH=arm CROSS_COMPILE=arm-none-eabi- distclean**
-> **make ARCH=arm CROSS_COMPILE=arm-none-eabi- anusoc_config**
-> **make ARCH=arm CROSS_COMPILE=arm-none-eabi-**

This will generate the **uboot** elf file which is to be loaded using JTAG.

3.1 Building U-boot For MALI

To build u-boot for GPU need to enable the below flags in the **include/configs/anusoc.h** file. GPU validation has two test cases GPU and GPU Scene. These test cases by default are disabled in the u-boot. To enable need to enable flags in the path mentioned above. These test cases will not be validated at a time. When validating the test cases only related macros should be enabled.

When validating **GPU** enable the below macro only.

Define **CONFIG_SHIKHARA_GPU**

Enable the below macros for validating the **GPU scene** and when enabling this macros should disable the **CONFIG_SHIKHARA_GPU** macro.



```
Define      CONFIG_GPU_SCENE  and  
Define      HDMI_720_480
```

Use the following commands to compile U-boot source code.

```
->make ARCH=arm CROSS_COMPILE=arm-none-eabi- distclean  
->make ARCH=arm CROSS_COMPILE=arm-none-eabi- anusoc_config  
->make ARCH=arm CROSS_COMPILE=arm-none-eabi-
```

4. Shikhara SoC (ASIC) Boot modes and bootstraps:

4.1 Booting from JTAG

4.1.1 Connecting DSTREAM to Shikhara Target Board

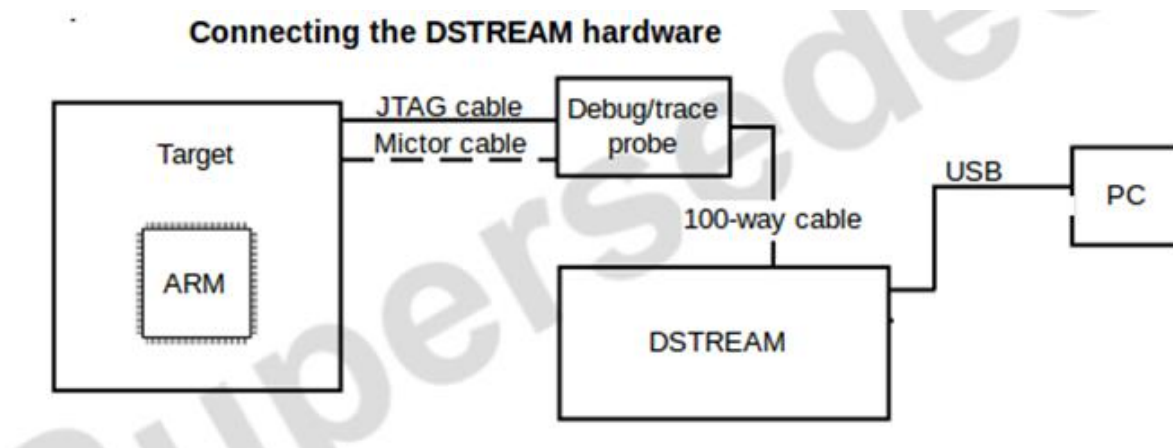
Following are the steps to be followed to connect the ARM DSTREAM unit to new Shikhara target:

To connect the DSTREAM unit to host computer and to the target hardware, carry out the following:

Have the Host PC running Ubuntu 14.04 LTS [64-bit], with DS-5 installed with required license. Connect the host computer to the DSTREAM unit as shown in the following figure using the USB port

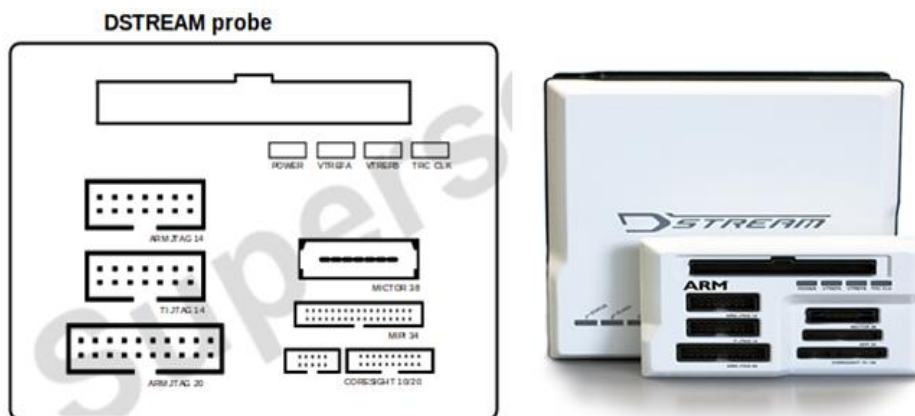
- If you are connecting using the USB port, connect one end of the supplied USB cable to a USB port on the host computer, and the other end of the cable to the USB port on the DSTREAM unit.

Note: The USB drivers are installed with the debug host software.



Connect the DSTREAM unit to the target hardware, using the appropriate debug or trace cables:

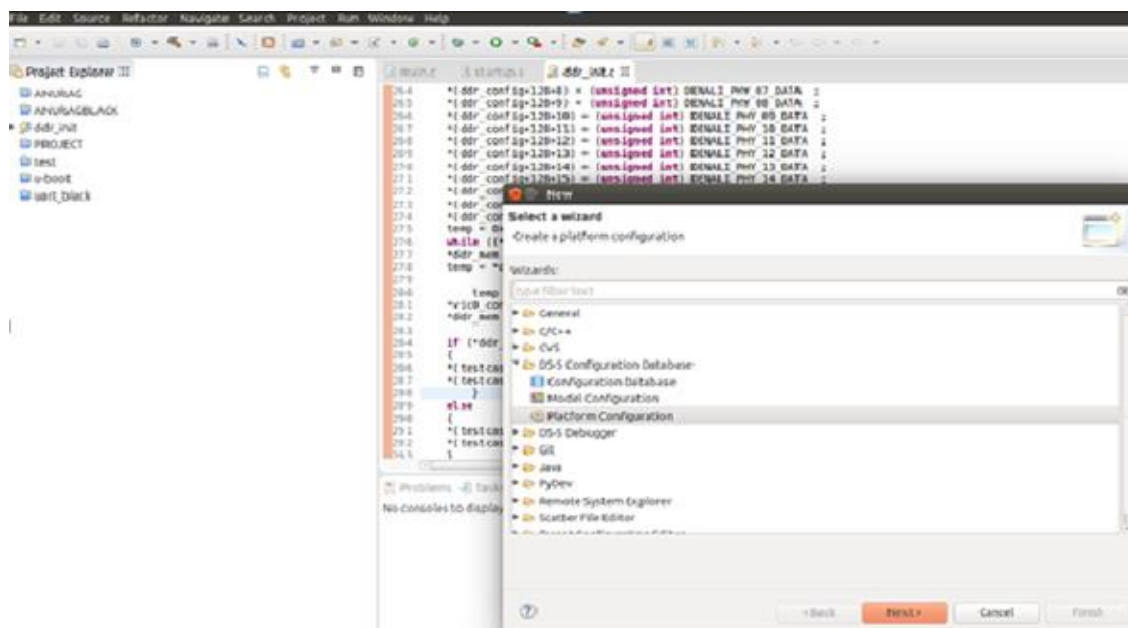
- Connect one end of the supplied 100-way cable to the DSTREAM unit, and connect the other end of the cable to the probe unit.
- Connect the target hardware to the probe using the appropriate cables and connector.
- ARM JTAG 20, This is the most commonly-used debug connector standard for ARM-based target boards



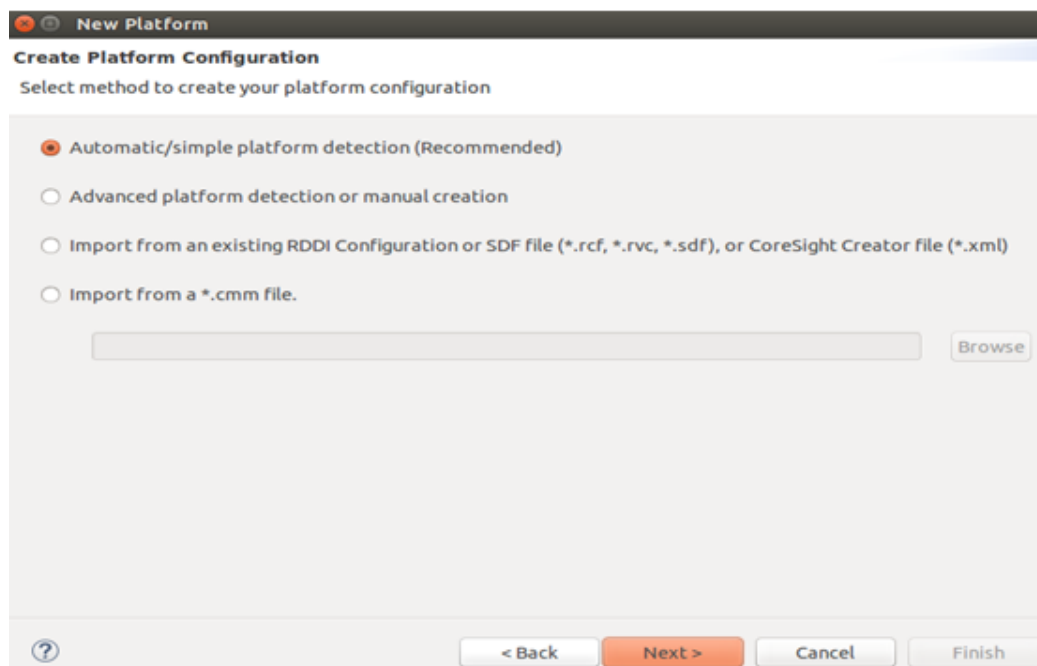
4.1.2 Creating a new platform configuration

Use the Platform Configuration Editor (PCE) within DS-5 to create debug configurations for new platforms. Creating a new platform configuration in DS-5 requires a new Platform Configuration project in Eclipse

- From the main menu in DS-5, select **File > New > Other** to open the new project dialog.
- Select **DS-5 Configuration Database > Platform Configuration** and then click **Next**.

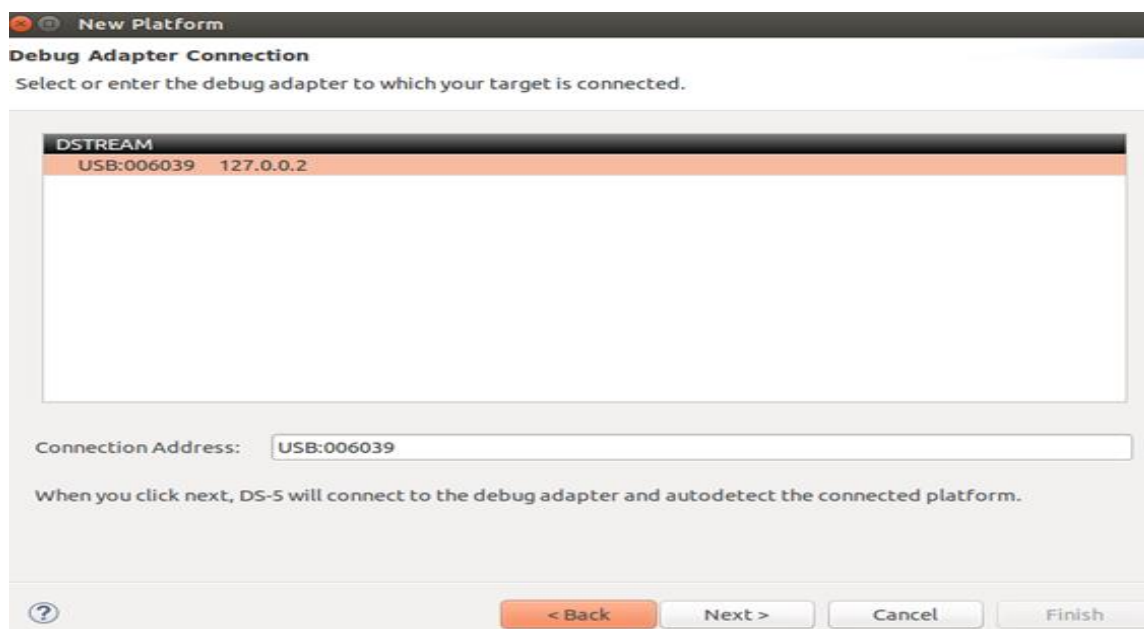


- This shows the **Create Platform Configuration dialog box**. Select the required method to create the configuration for your platform and click **Next**.
- Select **DS-5 Configuration Database > Platform Configuration** and then click **Next**

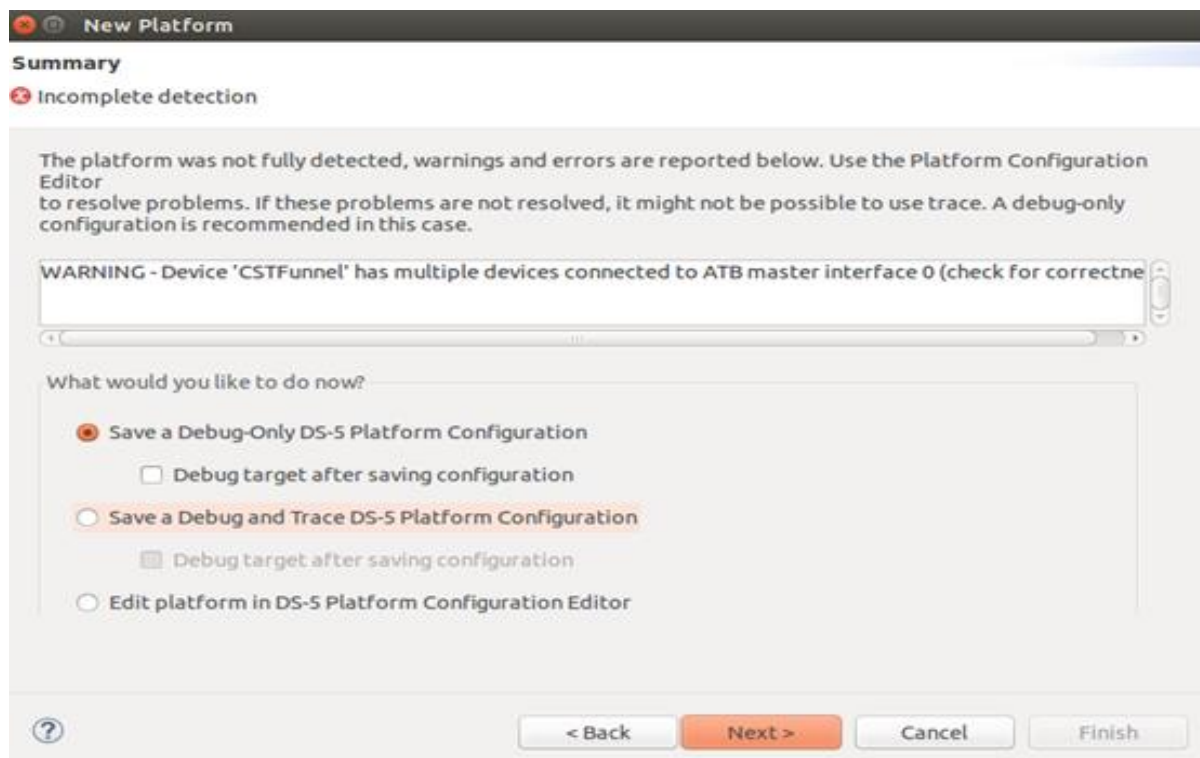


➤ Automatic/simple platform detection

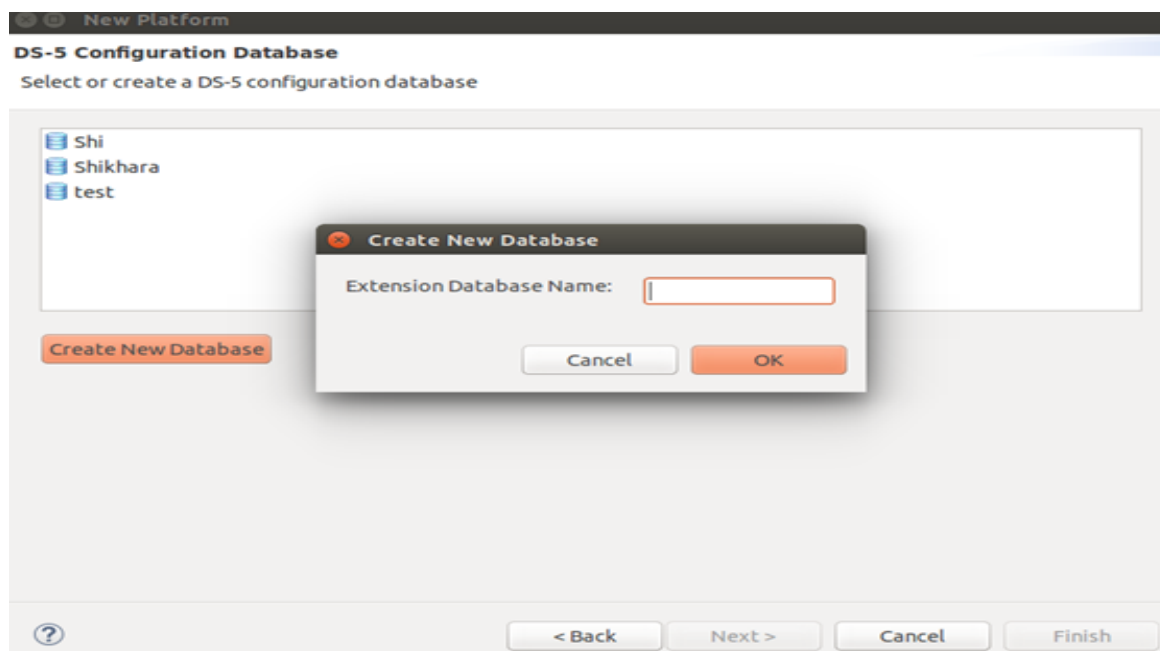
This is the recommended option. DS-5 automatically detects the devices that are present on your platform. It then provides you the opportunity to add more devices if needed and to specify how the devices are interconnected.



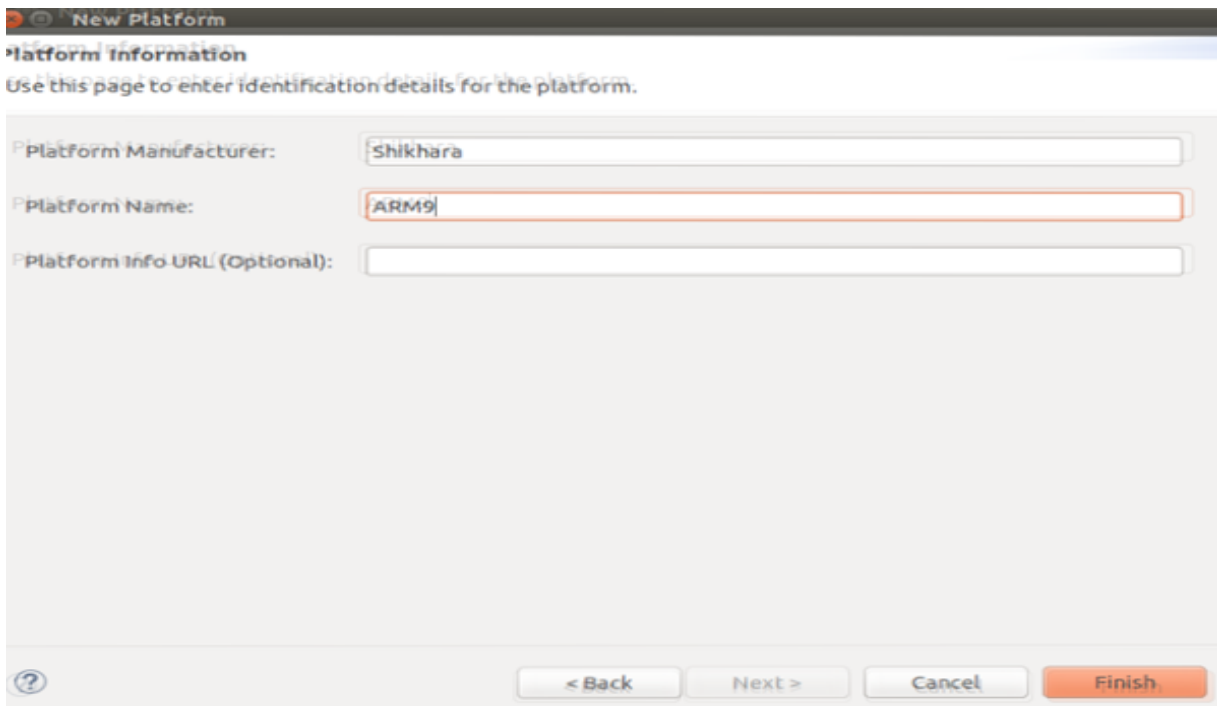
➤ Once DSTREAM unit has been selected, DS-5 will connect to the SoC and try to read all the information it needs for debug and trace , on Successful



- Click on **Create New Database**; enter the name for Extension Database Name



- Enter Platform Manufacture and Platform Name



New Platform

Platform Information

Use this page to enter identification details for the platform.

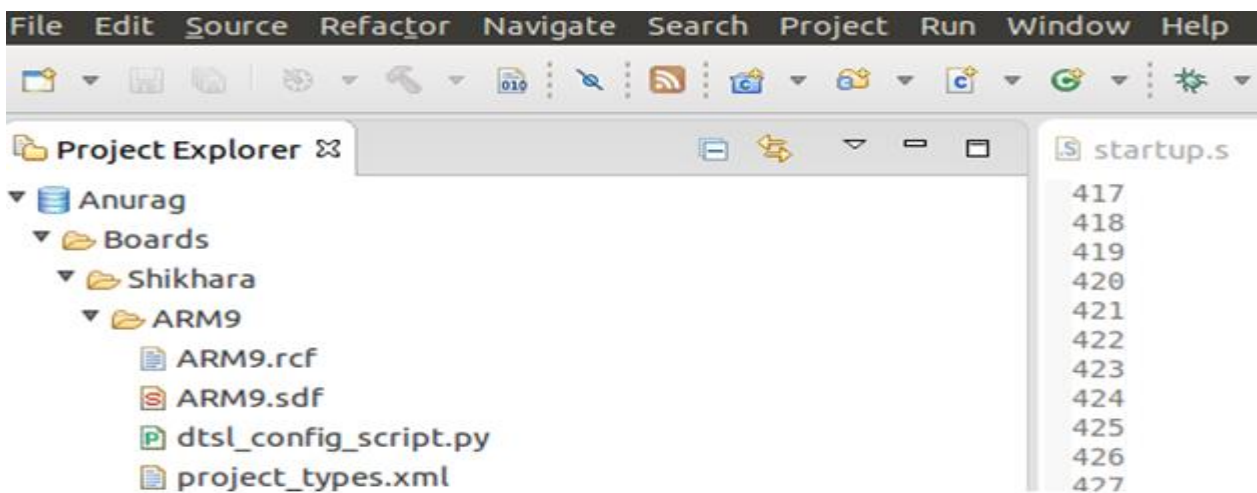
Platform Manufacturer:

Platform Name:

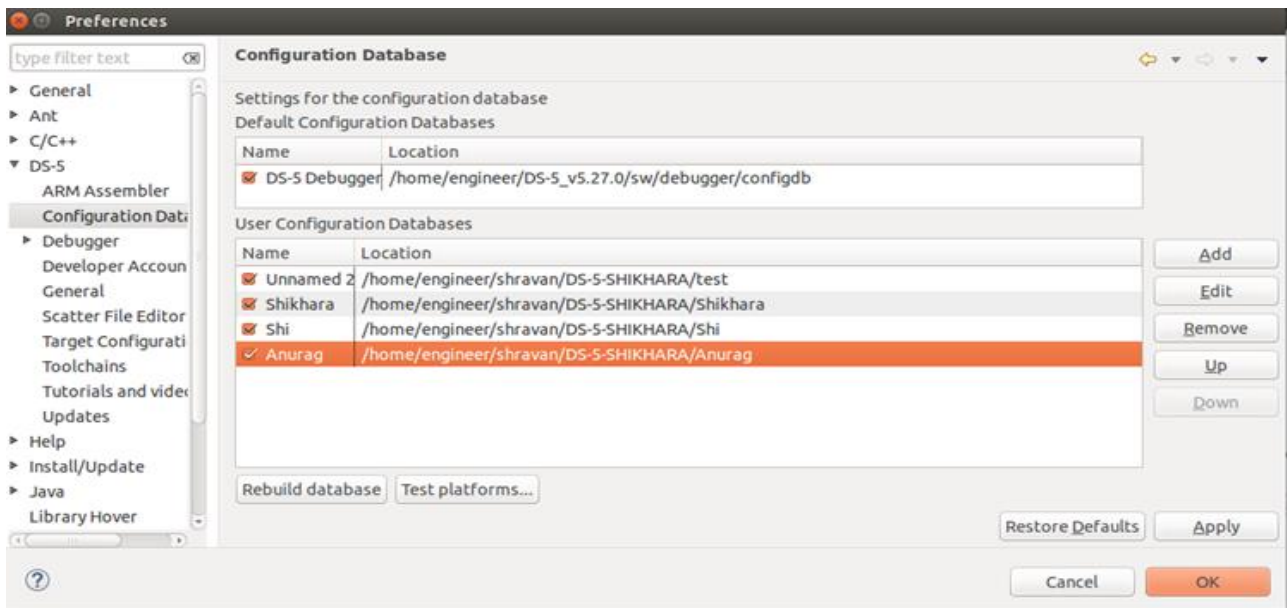
Platform Info URL (Optional):

< Back Next > Cancel Finish

- On successful, it will show as below



- Window -> Preferences -> DS-5 -> Configuration -> Add the Configuration Path , then do the Rebuild the database

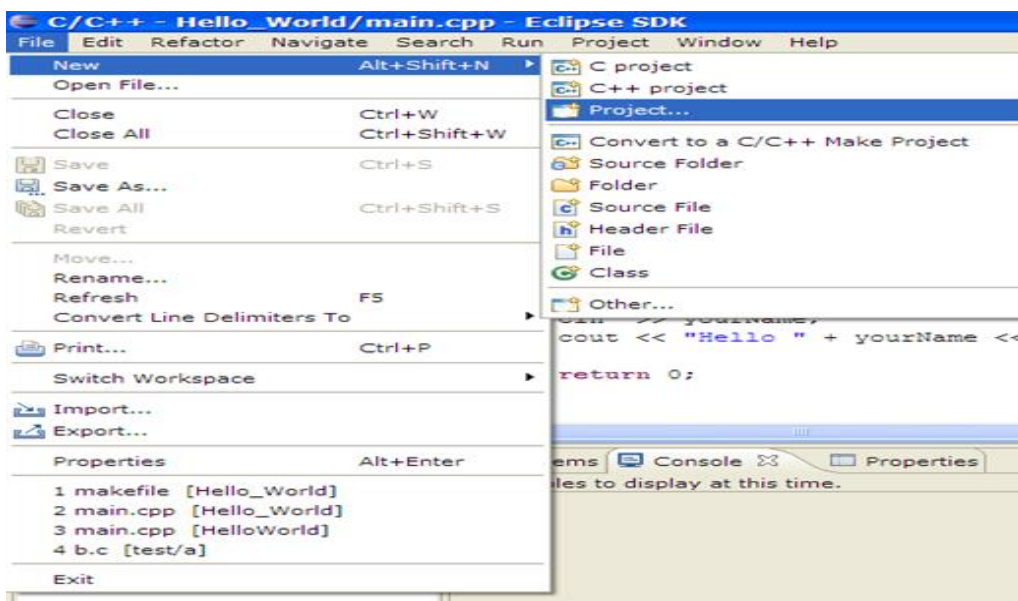


4.1.3 Creating Project in Eclipse

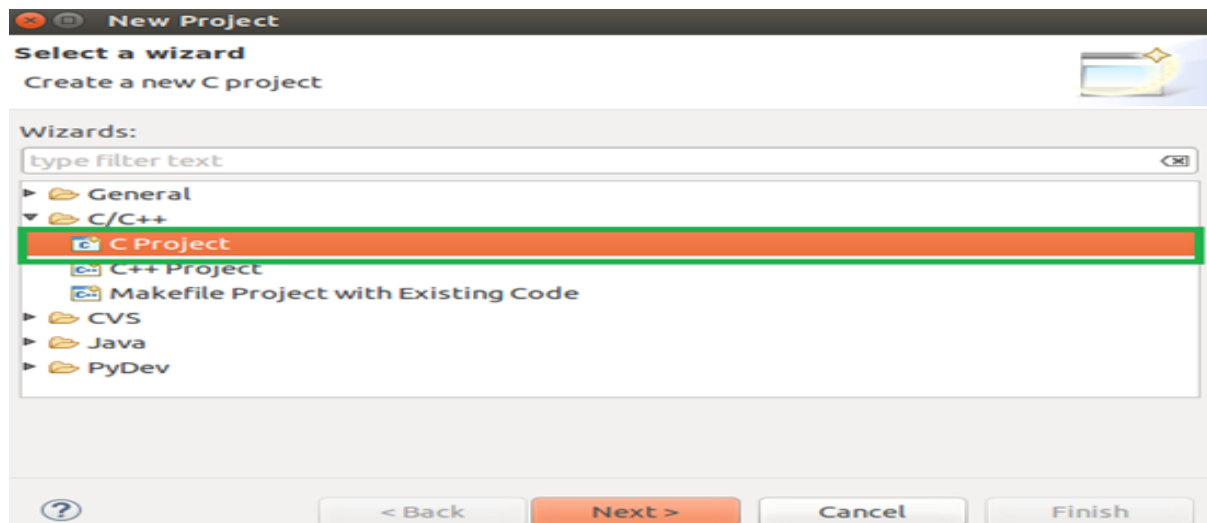
You can create a standard make or managed make C or C++ project.

To create a project:

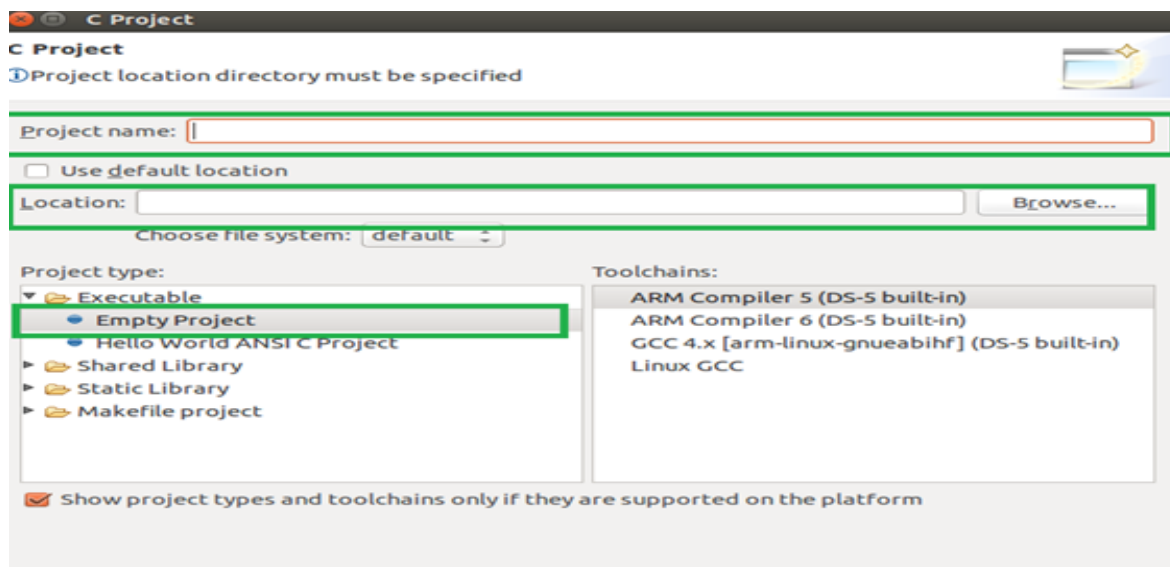
- **File > New > Project.**



- Select **C Project** [Select the type of project to create. Expand the **C/C++** folder and select **C Project**



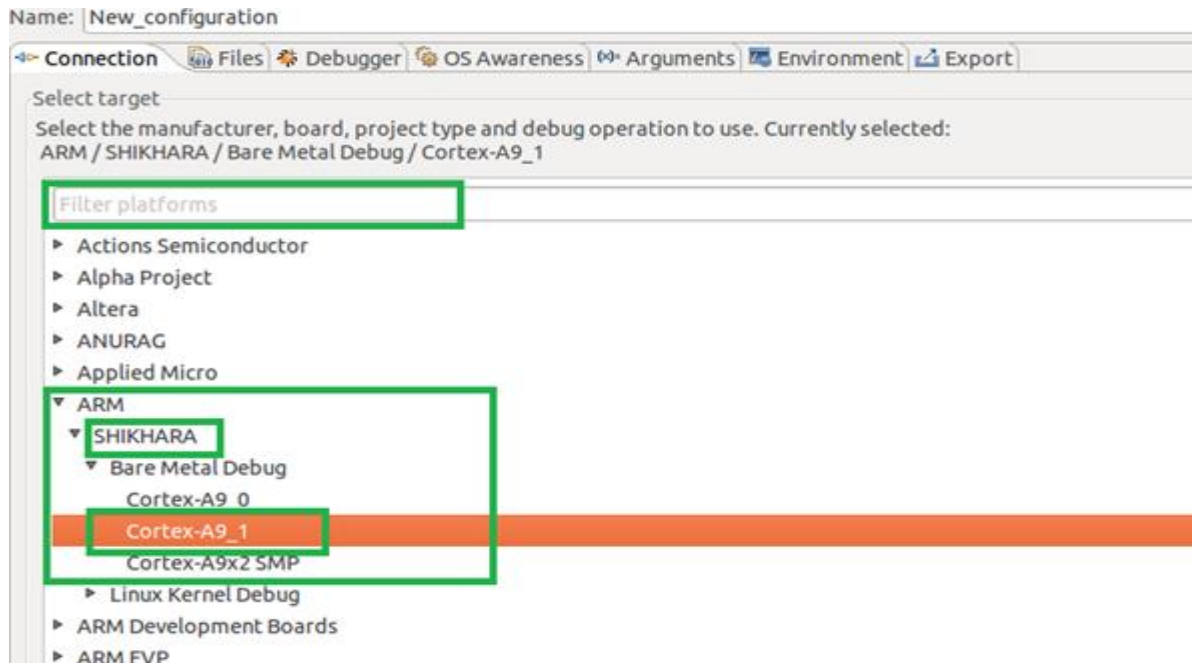
- In the **Project name** field, type `test_led`. Leave the **Use Default Location** option selected. **Empty Project** provides a single source project folder that contains files



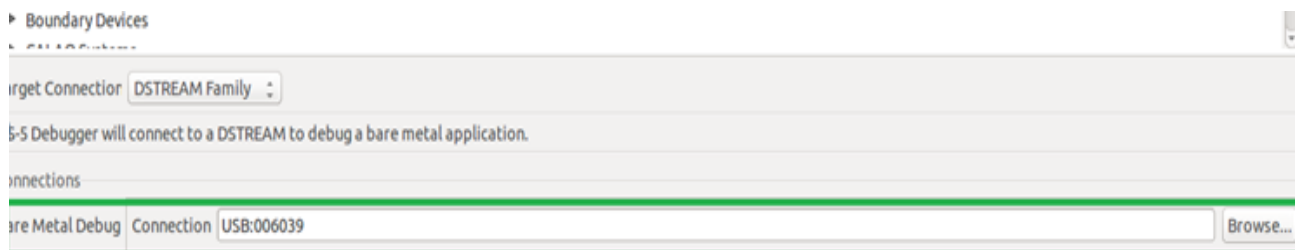
- Project Explorer, will show the **project name, with source and executable [elf] file**
- To debug a project:
 - **Run > Debug Configurations...** menu option. The **Debug Configurations** dialog opens.
 - From the new configuration , workspace , select the elf file location

- From the new configuration ,Debugger, select entry point

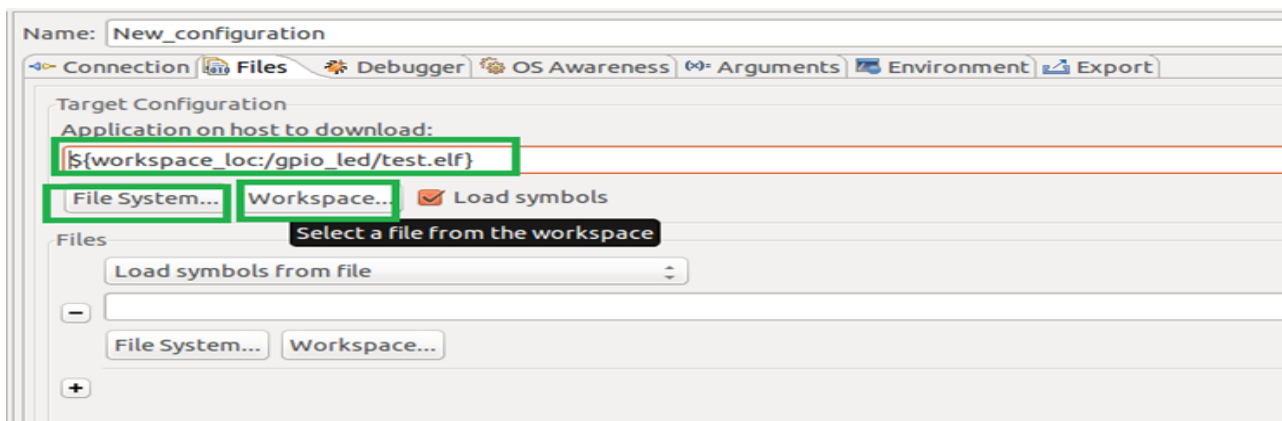
Search for “Platform Manufacture Name” in Filter Platforms, after that it will show, select Bare Metal Debug, then Cortex_A9_1



- Browse and detect the USB Device from “Bare Metal Debug”



- Select the generated elf from File System or workspace



- From the new configuration, Debugger tab, select entry point as “Debug from the entry point”



- Then click on apply icon at bottom, by this connecting D-stream to shikhara board completed will get the u-boot prompt on the serial console after succesfull completion of these procedure.

NOTE:

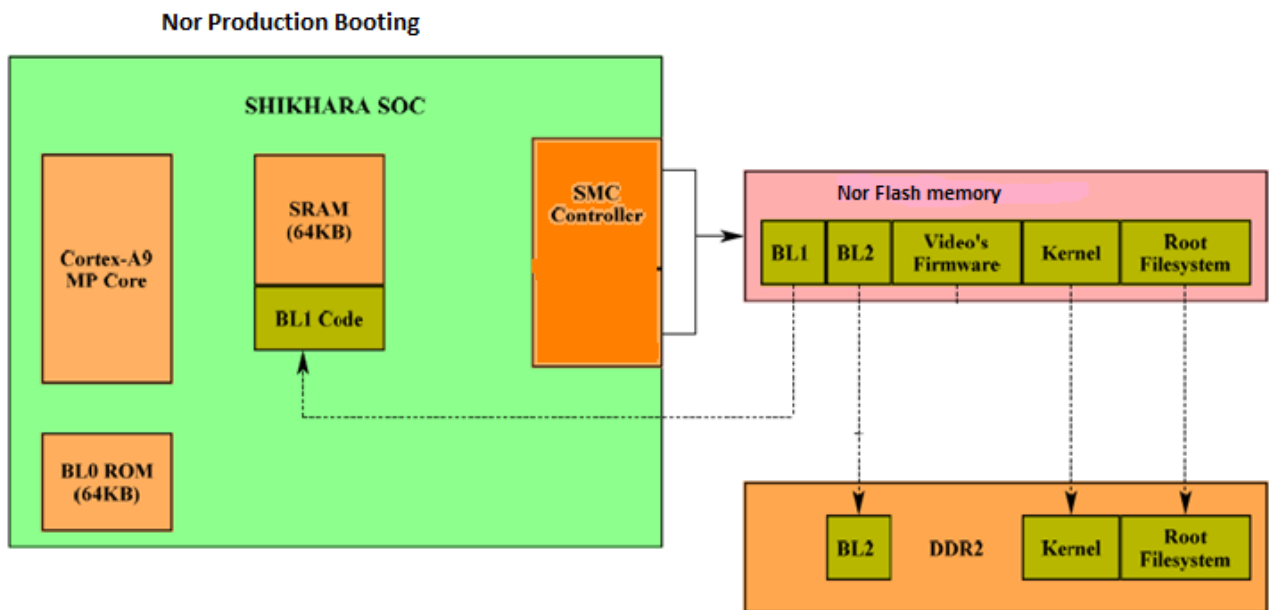
1. When selecting the files from the work place, firstly, need to select the bare metal code of DDR memory initialization which will initialize ddr and then we can copy u-boot into the ddr memory.
2. Then after ddr initialization, select the u-boot elf file from the workplace and run which will give the u-boot prompt on the serial console to enter the test commands.

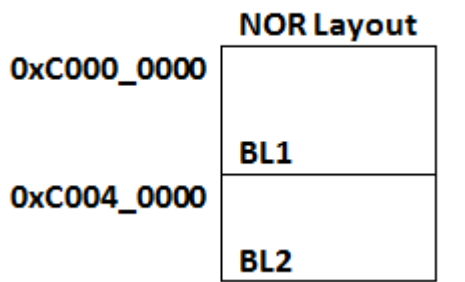
4.2 Booting from NOR

4.2.1 Booting from NOR Production mode:

- On power on Reset (PoR), PC (Program Counter register of ARM) value (0x00000000) points to NOR flash.
- NOR flash contains of BL1 and BL2.
- BL1 code gets executed from the NOR flash with PC value of NOR_BASE address. This BL1 firmware involves in initializing the DDR memory.
- BL1 firmware copies BL2 firmware (U-BOOT) to DDR memory and run the BL2 code (U-BOOT) from DDR.
- Responsibility of BL2 code is to initialize the other peripherals (E.g.: LCD, GPIO, USB) which are not initialized by the BL1.

Can observe the images below for details. And for the address mapping to where the BL1 and BL2 bin are to be stored and followed by the procedure to be followed for the NOR production booting.





	offset	length
BL1	0xC0000000	DDR_bare_code_size
BL2(u-boot.bin)	0xC0040000	uboot_bin_size

Procedure to be followed:

Firstly, be ready with the u_boot prompt by following the above procedure (**Booting from JTAG section**). After getting u_boot prompt follow below procedure to load BL1 (DDR initialization bare code) and BL2 (U-boot binary) which are provided in the firmware release files.

The path for the BL1 (DDR initialization bare metal code)

->firmware\bare-metal-code-shikhara\Shikhara_Boot_failure_case_NOR_Production_Mode\sample_ddr_load_u_boot

The path for BL2(u-boot bin file)

->firmware\uboot_single_binary

Enter below commands in the u-boot prompt.

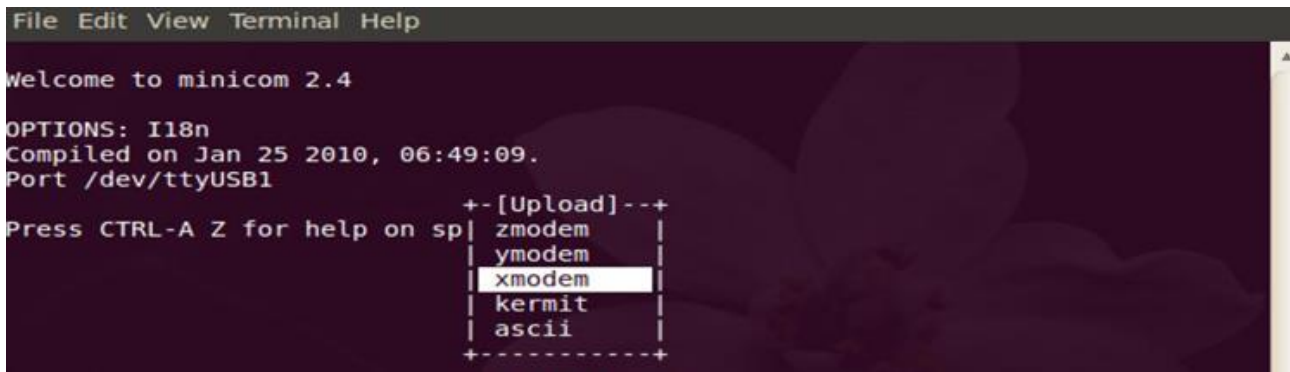
- 1. nor_init**
- 2. flash_start**
- 3. protect off all**
- 4. erase 0xc0000000 0xc00ffff**
- 5. loadx 0x100 (BL1 bin file)**

Follow the below procedure for transferring file from host pc using minicom xmodem.

File Transfer from Host PC using minicom xmodem

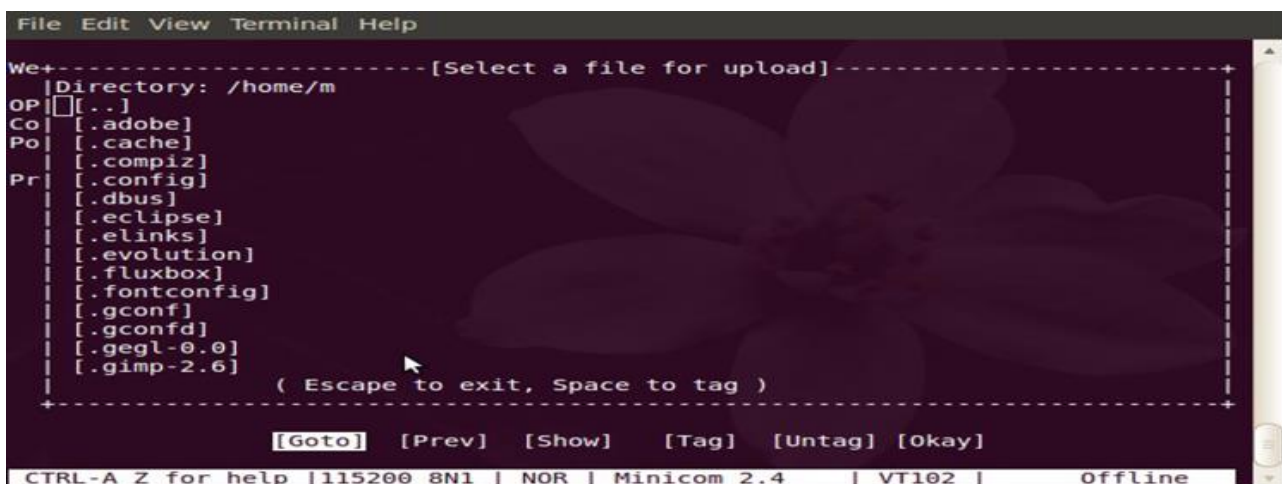
Send a file through xmodem protocol using minicom. For this it is assumed the target device connected at the other end of the port uses the same protocol and configured with same settings like Bps/Par/bits.

- To send a file press CTRL – A S , this selects the option Send file , Highlight xmodem protocol and press enter



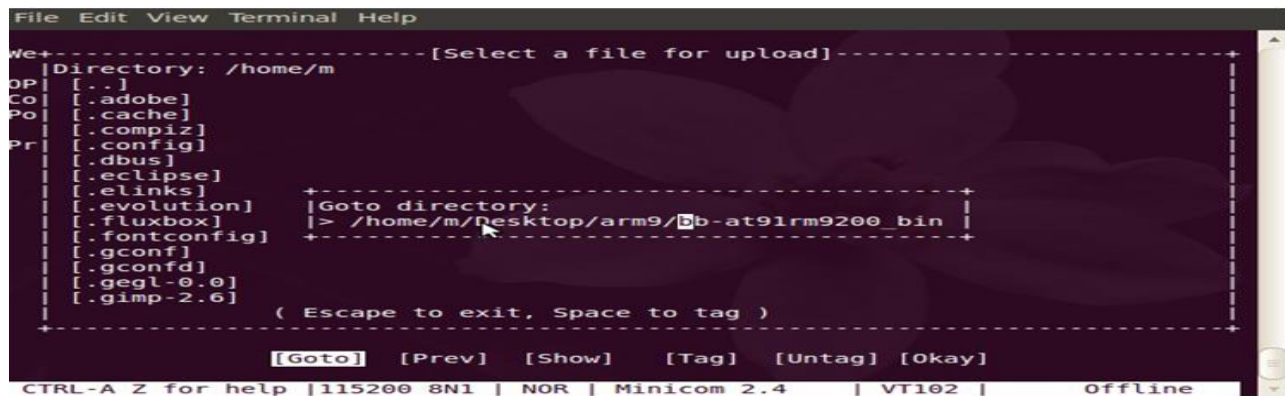
```
File Edit View Terminal Help
Welcome to minicom 2.4
OPTIONS: I18n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyUSB1
Press CTRL-A Z for help on sp
+-[Upload]--+
| zmodem
| ymodem
| xmodem
| kermi
| ascii
+-----+
```

- Use arrow keys to highlight [Goto] and press enter

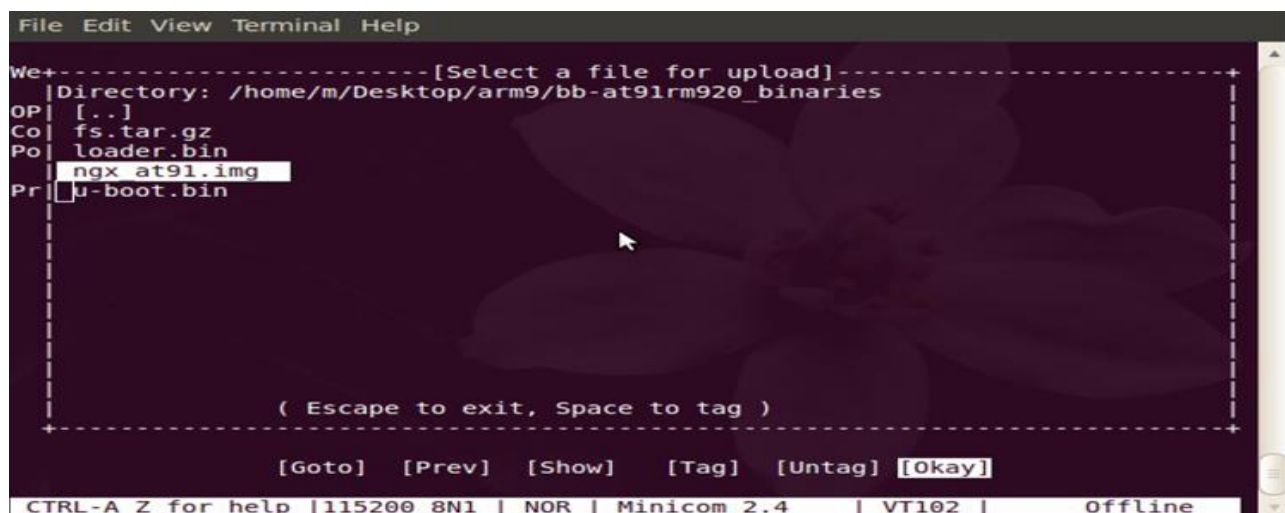


```
File Edit View Terminal Help
We+-----[Select a file for upload]-----+
|Directory: /home/m
OP|[[...]]
Co|[[.adobe]]
Po|[[.cache]]
Pr|[[.compiz]]
|[[.config]]
|[[.dbus]]
|[[.eclipse]]
|[[.elinks]]
|[[.evolution]]
|[[.fluxbox]]
|[[.fontconfig]]
|[[.gconf]]
|[[.gconfd]]
|[[.gegl-0.0]]
|[[.gimp-2.6]]
|
| ( Escape to exit, Space to tag )
+-----+
|[[Goto]]| [[Prev]] | [[Show]] | [[Tag]] | [[Untag]] | [[Okay]]
+-----+
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4 | VT102 | Offline
```

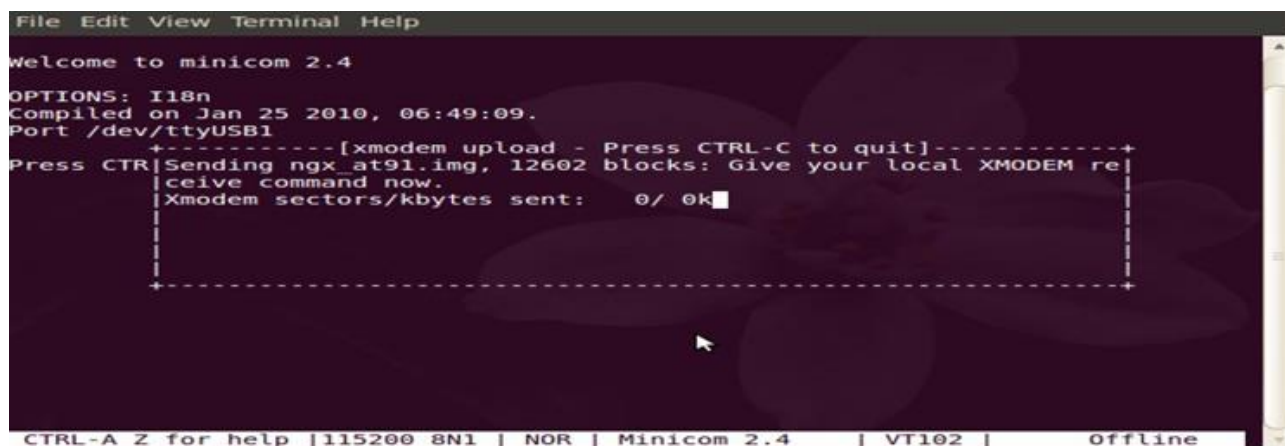
- Enter the path of the directory to send the file. To get the path of the files you you can do as follows...



- Highlight the file to send using arrow keys and press spacebar to select the file. Press enter, **select BL1.bin**



- File is sent and the progress is shown in xmodem sectors and Kbytes sent

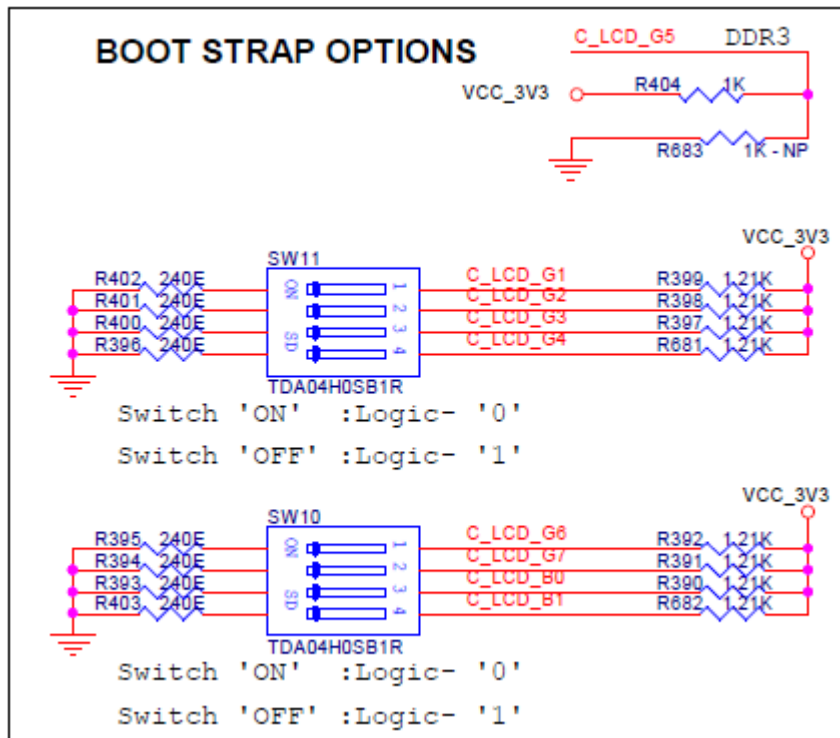


6. cp.b 0x100 0xc0000000 0x1dc0 (length in hexadecimal depends on the file transferred)

7.loadx 0x100 (u-boot,BL2 bin file)(Procedure is same as above for loadx file transfer)

8.cp.b 0x100 0xc0040000 0x83478 (length in hexadecimal)

Refer the following image for NOR production booting for changing the bootstrap pins/switches.



Select the switches from SW10

C_LCD_G6	C_LCD_G7	C_LCD_B0	C_LCD_B1	Boot Mode
0	0	0	0	Production Level Booting(From NOR)

After completion of the above procedure, power off the board and follow the below steps,

1. Change the boot strap mode to NOR production boot (0000) as shown in the above images,
2. Disconnect the D-stream and Connect serial terminal to UART-0,
3. power on

Then you will be able to see the u-boot prompt upon power on the board.

4.2.2 Booting from NOR BootRom Mode

Shikhara SoC has got inbuilt Boot Rom Code which gets executed on power on if the Boot Strap Pins are set as below.



Firstly, be ready with the u_boot prompt by following the above procedure (**Booting from JTAG section**). After getting u_boot prompt follow below procedure to load BL1 (DDR initialization bare code) and BL2 (U-boot binary) which are provided in the firmware release files.

To use this booting mechanism, Please follow below steps to flash BL1 and BL2 images on to NOR flash.

The path for the BL1 (DDR initialization bare code) is under the firmware release folder mentioned below,

- **firmware\bare-metal-code-shikhara\ROM_CODE_BL1\NOR\BL1_NOR_DDR_Init_load_uboot**

The path for the BL2 (U-boot bin file) is ,

- **firmware\uboot_single_binary**

Enter below commands in the u-boot prompt.

1. **nor_init**
2. **flash_start**
3. **protect off all**
4. **erase 0xc0000000 0xc00fffff**
5. **loadx 0x100 (BL1 bin file)**

File Transfer from Host PC using minicom xmodem

Send a file through xmodem protocol using minicom. For this it is assumed the target device connected at the other end of the port uses the same protocol and configured with same settings like Bps/Par/bits.

- To send a file press CTRL – A S , this selects the option Send file , Highlight xmodem protocol and press enter

```
File Edit View Terminal Help
Welcome to minicom 2.4
OPTIONS: I18n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyUSB1
Press CTRL-A Z for help on sp|
+-[Upload]--+
| zmodem    |
| ymodem    |
| xmodem    |
| kermi     |
| ascii     |
+-----+
```

- Use arrow keys to highlight [Goto] and press enter


```
File Edit View Terminal Help
Welcome to minicom 2.4
-----[Select a file for upload]-----
Directory: /home/m/Desktop/arm9/bb-at91rm920_binaries
OP| [...]
Co| fs.tar.gz
Po| loader.bin
Pr| ngx_at91.img
  | u-boot.bin

( Escape to exit, Space to tag )

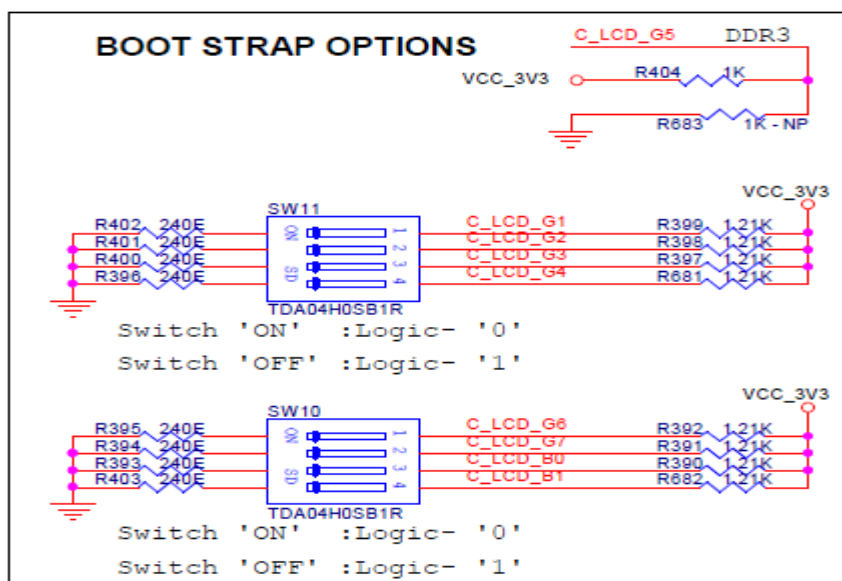
[Goto] [Prev] [Show] [Tag] [Untag] [Okay]
CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4 | VT102 | Offline
```

- File is sent and the progress is shown in xmodem sectors and Kbytes sent

```
File Edit View Terminal Help
Welcome to minicom 2.4
OPTIONS: I18n
Compiled on Jan 25 2010, 06:49:09.
Port /dev/ttyUSB1
-----[xmodem upload - Press CTRL-C to quit]-----
Press CTRL-C to abort.
Sending ngx_at91.img, 12602 blocks: Give your local XMODEM receive command now.
Xmodem sectors/kbytes sent:  0/ 0k
-----

CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.4 | VT102 | Offline
```

6. cp.b 0x100 0xc0000000 0x1dc0 (length in hexadecimal depends on the file transferred)
7. loadx 0x100 (u-boot,BL2 bin file)(Procedure is same as above for loadx file transfer)
8. cp.b 0x100 0xc0040000 0x83478 (length in hexadecimal)



Select the switches of SW10

C_LCD_G6	C_LCD_G7	C_LCD_B0	C_LCD_B1	Description
1	0	0	1	NOR booting from ROM

After completion of the above procedure, power off the board and follow the below steps,

1. Change the boot strap mode to NOR production boot (1001) as shown in the above images,
2. Disconnect the D-stream and Connect serial terminal to UART-0,
3. power on

Then you will be able to see the u-boot prompt upon power on the board.

4.3 Memory Arrangement/Allotment for the IP's

Memory Arrangement/Allotment for the IP's, to remove the conflict during validation.

NAME	START ADDRESS	END ADDRESS	SIZE
U-BOOT(Code/Text)	0X3D00_0000	+50MB(Size)	50MB Allocated
HDMI - FB	0X0120_0000	+ 0x600000	6.2 MB
MIPI-DSI-FB	0X0180_0000	+ 0x600000	6.2 MB
CLCD-FB	0X01E0_0000	+ 0x600000	6.2 MB
HDMI - FB_SLIDESHOW_1	0X0640_0000	+ 0x600000	6.2 MB

HDMI - FB_SLIDESHOW_2	0x06A0_0000	+ 0x600000	6.2 MB
HDMI - FB_SLIDESHOW_3	0x0700_0000	+ 0x600000	6.2 MB
HDMI - FB_SLIDESHOW_4	0x0760_0000	+ 0x600000	6.2 MB
HDMI - FB_SLIDESHOW_5	0x07C0_0000	+ 0x600000	6.2 MB
HDMI- VIDEO PLAY	0X0640_0000	+ 0x600000	6.2 MB
HDMI- DDR_AUDIO_RAW	0x0640_0000		
HDMI - BMP_SLIDESHOW_1	0X08C0_0000	+ 0x600000	6.2 MB
HDMI - BMP_SLIDESHOW_2	0X0920_0000	+ 0x600000	6.2 MB
HDMI - BMP_SLIDESHOW_3	0X0980_0000	+ 0x600000	6.2 MB
HDMI - BMP_SLIDESHOW_4	0X09E0_0000	+ 0x600000	6.2 MB
HDMI - BMP_SLIDESHOW_5	0X0A40_0000	+ 0x600000	6.2 MB
SHIKHARA_USB_DE VICE_STORAGE_ST ART_OFFSE T	0x0AA0_0000	+ 100MB	100MB
CSI- Camera	0X1100_0000	+200_0000	+32MB
DDR-TEST	0X1300_0000	+10000	+64KB
AV417	0x0150_0000 0x015F_F200 0x01D0_0000 0x01B0_0000		

5. U-BOOT BASED VALIDATION TEST REPORT

5.1 TEST CASES FOR MEMORY VALIDATION

5.1.1 Ex-SRAM Test: Write & Read Operations On External SRAM

To validate the memory, a command was ported into u-boot. Memory locations were tested by writing the known patterns and by reading back the same location, and compared read values with the written values and prints the test result on the console.

Execution of test case using Commands:

- **sram_test**

Upon executing the above test command it will perform write and read operations to SRAM and displays the test result on the console.

Console Output:

```
shikhara-uboot>sram_test  
SRAM data is same and SMC Controller is working
```

5.1.2 NOR Test: Write & Read Operations On NOR

The NOR flash validated by writing some known patterns on to the NOR memory locations. And then the same memory locations were read back and compared with known written pattern. To test the NOR Memory functionality, a command was ported into u-boot to perform pattern tests.

Execution of test case using Commands:

- **nor_init**
- **nortest <Addr> <Len>**

This is the test command used to test the NOR flash, For this ported “**nortest**” command in the u-boot. First argument is the address of the NOR flash and second argument is the length that how much size you want to test.

Console Output:

```
shikhara-uboot>nortest 0xc2000000 0x40000  
Erasing 0xc2000000 - 0xc203ffff  
.. done  
Erased 2 sectors  
Copying to flash...  
2....1....*****Copying from flash...  
Comparing Data ...  
Test passed... Total no of bytes: 262144
```


5.1.3 NAND Validation

This NAND test case is validated by U-boot command “**nandtest**” which will write some random data to nand flash and read back from same address. Compares written values with read values and print the result and prints the test result on the console.

Execution of test case using Commands:

- **nand_start**
- **nand info**
- **nandtest**

Check for list of available NAND devices on board using command **nand info**. In U-boot commands, run “**nandtest**” command which will start writing and reading to the NAND flash and prints the result of the test case.

Console Output:

```
shikhara-uboot> nand_start
NAND: ONFI flash detected
ONFI flash detected
2048 MiB
shikhara-uboot> nandtest
Bad NAND Block Count 0
Running Erase Test...Please wait...Start Block : 3, End Block : 10
PASSED
Running Write test...Please wait...Start Block : 3, End Block : 10
PASSED
Running Read test...Please wait...Start Block : 3, End Block : 10
PASSED
Erase nand after completing the Nand write read test...
Running Erase Test...Please wait...Start Block : 3, End Block : 10
Nand Test Completed... PASS
```

5.1.4 DDR Test: Write & Read Operations On DDR

The DDR memory was validated by writing known patterns. Firstly by using the command will write a fixed pattern on to the specified locations. After writing to the locations, the same locations were read back and validated with the fixed pattern. And prints the test result on the console.

Execution of test case using Commands:

- **mw <Addr> <Data> <Len>**
- **md <Addr> <Len>**

Upon executing the above u-boot commands first will write data in the address specified the size of length specified using “**mw**” command. Then by using the “**md**” memory display command will



display the content on the address which we have written.

Console Output:

```
shikhara-uboot>mw 0x0 0x1234 0x100
shikhara-uboot>md 0x0 0x100
00000000: 00001234 00001234 00001234 00001234  4...4...4...4...
00000010: 00001234 00001234 00001234 00001234  4...4...4...4...
00000020: 00001234 00001234 00001234 00001234  4...4...4...4...
00000030: 00001234 00001234 00001234 00001234  4...4...4...4...
00000040: 00001234 00001234 00001234 00001234  4...4...4...4...
00000050: 00001234 00001234 00001234 00001234  4...4...4...4...
```

5.2 TEST CASES FOR HIGH SPEED PERIPHERALS

5.2.1 SD / MMC Validation

5.2.1.1 SD / MMC Test (MSHC Host Controller 0) – File transfer test

This test case is validated by transferring a file from ram memory to the SD/MMC card. And then copying the file from **sd/mmc** to the other ram memory location and then displaying the two locations through “md” command and verifying the same data that should be present on both locations.

Peripheral Instance	Command Used	Test Status	Remarks
SD / MMC	mmc_start , mmc_start, fatls mmc, fatwrite mmc fatload mmc	Passed	Tested using U-Boot

Execution of test case using Commands:

- **mmc_start**
- **mmc dev 0**
- **fatls mmc 0**
- **fatwrite mmc 0 <Addr> <file_name> <Len>**
- **fatload mmc 0 <Addr> <file_name>**

These are the test commands used to validate sd/mmc. **mmc_start** is the command used to initialize the MMC card, **fatwrite** is the command used to write data to a file in file system on SD card using fatwrite. Read the dumped file to memory location on RAM using **fatload** command.

Console Output:

```
shikhara-uboot>mmc_start
MMC initialization:
SHIKHARA DWMMC: 0, SHIKHARA DWMMC: 1, SHIKHARA DWMMC: 2
Dwmc_i_init Host base address = 0xd4555000
MMC Device 0: found
Dwmc_i_init Host base address = 0xd4555000
MMC Device 1: found
Dwmc_i_init Host base address = 0xd4556000
MMC Device 2: found
shikhara-uboot>mmc dev 0
Dwmc_i_init Host base address = 0xd4555000
mmc0 is current device
```

```
shikhara-uboot>fatls mmc 0
Dwmci_init Host base address = 0xd4555000
1036854 1.bmp
3668 app_alphablending.bin
shikahra/
1036854 2.bmp
1036854 3.bmp
3452 app_ipc.bin
.....
.....
17000532 6.bmp
34080436 sia_common_common.wav
5760138 image1.bmp

31 file(s), 3 dir(s)

shikhara-uboot>fatload mmc 0 0x10 tqwe.txt
Dwmci_init Host base address = 0xd4555000
reading tqwe.txt
...20 bytes read
```

5.2.1.2 SDIO Test (MSHC Host Controller 1) – File transfer test

This test case is validated by transferring a file from ram memory to the SDIO. And then copying the file from sd/mmc to the other ram memory location and then displaying the two locations through “md” command and verifying the same data should be present on both locations.

Peripheral Instance	Command Used	Test Status	Remarks
SDIO	mmc_start , mmc_start, fatls mmc, fatwrite mmc fatload mmc	Passed	Tested using U-Boot

Execution of test case using Commands:

- **mmc_start**
- **mmc dev 1**
- **fatls mmc 1**



- **fatwrite mmc 1 <Addr> <file_name> <Len>**
- **fatload mmc 1 <Addr> <file_name>**

These are the test commands used to validate sd/mmc. **mmc_start** is the command used to initialize the MMC card, **fatwrite** is the command used to write data to a file in file system on SD card using fatwrite. Read the dumped file to memory location on RAM using **fatload** command.

Console Output:

```
shikhara-uboot>mmc_start
MMC initialization:
MMC Device 2: found
shikhara-uboot>mmc dev 1
Dwmci_init Host base address = 0xd4555000
mmc1 is current device
shikhara-uboot>env export 0x200 bootdelay
shikhara-uboot>md 0x200 0x10
00000200: 746f6662 616c6564 0a333d79 0a000000  bootdelay=3.....
00000210: 03083004 3300052d 0936060b 05093605  .0..-..3..6..6..
00000220: 35070b37 05310509 02073401 3002052f  7..5..1..4../..0
00000230: 08300308 00052d03 31030830 0a320409  ..0..-..0..1..2.
shikhara-uboot>fatwrite mmc 1 0x200 env_export.txt 0x50
writing env_export.txt
80 bytes written
shikhara-uboot>fatls mmc 1
 921738  1.bmp
 921738  2.bmp
      shikahra/
.....
      convert_bmp_files_to_yuv444/
      80  env_export.txt
6 file(s), 3 dir(s)
shikhara-uboot>fatload mmc 1 0x150 env_export.txt
reading env_export.txt
80 bytes read
shikhara-uboot>md 0x150 0x50
00000150: 746f6662 616c6564 0a333d79 0a000000  bootdelay=3.....
00000160: 03083004 3300052d 0936060b 05093605  .0..-..3..6..6..
```

5.2.2 USB Host Validation

- USB Host Test_1: Start/Stop USB Controller
- USB Host Test_2: USB Device Enumeration test.
- USB Host Test_3: BULK-OUT Transfer
- USB Host Test_4: BULK-IN Transfer

5.2.2.1 USB Host Test_1: Start/Stop USB Controller

Start/Stop USB Controller using built in 'usb' commands in the u-boot.

Peripheral Instance	Command Used	Test Status	Remarks
USB Host	usb start	Passed	Tested using U-Boot

Execution of test case using Commands:

- **usb start**

This is the test command to be executed to start or stop the usb controller, which will display the information on the console regarding the usb.

Console Output:

```
shikhara-u-boot>usb start
(Re)start USB...
USB0: Starting the controller
USB XHCI 1.00
scanning bus 0 for devices... 2 USB Device(s) found
    scanning usb for storage devices... 1 Storage Device(s) found
```

5.2.2.2 USB Host Test_2: USB Device Enumeration test

This test case is used to display the debugging information regarding the usb ,which will show whether the device connected to board are found or not.

Peripheral Instance	Command Used	Test Status	Remarks
---------------------	--------------	-------------	---------

USB Host	usb info	Passed	Tested using U-Boot
----------	----------	--------	---------------------

Execution of test case using Commands:

- *usb info*

This is the test command which is a built in u-boot used to display the debugging information of the usb at enumeration time.

Console Output:

```
shikhara-uboot>usb info
```

```
1: Hub, USB Revision 3.0
```

- u-boot XHCI Host Controller
- Class: Hub
- PacketSize: 9 Configurations: 1
- Vendor: 0x0000 Product 0x0000 Version 1.0
- Configuration: 1
 - Interfaces: 1 Self Powered 0mA
 - Interface: 0
 - Alternate Setting 0, Endpoints: 1
 - Class Hub
 - Endpoint 1 In Interrupt MaxPacket 8 Interval 255ms

```
2: Mass Storage, USB Revision 2.0
```

- hp v236w E6007420F77E9643
- Class: (from Interface) Mass Storage
- PacketSize: 64 Configurations: 1
- Vendor: 0x03f0 Product 0x7d40 Version 1.0
- Configuration: 1
 - Interfaces: 1 Bus Powered 200mA
 - Interface: 0
 - Alternate Setting 0, Endpoints: 2
 - Class Mass Storage, Transp. SCSI, Bulk only
 - Endpoint 1 In Bulk MaxPacket 512
 - Endpoint 2 Out Bulk MaxPacket 512

5.2.2.3 USB Host Test_3: BULK-OUT Transfer

This USB Bulk Out transfer is validated by transferring a file in the RAM memory to the USB device using the u-boot commands. Create a file in the memory using **env export** command.

Peripheral Instance	Command Used	Test Status	Remarks
USB Host	fatwrite usb, fatload usb	Passed	Tested using U-Boot

Execution of test case using Commands:

- **env export <Addr> bootdelay**
- **fatwrite usb 0 <Addr> <file_name> <Len>**
- **fatload usb 0 <Addr> <file_name>**

These are the test commands used to validate the bulk out transfer. As an example you can see **env export** command as above and create a file in the RAM memory. **fatwrite** command is used to write the file from the RAM memory to the USB device (Pen Drive), before we need to copy or create a file in the ram memory location to copy to usb device. Then by using **fatload** command we can copy the file which is copied before to the usb device to the RAM memory. Then by memory display “md” command we can display the content from the memory location where we have copied the file from usb device and can compare the data and validate the usb bulk out transfer.

Console Output:

```
shikhara-uboot>fatwrite usb 0 0x0 sample.txt 0x10
writing sample.txt
16 bytes written
shikhara-uboot>fatload usb 0 0x10 sample.txt
reading sample.txt
16 bytes read
```

5.2.2.4 USB Host Test_4: BULK-IN Transfer

This USB Bulk In transfer is validated by transferring a file in the USB device to the RAM memory



location using the u-boot commands.

Peripheral Instance	Command Used	Test Status	Remarks
USB Host	fatls usb fatload usb	Passed	Tested using U-Boot

Execution of test case using Commands:

- **fatls usb 0**
- **fatload usb 0 <Addr> <file_name>**

These are the test commands used to validate the bulk out transfer. By using the **fatload** we can copy the file from the usb device to the RAM memory location. And by memory display “md” command we can display the content from the memory location where we have copied the file from usb device and can compare the data to the file.

Console Output:

```
shikhara-uboot>fatls usb 0
1382538  moschip-logo-high-resolution.bmp
9453    rksir_usb_cmds
3668    app_alphablending.bin
12      background.raw
.....
12      boot.txt
921654  amar.bmp
18 file(s), 4 dir(s)
shikhara-uboot>fatload usb 0 0x100 moschip-logo-high-resolution.bmp 1382538
reading moschip-logo-high-resolution.bmp
1382538 bytes read
```

5.2.2.5 USB Host controller register dumps

We can see the registerdump values by using the u-boot commands. First by executing the “usb start” command give the “**usbregdump**” command which will display the register contents on the console.

Peripheral Instance	Command Used	Test Status	Remarks
USB Host	usbregdump	Passed	Tested using U-Boot

Execution of test case using Commands:

- usbregdump

Console Output:

```
shikhara-uboot>usbregdump
Anusoc-xhci: init hccr d4300000 and hcor d4300020 hc_length 32
CAPLENGTH AND HCVERSION 0x1000020:
CAPLENGTH: 0x20
HCVERSION: 0x100
HCSPARAMS 1: 0x2000140
  Max device slots: 64
  Max interrupters: 1
  Max ports: 2
HCSPARAMS 2: 0xc0000f1
  Isoc scheduling threshold: 1
  Maximum allowed segments in event ring: 15
HCSPARAMS 3 0x7ff000a:
  Worst case U1 device exit latency: 10
  Worst case U2 device exit latency: 2047
HCC PARAMS 0x220f66c:
  HC generates 32 bit addresses
RTSOFF 0x440:
xHCI operational registers at d4300020:
USBCMD 0x5:
  HC is running
  HC has finished hard reset
  Event Interrupts enabled
  Host System Error Interrupts disabled
  HC has finished light reset
USBSTS 0x18:
```

Event ring is not empty

USBSTS 0x18:

Event ring is not empty

No Host System Error

HC has finished light reset

USBSTS 0x18:

Event ring is not empty

USBSTS 0x18:

Event ring is not empty

No Host System Error

HC is running

port registers :

d4300004 port status reg = 0x2000140

d4300008 port power reg = 0xc0000f1

d430000c port link reg = 0x7ff000a

d4300010 port reserved reg = 0x220f66c

d4300014 port status reg = 0x480

d4300018 port power reg = 0x440

d430001c port link reg = 0x0

d4300020 port reserved reg = 0x5

5.2.3 USB Device Validation

- ➔ USB Device Test_1: Control Transfer
- ➔ USB Device Test_2: BULK OUT Transfer

5.2.3.1 USB Device Test_1: Control Transfer

We can connect the board with a Host-PC using an OTG cable over the USB device port and can perform the Control transfer by using the “lsusb” command on the Host pc.

Peripheral Instance	Command Used	Test Status	Remarks
USB Device	fut(command prompt), lsusb(Host pc)	Passed	Tested using U-Boot

Execution of test case using Commands:

On the Shikhara command prompt

- *fut <Address>*

On the Host PC

- *lsusb*

Console Output:

```
shikhara-uboot>fut 0x100
IP version is 2.60a
Insert a USB cable into the connector!
USB cable Connected![0x4]
Download address 0x00000100
Now, Waiting for USB Host to transmit data
Enumeration success
downloadAddress : 0x100, downloadFileSize: 421981
Download Done!! Download Address: 0x100, Download Filesize:0x67053
```

Console output on Host PC

```
engineer@mcinhvendes44:~/Downloads/USBTOOL$ lsusb
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 007: ID 413c:301a Dell Computer Corp.
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 002: ID 04e8:1234 Samsung Electronics Co., Ltd
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 004: ID 04b3:301b IBM Corp. SK-8815 Keyboard
Bus 003 Device 002: ID 04b3:301a IBM Corp.
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

5.2.3.2 USB Device Test_2: BULK OUT Transfer

We can connect the board with a Host-PC using a OTG cable over the USB device port and can perform Sending a file from Host-PC to Shikara board by using **dltool** utility. Perform Bulk-Out Transfer by



using command **fut**.

Peripheral Instance	Command Used	Test Status	Remarks
USB Device	fut	Passed	Tested using U-Boot

Execution of test case using Commands:

On the Shikhara command prompt

- **fut <Address>**

On the Host PC

- Install Linux Utility dltool (smdk-usbd) in host PC, the files required are under firmware release folder, the path for this is, -> **USBTOOL.zip**.
- Copy the u-boot.bin or any file into the uncompressed folder.
- Copy the executable file smdk-usbd to /opt/toolchains/arm-2010q1/bin.

- **sudo ./smdk-usbd -f <File_name> -a <Address>**

While executing the commands on make sure the same address should be given in both the commands on host pc terminal and u-boot command prompt.

Console Output:

```
shikhara-uboot>fut 0x100
IP version is 2.60a
Insert a USB cable into the connector!
USB cable Connected![0x4]
Download address 0x00000100
Now, Waiting for USB Host to transmit data
Enumeration success
downloadAddress : 0x100, downloadFileSize: 421981
Download Done!! Download Address: 0x100, Download Filesize:0x67053
```

```
shikhara-uboot>md 0x200
00000200: 61766441 6465636e 6d554e20 63697265  Advanced NUmERIC
00000210: 52206c61 61657365 20686372 20646e61  al Research and
00000220: 6c616e41 73697379 6f724720 28207075  Analysis Group (
00000230: 52554e41 20294741 61207369 62616c20  ANURAG) is a lab
00000240: 7461726f 2079726f 7420666f 44206568  oratory of the D
00000250: 6e656665 52206563 61657365 20686372  efence Research
00000260: 20646e61 65766544 6d706f6c 20746e65  and Development
00000270: 6167724f 6173696e 6e6f6974 52442820  Organisation (DR
00000280: 2e294f44 636f4c20 64657461 206e6920  DO). Located in
00000290: 636e614b 626e6168 2c686761 64794820  Kanchanbagh, Hyd
000002a0: 62617265 202c6461 69207469 6e692073  erabad, it is in
000002b0: 766c6f76 74206465 64206568 6c657665  volved the devel
000002c0: 656d706f 6f20746e 6f632066 7475706d  opment of comput
```

```
000002d0: 20676e69 756c6f73 6e6f6974 6f662073   ing solutions fo
000002e0: 756e2072 6972656d 206c6163 6c616e61   r numerical anal
000002f0: 73697379 646e6120 65687420 75207269   ysis and their u
```

Output log on Host PC:

```
engineer@mcinhvendes02:~$ cd USBTOOL/
engineer@mcinhvendes02:~/USBTOOL$ ls
bulk1.bin bulk3.bin bulk.bin dltool.c libusb-1.0.9.tar.bz2 libusb-compat-0.1.4.tar.bz2 readme.txt u-boot.bin
bulk2.bin bulk4.bin bulk.txt libusb-1.0.9 libusb-compat-0.1.4 Makefile smdk-usbd1
engineer@mcinhvendes02:~/USBTOOL$ cat bulk.bin
Advanced NUmberical Research and Analysis Group (ANURAG) is a laboratory of the Defence Research and
Development Organisation (DRDO). Located in Kanchanbagh, Hyderabad, it is involved the development of computing
solutions for numerical analysis and their use in other DRDO projects.
engineer@mcinhvendes02:~/USBTOOL$ sudo ./smdk-usbd1 -f bulk.bin -a 0x200
SMDK42XX,S3C64XX USB Download Tool
Version 0.20 (c) 2004,2005,2006 Ben Dooks <ben-linux@fluff.org>
S3C64XX Detected!
=> found device: bus 001, dev 008
=> loaded 283 bytes from bulk.bin
=> Downloading 293 bytes to 0x00000200
Len: 285
=> Data checksum 650a
=> usb_bulk_write() returned 293
Version 0.20 (c) 2004,2005,2006 Ben Dooks <ben-linux@fluff.org>
```

5.2.4 DMA Validation

5.2.4.1 DMA-2 Test – Memory to Memory transfer using DMA

Memory to Memory transfer using DMA by hardcoded (min) code

Call “**dmamemtest**” to test DMAC2 for Memory to Memory transfer. If data of both buffer is same test case is PASSED.

Peripheral Instance	Command Used	Test Status	Remarks
DMA-2	dmamemtest	Passed	Tested using U-Boot

Execution of test case using Commands:

- **dmamemtest**

Console Output:

```
shikhara-uboot>dmamemtest
DMA Test: Perform Non Secure Memory to Memory Test from address 0x10 and to 0x2000
Print the register value before Starting Non Secure Memory to Memory Transfers
Starting Non Secure Memory to Memory Transfers
Print the register value after Non Secure Memory to Memory Transfers
PASSED: Non-Secure DMA interrupt test
PASSED: Non-Secure M2M DMA transfer test

PASSED: Non Secure DMA test
DMA Test: Perform SRAM Memory to DDR Memory Test from address 0x10 and to 0xC4000000
Print the register value before Starting Secure Memory to Memory Transfers
Starting Secure Memory to Memory Transfers
Print the register value after Secure Memory to Memory Transfers
PASSED: DDR to SRAM memory test with Secure DMA

PASSED: DDR to SMC SRAM Memory test

DMA Test: Perform Secure Memory to Memory Test from address 0x10 and to 0x2000
Print the register value before starting Secure Memory to Memory Transfers
Starting Secure Memory to Memory Transfers
Print the register value after Secure Memory to Memory Transfers
PASSED: Secure DMA interrupt test
PASSED: Secure DMA transfer test

PASSED: Secure DMAC test
DMA Test is Successfull
```

Memory to Memory transfer using DMA by api code (using polling for interrupts method)

Call **dma_m2m** to test DMAC2 for Memory to Memory transfer. Make sure flag "CONFIG_PL330_INTERRUPT_MODE" should undefine (in anusoc.h) at the time of uboot compilation. If data of both buffer is same test case is PASSED.

Peripheral Instance	Command Used	Test Status	Remarks
DMA-2	dma_m2m	Passed	Tested using U-Boot

Execution of test case using Commands:

- **dma_m2m**

Console Output:

```
shikhara-uboot>dma_m2m
SHIKARA DMA M2M Test: bufferSRC - 3ebc0960 memory size : 100
SHIKARA DMA M2M Test: buffer DST - 3ebc09c8 memory size : 100
PL330: PERIPH_ID 0x141330, PCELL_ID 0xb105f00d !
PL330: Number of channels for DMAC: 8
PL330: micro code buffer size = 512
PL330: allocated coherent memory for DMAC! size: 4096
PL330: mcode_bus: 1052511024 mcode_cpu : 3ebc0b30
SHIKARA-PL330 DBUFF-128x16bytes Num_Chans-8 Num_Peri-2 Num_Events-2
DMAC2 is initialised..
SHIKARA-PL330: channel request is successful DMAC
SHIKARA-PL330: channel got to submit request for DMAC 1D
SHIKARA-PL330:call chan_ctrl to start DMAC: 0
PL330: address in emitGo = 3ebc0b30
PL330:Set to generate interrupts for SEV
SHIKARA-PL330: Now poll for interrupts
PL330:interrupt occured DMAC
Manager Status is 0x0
PL330: manager thread status 0
PL330: Value channel of FSC 0
PL330: Reset Channel-0 CS-0 FTC-0 FTC-0
PL330: Channel PC value 3ebc0b4d
PL330: Value channel of interrupt ES 1 and status 1
PL330: Status of INTEN interrpts
PL330: no requests for channel
SHIKARA_DMA_TEST: Both buffers are having same data
SHIKARA_DMA_TEST : Xfer is successful and Test is exiting
PL330:Channel is going to stop
```

5.3 TEST CASES FOR LOW-SPEED PERIPHERALS

5.3.1 UART Validation

5.3.1.1 UART Test1: UART test Case

This test case validates basic functionality of ARM PL011 UART. It is configured to print the messages on the console. Console messages indicate that the UART is able to print messages and hence its validation is said to be successful.

Console Output:

```
U-Boot 2012.04.01 (Oct 17 2017 - 20:49:34)

DRAM: 1 GiB

WARNING: Caches not enabled

SF: Detected M25P128 with page size 256, total 16 MiB
SF: Detected M25P128 with page size 256 KiB, total 16 MiB
```



```
fb_size for HDMI is 0x e1000
In:  serial
Out: serial
Err: serial
Net: Hit any key to stop autoboot_sh: 0
shikhara-uboot>
shikhara-uboot>
```

5.3.1.2 UART Test 2: LOOPBACKTEST

This test case validates basic functionality of ARM PL011 UART. It is configured to print the messages on the console. Console messages indicate that the UART is able to print messages and hence its validation is said to be successful.

Peripheral Instance	Command Used	Test Status	Remarks
UART	uartloopback_test	Passed	Tested using U-Boot

Execution of test case using Commands:

- **uartloopback_test <Uart_Instance>**

This is the command used to test loopback test for the instance number specified as a second argument and will display the test result on the console output.

Console Output:

```
shikhara-uboot> uartloopback_test 1
the uart loopback test pass
shikhara-uboot> uartloopback_test 2
the uart loopback test pass
shikhara-uboot> uartloopback_test 3
the uart loopback test pass
```

5.3.2 I2C Validation

- **I2C Test_1:** To test I2C speed and slave device connected to I2C.
- **I2C Test_2:** To test read and write operations on EEPROM using I2C-0.
- **I2C Test_3** To test read and write operations on SGTL5000 using I2C-1

➤ **I2C Test_4 To test read and write operations on Camera sensor using I2C-2**

5.3.2.1 I2C Test_1: To test I2C speed and slave device connected to I2C

I2C was tested using inbuilt U-Boot commands to check the speed and whether the device is attached and to change the speed using the commands listed below.

Peripheral Instance	Command Used	Test Status	Remarks
I2C	I2c dev, i2c probe, i2c speed	Passed	Tested using U-Boot

Execution of test case using Commands:

- **i2c dev**
- **i2c probe**
- **i2c speed**

Check the speed of the I2C, using **i2c speed** command and I2C speed can be changed using **i2c speed [speed]**. Check the devices attached to the i2c using **i2c probe**, this command returns devices connected to I2C.

Console Output:

```
shikhara-uboot>i2c dev 0
Setting bus to 0
shikhara-uboot>i2c speed
Current bus speed=400000
shikhara-uboot>i2c speed 100000
Setting bus speed to 100000 Hz
shikhara-uboot>i2c speed
Current bus speed=100000
shikhara-uboot>i2c probe
Valid chip addresses: 0A
```

5.3.2.2 I2C Test_2: To test read and write operations on EEPROM using I2C-0

This test case is used for doing read and write operations on the slave EEPROM using the i2c-0

instance.

Peripheral Instance	Command Used	Test Status	Remarks
I2C	i2c dev 0, i2c_eeprom_test	Passed	Tested using U-Boot

Execution of test case using Commands:

- i2c dev 0
- i2c_eeprom_test

This are the test commands used to test the read and write operations on EEPROM .

Console Output:

```
shikhara-uboot>i2c dev 0
Setting bus to 0
shikhara-uboot>i2c_eeprom_test
This is EEPROM Interfaced on I2C, Address=0x50
Going to write to device location=0x0 of data=0x55, length=256
Going to Read from device location=0x0
I2C EEPROM Write and Read Success of data=0x55
```

5.3.2.3 I2C Test_3 To test read and write operations on SGTL5000 using I2C-1

This test case is used for doing read and write operations on the SGTL5000 using the i2c-1 instance.

Peripheral Instance	Command Used	Test Status	Remarks
I2C	I2c_start, i2c dev, I2c_mintest	Passed	Tested using U-Boot

Execution of test case using Commands:

- i2c_start
- i2c dev 1
- i2c_mintest <Dev_addr> <Addr> <Data> <Count>

Console Output:

```
shikhara-uboot>i2c dev 1
Setting bus to 1
shikhara-uboot>i2c probe
Valid chip addresses: 0A
shikhara-uboot>i2c_mintest 0x0a 0x0026.2 0x0004 0x2
i2c test is successfully completed
```

5.3.2.4 I2C Test_4 To test read and write operations on Camera sensor using I2C-2

This test case is used for doing read and write operations on Camera Sensor using I2C-2

Peripheral Instance	Command Used	Test Status	Remarks
I2C	i2c mw, i2c md	Passed	Tested using U-Boot

Execution of test case using Commands:

- i2c mw {i2c_chip} {devaddr} {data} {len}
- i2c md {i2c_chip} {devaddr} {len}

Use i2c md Command to read data from Camera Sensor and use i2c mw Command to write data into Camera Sensor. Use caminit Command to check camera sensor register configuration using i2c.

Console Output:

```
shikhara-uboot>i2c dev 2
Setting bus to 2
shikhara-uboot>i2c probe
Valid chip addresses: 3C
shikhara-uboot>i2c md 0x3c 0x300a.2 0x1
300a: 56 V
shikhara-uboot>i2c md 0x3c 0x300b.2 0x1
300b: 40 @
shikhara-uboot>i2c mw 0x3c 0x3008.2 0x82 0x1
shikhara-uboot>i2c md 0x3c 0x3008.2 0x1
3008: 82 .
```

5.3.3 RTC Validation

To test RTC device, use date command to get the current date and time which will be displayed on the console output.

Peripheral Instance	Command Used	Test Status	Remarks
RTC	Date	Passed	Tested using U-Boot

Execution of test case using Commands:

- **date MMDDHHMMCCYY.SS**

The above command will be used to set the date and time, and type the “**date**” will be displaying the updated date and time on the console.

Console Output:

```
shikhara-uboot>date
Date: 1970-01-01 (Thursday) Time: 0:00:00
shikhara-uboot>date 092010002017.09
Date: 2017-09-20 (Wednesday) Time: 10:00:09
shikhara-uboot>date
Date: 2017-09-20 (Wednesday) Time: 10:00:09
shikhara-uboot>
```

5.3.4 I2S Validation

5.3.4.1 I2S Test_1: Verify the Sound driver components (I2s and Codec) working status on the board

Initialise the sound driver using sound init command. Sound driver will display the status messag on the console, After processing **sound init** command.

Peripheral Instance	Command Used	Test Status	Remarks
I2S	i2c_start, sound init	Passed	Tested using U-Boot

Execution of test case using Commands:

- **i2c_start**
- **sound init**

This are the test commands used to validate and verify the sound driver component status .

Console Output:

```
shikhara-uboot>i2c_start
I2c is initializing
I2C
shikhara-uboot>sound init
SGTL5000 CODEC CHIP ID: a011
REGISTERS
```

5.3.4.2 I2S Test_2: Verify functioning of I2S at different sampling frequencies using different Sound commands

This test case is validated using the test commands first use sound init command to initialise the sound. Use different sound commands for testing I2S audio.

Peripheral Instance	Command Used	Test Status	Remarks
I2S	sound square, sound sine, sound wave	Passed	Tested using U-Boot

Execution of test case using Commands:

- **sound square [msec]** – play sound as square wave for ,msec' duration
- **sound sine** – play sound resembling a sine wave
- **sound wave [addr]** – play sound resembling a sine wave

These are the test commands used to validate the i2s sound. “**sound wave**” command is used to validate the i2s by playing recording which is at the address which will be given as a third



argument in this command.

For this “**sound wave**” command, first need to copy the recordings from the SD card to the Ram location. And by specifying the address in the sound wave command as third argument will play the copied recording from that ram location.

Console Output:

```
shikhara-uboot>i2c_start
I2c is initializing
I2C
shikhara-uboot>sound init
SGTL5000 CODEC CHIP ID: a011
REGISTERS
shikhara-uboot>sound sine
Play sinewave
PLAYBACK STARTED

PLAYBACK COMPLETED
shikhara-uboot>fatload mmc 0 0x02500000 shape_of_you.wav
Dwmci_init Host base address = 0xd4555000
reading test.wav
44878832 bytes read
shikhara-uboot>sound wave 0x02500000
Play Wavefile
    AudioFormat = 1
    NumChannels = 2
    SampleRate = 48000
    ByteRate = 192000
    BlockAlign = 4
    BitsPerSample = 16
    Size of wav file is 44878788
PLAYBACK STARTED

PLAYBACK COMPLETED
```

5.3.4.3 I2S Test_3: Verify I2S recording functionality

Initialise the sound driver using sound init command. Use sound record commands for testing I2S recordings. After recording , can playback the recording through the command and validate the i2s recording functionality.

Peripheral Instance	Command Used	Test Status	Remarks
I2S	sound init sound record sound wav	Passed	Tested using U-Boot

Execution of test case using Commands:

- **sound init**
- **sound record <Address>**
- **sound wav <Address>**

This are the test commands use to validate. By using the **sound record** command we can record at a specified address which is given as a third argument in that command. And can playback the recordings using **sound wav** from the address where recording is stored by specifying that address as a third argument.

Console Output:

```
shikhara-uboot>i2c_start
I2c is initializing
I2C: shikhara-uboot>
shikhara-uboot>sound init
SGTL5000 CODEC CHIP ID: a011
REGISTERS
shikhara-uboot>sound record 0x0
Recording START
-----Recorded WAVE file-----
AudioFormat = 1
NumChannels = 2
SampleRate = 48000
ByteRate = 192000
BlockAlign = 4
BitsPerSample = 16
```



```

Size of wav file is 1536000
shikhara-uboot>sound wave 0x0
Play Wavefile
    AudioFormat = 1
    NumChannels = 2
    SampleRate = 48000
    ByteRate = 192000
    BlockAlign = 4
    BitsPerSample = 16
    Size of wav file is 1536000
PLAYBACK STARTED

PLAYBACK COMPLETED

```

5.3.4.4 I2S controller register dumps

Initialise the sound driver using sound init command. Dump I2S controller registers by running **i2sdump** command on console.

Peripheral Instance	Command Used	Test Status	Remarks
I2S	i2sdump	Passed	Tested using U-Boot

Execution of test case using Commands:

- **i2sdump**

By using this command can display register dump of i2s on the console output.

Console Output:

```

shikhara-uboot>i2sdump
*****Dumping I2S Registers*****
IER:0x0
IRER:0x0
ITER:0x0
CER:0x0
CCR:0x10
TXFFR:0x0
TER0:0x1
TCR0:0x5
ISR0:0x10
IMR0:0x33
TFCR0:0x8
TFF0:0x0

```

5.3.5 CAN Validation

- CAN Test_1: Start/Stop CAN Controller
- CAN Test_2: Send CAN Frames in Standard Format
- CAN Test_3: Receive CAN Frames in Standard Format
- CAN Test_4: Send CAN Frames in Extended Format
- CAN Test_5: Receive CAN Frames in Extended Format

5.3.5.1 CAN Test_1: Start/Stop CAN Controller

This test case is validated Start/Stop CAN Controller using built in 'can' commands .Before need to Connect two CAN boards one for Transmitting and other for Receiving. Check for the Debug messages on console.

Peripheral Instance	Command Used	Test Status	Remarks
CAN	can_start can init	Passed	Tested using U-Boot

Execution of test case using Commands:

- *can_start*
- *can init*

These are the test commands used to initialize the CAN controller.

Console Output:

```
shikhara-uboot> can_start
can is initializing
CAN: SHIKHARA_CAN at 0xd4587000 registered
shikhara-uboot> can init
Initialise CAN
Initializing CAN at 0xd4587000 with baudrate 125K
```

5.3.5.2 CAN Test_2: Send CAN Frames in Standard Format

This test case is validated using built in 'can' commands .Before need to Connect two CAN boards one for Transmitting and other for Receiving. Check for the Debug messages on console. By using the test commands will transmit data to the other CAN.

Peripheral Instance	Command Used	Test Status	Remarks
CAN	can xmit	Passed	Tested using U-Boot

Execution of test case using Commands:

- `can xmit <id> <dlc> <do> <d1><d7>`

This is test command **can xmit** used to transmit the data to other CAN. Second argument is the id and third argument is dlc, and remaining arguments are the data to be transmitted.

Console Output:

```
shikhara-uboot> can xmit 0x1 0x4 0x11 0x22 0x33 0x44
Transmitting id 0x1 dlc 4
Transmission OK
```

5.3.5.3 CAN Test_3: Receive CAN Frames in Standard Format

This test case is validated using built in 'can' commands .Before need to Connect two CAN boards one for Transmitting and other for Receiving. Check for the Debug messages on console. By using the test commands will receive data to the other CAN.

Peripheral Instance	Command Used	Test Status	Remarks
CAN	can rcv	Passed	Tested using U-Boot

Execution of test case using Commands:

- `can rcv`

This is the test command used to receive the data from the other CAN controller. As in the previous test case transmitted the data and now will receive the same data from that CAN by using this command.

Console Output:

```
shikhara-uboot> can rcv
<0x1> [4] 11 22 33 44
Reception OK
```

5.3.5.4 CAN Test_4: Send CAN Frames in Extended Format

This test case is validated using built in 'can' commands .Before need to Connect two CAN boards one for Transmitting and other for Receiving. Check for the Debug messages on console. By using the test commands will transmit data to the other CAN.

Peripheral Instance	Command Used	Test Status	Remarks
CAN	can xmit	Passed	Tested using U-Boot

Execution of test case using Commands:

- `can xmit <id> <dlc> <do> <d1><d7>`

This is test command **can xmit** used to transmit the data to other CAN. Second argument is the id and third argument is dlc, and remaining arguments are the data to be transmitted.

Console Output:

```
shikhara-uboot> can xmit 0x91023045 0x8 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
Transmitting id 0x9102305 dlc 8
Transmission OK
```

5.3.5.5 CAN Test_5: Receive CAN Frames in Extended Format

This test case is validated using built in 'can' commands .Before need to Connect two CAN boards one for Transmitting and other for Receiving. Check for the Debug messages on console. By using the test commands will receive data to the other CAN.

Peripheral Instance	Command Used	Test Status	Remarks
CAN	can recv	Passed	Tested using U-Boot

Execution of test case using Commands:

- `can recv`

This is the test command used to receive the data from the other CAN controller. As in the previous test case transmitted the data and now will receive the same data from that CAN by using this command.

Console Output:

```
shikhara-uboot> can recv
<0x11023045> [8] 11 22 33 44 55 66 77 88
Reception OK
```

5.3.6 KMI (PS2) Validation

- ➔ KMI Test_1 PS2 keyboard Input and Serial console Output test.
- ➔ KMI Test_2 PS2 mouse Input and serial console Output test.

5.3.6.1 KMI Test_1 PS2 keyboard Input and Serial console Output test

This test case is validated by connecting the Shikhara board with a PS/2 keyboard. Perform Keyboard-Test by using command apkybd. Then will ask to enter the keys from the connected keyboard and will display the



keys entered on the console.

Peripheral Instance	Command Used	Test Status	Remarks
KMI	apkybd	Passed	Tested using U-Boot

Execution of test case using Commands:

- **apkybd**

This is the test command used to validate the ps2 keyboard as a input to the board and entered keys from this connected keyboard will be displayed on serial console.

Console Output:

```
shikhara-uboot> apkybd
Please plug a Keyboard into PS/2 port (bottom)
reset done=27
Please type on the Keyboard (use Space Bar to exit) :
Key : h
Key : s
Key :
Key : y
Key : u
Key : l
Key :
Test Result is : Success
```

5.3.6.2 KMI Test_2 PS2 mouse Input and serial console Output test.

This test case is validated by connecting the Shikhara board with a PS/2 mouse. Perform Mouse-Test by using command apmouse.

Peripheral Instance	Command Used	Test Status	Remarks
KMI	apmouse	Passed	Tested using U-Boot

Execution of test case using Commands:

- **apmouse**

This is the test command used to validate the ps2 mouse connected to the board and will display the mouse X, Y and button values on the serial console.

Console Output:

shikhara-uboot> *apmouse*

Please plug a Mouse into J16 (top)
The test displays the mouse X, Y and button values.
Use the right button to exit the test

```
stat 1 is fa
stat 2 is aa
reset done=97
stat 3 is fa
enable done=97
MouseX:-1, MouseY: 0, Buttons: 0
MouseX:-1, MouseY:-2, Buttons: 0
MouseX:-1, MouseY:-7, Buttons: 0
MouseX: 1, MouseY:-20, Buttons: 0
MouseX: 14, MouseY:-35, Buttons: 0
MouseX: 35, MouseY:-42, Buttons: 0
MouseX: 50, MouseY:-42, Buttons: 0
MouseX: 64, MouseY:-41, Buttons: 0
MouseX: 88, MouseY:-47, Buttons: 0
```

5.3.7 *SPI Validation*

➔ **SPI Test_1 – Write & Read operations on SPI Flash**

➔ **SPI Test_2 – Touch Screen controller**

5.3.7.1 *SPI Test_1 – Write & Read operations on SPI Flash*

This test case is validated by writing some known patterns on RAM locations by using “mw” command and then copying the data from this RAM locations to the Flash memory using the “sf” commands and copying flash data to the other RAM locations and compare the this data with the source RAM locations data which should be same and hence this test case is passed if data is same.

Peripheral Instance	Command Used	Test Status	Remarks
SPI	sf probe, sf erase, sf read, sf write, sf protect lock/unlock	Passed	Tested using U-Boot

Execution of test case using Commands:



- ***sf probe*** **[[bus:]cs] [hz] [mode]**
- ***sf erase*** **<offset|partition> <len>**
- ***sf read*** **<addr> <offset|partition> <len>**
- ***sf write*** **<addr> <offset|partition> <len>**
- ***sf protect*** **<lock/unlock> <sector> <len>**

These are the test commands used to validate the read and write operations of flash. ***sf probe*** command will initialize flash device on given SPI bus and chip select. ***sf erase*** will erase `len' bytes from `offset' or from start of mtd `partition'. ***sf read*** read `len' bytes starting at `offset' or from start of mtd `partition' to memory at `addr'. ***sf write*** write `len' bytes from memory at `addr' to flash at `offset' or to start of mtd `partition'. ***sf protect*** protect/unprotect `len' bytes starting at address `sector'.

Console Output:

```
shikhara-uboot> sf probe 0 0 0
```

```
SF: Detected M25P128 with page size 256, total 16 MiB
```

```
SF: Detected M25P128 with page size 256 KiB, total 16 MiB
```

```
shikhara-uboot> sf erase 0x0 0x40000
```

```
shikhara-uboot> md 0x200 0x10
```

```
40000000: 00001234 00001234 00001234 00001234  4...4...4...4...
40000010: 00001234 00001234 00001234 00001234  4...4...4...4...
40000020: 00001234 00001234 00001234 00001234  4...4...4...4...
40000030: 00001234 00001234 00001234 00001234  4...4...4...4...
```

```
shikhara-uboot> sf write 0x200 0x0 0x10
```

```
shikhara-uboot> md 0x300 0x10
```

```
41000000: d2603fbd 17b701d5 ed7f8fdf 0dffbbde  ?`.....
41000010: fd6faa5e e66cf2d7 f86dbf3d ad5fd6a7  ^_o...l.=.m..._.
41000020: d06bd244 7d4dcf82 d0a19a0a 6d07e297  D.k...M}.....m
41000030: 34290f01 af076a85 992de13d 2d0f3802  ...) 4.j...=-...8.-
shikhara-uboot> mw 0x41000000 0x1234 0x10
```

```
shikhara-uboot> sf read 0x300 0x0 0x10
```

```
shikhara-uboot> md 0x300
```

```
41000000: 00001234 00001234 00001234 00001234  4...4...4...4...
41000010: 00001234 00001234 00001234 00001234  4...4...4...4...
41000020: 00001234 00001234 00001234 00001234  4...4...4...4...
41000030: 00001234 00001234 00001234 00001234  4...4...4...4...
```

5.3.7.2 SPI Test_2 – Touch Screen controller

To validate this test case enable CONFIG_TC in anusoc.h and then it will automatically compile touch screen support and LCD panel should be connected.

Peripheral Instance	Command Used	Test Status	Remarks
SPI	ts print, ts calibrate	Passed	Tested using U-Boot

Execution of test case using Commands:

- *ts print*
- *ts calibrate*

These are the test commands used to validate this test case. *ts print* command will gives positions where you touched on LCD and *ts calibrate* command will calibrate the touch screen controller.

Console Output:

```
shikhara-uboot>lcd_enable
shikhara-uboot>ts print
SPI Initialised..
....Please touch the LCD Screen....
X:= FE6 Y:= 707 Rt=0
....Please touch the LCD Screen....
X:= A30 Y:= 980 Rt=0
....Please touch the LCD Screen....
X:= DC0 Y:= 8C4 Rt=0
....Please touch the LCD Screen....
X:= AFF Y:= 8BD Rt=0
*****Exiting Touch Screen Test*****
shikhara-uboot>ts calibrate
SPI Initialised..
Please touch at position 1 to sample
....Please touch the LCD Screen....
Calibration points are x= D80 and y= 72F
Please touch at position 2 to sample
....Please touch the LCD Screen....
Calibration points are x= C28 and y= 853
Please touch at position 3 to sample
....Please touch the LCD Screen....
Calibration points are x= BE6 and y= 790
Please touch at position 4 to sample
....Please touch the LCD Screen....
Calibration points are x= B3F and y= AC0
X- slope is -236 and Y- slope is 1685
Touch to take Sample for testing
....Please touch the LCD Screen....
Calibration Completed...
*****Exiting Touch Screen Test*****
```


5.4 TEST CASES FOR DISPLAY PERIPHERALS

5.4.1 HDMI Validation

The following are the test cases used for validating the HDMI.

- HDMI Test_1 – Display Color on HDMI (DVI Mode)
- HDMI Test_2 – Display Image on HDMI
- HDMI Test_3 – HDMI Audio Test Case
- HDMI Test_4 – HDMI BMP Slideshow Test Case
- HDMI Test_5 – HDMI RGB Videoplayer Test Case
- HDMI Test_6 – Test HDMI Console

5.4.1.1 HDMI Test_1 – Display Colour on HDMI (DVI Mode)

This test case is validated by display the selected colour pattern on the HDMI screen, for this need to connect the hdmi panel and board using cable and executing the u-boot command will display the colour on HDMI screen.

Peripheral Instance	Command Used	Test Status	Remarks
HDMI	hdmi_enable, hdmi_dvi_min	Passed	Tested using U-Boot

Execution of test case using Commands:

- **hdmi_enable**
- **hdmi_dvi_min**

These are the commands to be executed to display the colour. “**hdmi_enable**” will enable the hdmi and then by using “**hdmi_dvi_min**” will give you a menu to choose the colour to display, by selecting any one of the colours available, then will be able to see colour displayed on the HDMI screen.

Console Output:

```
shikhara-uboot>hdmi_enable
HDMI Panel Detected
Shikhara Display controller initilized successfully
HDMI configured Successfully
shikhara-uboot>hdmi_dvi_min
```

```

Enter your color choice to fill screen
B for Blue
G for Green
R for Red
W for white
X for Exit
choice is r

```

5.4.1.2 HDMI Test_2 – Display Image on HDMI

This test case is validated by displaying the image on the HDMI. Before need to Test BMP Image (24bpp), first we need to copy image from sd card to the RAM and display on HDMI console using command.

Peripheral Instance	Command Used	Test Status	Remarks
HDMI	hdmi_enable, mmc_start, fatload mmc, bmp display	Passed	Tested using U-Boot

Execution of test case using Commands:

- **hdmi_enable**
- **mmc_start**
- **fatload mmc 0 <Address> <.bmp_file>**
- **bmp display <Address> {x y}**

This are the commands used to display the image on the HDMI. First through **hdmi_enable** enable the hdmi and initialise the MMC using **mmc_start** and copy the image (.bmp) from the SD card to the RAM memory using **fatload** command. And by using the **bmp display** command will display the image on HDMI screen.

Console Output:

```

shikhara-uboot>mmc_start
shikhara-uboot>fatls mmc 0
921738 1.bmp
921738 2.bmp

```

```

921738 3.bmp
921738 5.bmp
921738 4.bmp
10 file(s), 2 dir(s)
shikhara-uboot>fatload mmc 0 0x01500000 1.bmp
reading 1.bmp
shikhara-uboot>bmp display 0x01500000
will display the image on hdmi..

```

5.4.1.3 HDMI Test_3: HDMI Audio Test Case

To validate the HDMI audio test case, need to copy the .wave file to the RAM location from the SD card. And by using the test commands will play the audio on HDMI.

Peripheral Instance	Command Used	Test Status	Remarks
HDMI	hdmi wave	Passed	Tested using U-Boot

Execution of test case using Commands:

- **fatload mmc 0 0x02500000 <.wave file>**
- **hdmi wave <Addr>**

This is the test command to be executed to play the audio on HDMI Audio from the address specified as a second argument in this command.

Console Output:

```

shikhara-uboot>fatload mmc 0 0x02500000 sia_common_common.wav
shikhara-uboot>hdmi wave 0x02500000
Play Wavefile
    AudioFormat = 1
    NumChannels = 2
    SampleRate = 48000
    ByteRate = 192000
    BlockAlign = 4
    BitsPerSample = 16
    Size of wav file is 34080392
hw_buffer is 6400000
HDMI audio playback completed

```

5.4.1.4 HDMI Test_4: HDMI BMP Slideshow Test Case

To validate this BMP Slideshow on the HDMI, first need to copy the .bmp images on to the RAM locations from the SD/MMC card and by using the u-boot test command we can run the slideshow on the hdmi.

Peripheral Instance	Command Used	Test Status	Remarks
HDMI	bmplideshow_help, mmc_start, fatload mmc, bmplideshow	Passed	Tested using U-Boot

Execution of test case using Commands:

- **bmplideshow_help**
- **mmc_start**
- **fatload mmc 0 <Address> <.bmp_file>**
- **bmplideshow <time delay b/n two slide (in ms) > <no of loop>**

This are the commands used to play slideshow on hdmi. We need to copy the .bmp images on to the RAM locations. Here it is the command usage of bmplideshow_help and showing the RAM address locations to where we need to copy the bmp images. After copying images through the “**fatload**” command and by running the “**bmplideshow**” will display the one by one images on the HDMI will run 20 times in loop by default and need to specify the time delay as a second argument.

Console Output:

```
shikhara-uboot>bmplideshow_help
```

First use MMC commands to copy 5 BMP files in below addresses. Then run bmplideshow command

Use : bmplideshow <time delay b/n two slide(in ms)> <no of loop>

Use : If no of loop is -1 it will run as infinite loop(use ctrl+c for break. default time delay and number of loop is 500ms and 20

BMP_SLIDESHOW_1 = 8c00000 , FB_SLIDESHOW_1= 6400000

BMP_SLIDESHOW_2 = 9200000 , FB_SLIDESHOW_2= 6a00000

BMP_SLIDESHOW_3 = 9800000 , FB_SLIDESHOW_3= 7000000

BMP_SLIDESHOW_4 = 9e00000 , FB_SLIDESHOW_4= 7600000

BMP_SLIDESHOW_5 = a400000 , FB_SLIDESHOW_5= 7c00000

```
shikhara-uboot>fatload mmc 0 0x8c00000 1.bmp
```

reading 1.bmp

```
shikhara-uboot>fatload mmc 0 0x9200000 2.bmp
```

```
shikhara-uboot>fatload mmc 0 0x9800000 3.bmp
shikhara-uboot>fatload mmc 0 0x9e00000 4.bmp
shikhara-uboot>fatload mmc 0 0xa400000 5.bmp
shikhara-uboot>bmpslideshow 1500
Delay value is 1500000 ms
loop will run 20 times
inside bmp_slideshow
Converted first file from BMP to BGR
Showing first BMP file in slideshow, ..
Count is 19 of 14, BMP_SLIDESHOW_1 = 8c00000 , FB_SLIDESHOW_1= 6400000
Showing second BMP file in slideshow, ..
Count is 19 of 14, BMP_SLIDESHOW_2 = 9200000 , FB_SLIDESHOW_2= 6a00000
Showing third BMP file in slideshow, ..
.....
Count is 0 of 14, BMP_SLIDESHOW_4 = 9e00000 , FB_SLIDESHOW_4= 7600000
Showing fifth BMP file in slideshow, ..
Count is 0 of 14, BMP_SLIDESHOW_5 = a400000 , FB_SLIDESHOW_5= 7c00000
Done with the slideshow, ..
```

5.4.1.5 HDMI Test_5: HDMI RGB/YUV420 Video player Test Case

This test case is validated by playing the video on the HDMI before need to copy the RGB video to the RAM location. Then by running the test command will play the video on the HDMI.

Peripheral Instance	Command Used	Test Status	Remarks
HDMI	rawvideoplayer_help, mmc_start, fatload mmc rawvideoplayer hdmi_disable	Passed	Tested using U-Boot

Execution of test case using Commands:

- **rawvideoplayer_help**
- **mmc_start**



- **fatload mmc 0 <Address> <.bmp_file>**
- **rawvideoplayer <width> <hight> <no of frames> <type-yuv420/rgb>**
- **hdmi_disable**

The command usage of rawvideoplayer_help will display the RAM address to where will copy the video on to that location. First use MMC commands to copy RGB video files in 0x1500000 addresses. Then run rawvideoplayer command will play the video on the HDMI.

use: rawvideoplayer <width> <hight> <no of frames> <type-yuv420/rgb>

Console Output:

```
shikhara-uboot>rawvideoplayer_help
First use MMC commands to copy RGB video files in 0x1500000 addresses. Then run rawvideoplayer
command
use : rawvideoplayer <width> <hight> <no of frames> <type-yuv420/rgb>
shikhara-uboot>fatload mmc 1 0x1500000 video_raw_176x144_24bpp_149frames
Dwmci_init Host base address = 0xd4555000
reading video_raw_176x144_24bpp_149frames
11328768 bytes read
shikhara-uboot>rawvideoplayer 176 144 149 rgb
shikhara-uboot>hdmi_disable
```

5.4.1.6 HDMI Test_6: Test HDMI Console

This test is validated by displaying the console output on the HDMI screen. To test this need to enable the macro “**CONFIG_HDMI_CONSOLE**” in anusoc.h file.

Peripheral Instance	Command Used	Test Status	Remarks
HDMI	coninfo, set stdout hdmi	Passed	Tested using U-Boot

Execution of test case using Commands:

- **coninfo**
- **set stdout hdmi**

This are the test commands used to redirect the console output to the HDMI. Check list of standard input and output devices using command **coninfo**. To set HDMI device as a output device using **stdout** environment variable, after assigning stdout as hdmi, console output will be redirected to HDMI panel. Check the debug messages on HDMI console.

Console Output:

```
shikhara-uboot>coninfo
```

```
List of available devices:
```

```
lcd 00000002 ..O
```

```
serial 80000003 SIO stdin stdout stderr
```

```
hdmi 00000002 ..O
```

```
shikhara-uboot>set stdout hdmi
```

5.4.2 CSI Validation

5.4.2.1 Displaying live streaming on HDMI

This test case is validated by streaming live video on HDMI. By using i2c will configure the CSI and will first enable the HDMI and then it starts displaying the streaming on hdmi.

Peripheral Instance	Command Used	Test Status	Remarks
CSI	hdmi_enable, i2c dev 2, i2c probe, caminit	Passed	Tested using U-Boot

Execution of test case using Commands:

- **hdmi_enable**
- **i2c dev 2**
- **i2c probe**
- **caminit**
- **preview_on_hdmi**

These are the test commands used enable the hdmi and i2c and will display the streaming on HDMI.



When executing the **caminit** command will initialize the CSI execute **preview_on_hdmi** will start the streaming and displays preview on hdmi. Then this live streaming will be displayed on the HDMI panel.

Console Output:

```
shikhara-uboot>hdmi_enable
HDMI Panel Detected
Shikhara Display controller initilized successfully
HDMI configured Successfully
shikhara-uboot>i2c dev 2
Setting bus to 2
shikhara-uboot>i2c probe
Valid chip addresses: 3C
shikhara-uboot>caminit
MIPI-CSI2 Version is: 825242666
Info: ov5640 chip id high byte: 56
Info: ov5640 chip id low byte: 40
Cam init is successfull
shikhara-uboot>preview_on_hdmi
Going to see the preview on HDMI
```

5.4.2.2 Displaying captured image on HDMI

This test case is validated by capturing the image from the CSI and then store it to the SD card through test commands and displaying it on HDMI.

Peripheral Instance	Command Used	Test Status	Remarks
CSI	caminit, framecapturetoSDCard	Passed	Tested using U-Boot

Execution of test case using Commands:

- **hdmi_enable**
- **i2c dev 2**



- **i2c probe**
- **caminit**
- **preview_on_hdmi**
- **capture_frame <Addr>**
- **bmp display <Addr>**

After initializing the hdmi and csi, through the command **caminit** it will initialize the cam and execute **preview_on_hdmi** will start the streaming and displays preview on hdmi. Then by using **capture_frame** will captures the focused image from prompt which is saved to the SD card and to the specified address location. And this image will be displayed on HDMI by executing the **bmp display** command and specifying image address location as a second argument.

Console Output:

```
shikhara-uboot>hdmi_enable
HDMI Panel Detected
Shikhara Display controller initilized successfully
HDMI configured Successfully
shikhara-uboot>i2c probe
Valid chip addresses: 3C
shikhara-uboot>i2c dev 2
Setting bus to 2
shikhara-uboot>caminit
Going to call MIPI CSI init
mipi_csi_bridge_init() completed.
MIPI-CSI2 Bridge Initilized Successfully
Cam init is successfully completed
shikhara-uboot>preview_on_hdmi
Going to see the preview on HDMI
Specified frames are received
shikhara-uboot>capture_frame 0x12f00000
Adding BMP Image header
Pixel format is 24 bpp mode.
```

```
writing camera_snap.bmp

** Unable to write ..

Camera snap is successfully stored in SD(With Res is 640x480)

shikhara-uboot>bmp display 0x12f00000
```

5.4.2.3 CSI register Dump

To display the register contents of CSI controller and CSI Bridge controller .By using the commands will display the register dump values.

Peripheral Instance	Command Used	Test Status	Remarks
CSI	csi_dump, csi_bridge_dump	Passed	Tested using U-Boot

Execution of test case using Commands:

- **csi_dump**
- **csi_bridge_dump**

These are the commands used to display the register values of csi and csi bridge controller on the console.

Console Output:

```
shikhara-uboot>csi_dump
=====CSI Register dump=====
CSI2_VERSION: Address at d4563000 and value 3130342A
CSI2_N_LANES: Address at d4563004 and value 1
CSI2_PHY_SHUTDOWNZ: Address at d4563008 and value 1
CSI2_DPHY_RSTZ: Address at d456300c and value 1
CSI2_RESETN: Address at d4563010 and value 1
CSI2_PHY_STATE: Address at d4563014 and value 330
CSI2_DATA_IDS_1: Address at d4563018 and value 0
CSI2_DATA_IDS_2: Address at d456301c and value 0
CSI2_ERR1: Address at d4563020 and value 0
CSI2_ERR2: Address at d4563024 and value 0
CSI2_MSK1: Address at d4563028 and value 0
CSI2_MSK2: Address at d456302c and value 0
CSI2_PHY_TEST_CTRL0: Address at d4563030 and value 0
```

```
CSI2_PHY_TEST_CTRL1: Address at d4563034 and value 0
shikhara-uboot>csi_bridge_dump
=====CSI BRIDGE Register dump=====
CSI Dest FB: Address at d45a8000 and value 11000000
CSI frame status register at d45a8004 and value 0
AXI_Error_FIFO_Mask: Address at d45a8c00 and value 3FF
AXI_Error_FIFO_Status: Address at d45a8c08 and value 15
Current_AXI_Write_Addr: Address at d45a8c10 and value 111E0F80
Packet_Header_Info: Address at d45a8c18 and value 24078003
CSI_DSI_Direct_Config: Address at d45a8c20 and value 24
FRAME_FORMAT_CON_VC1: Address at d45a8800 and value 24
FRAME_INT_VC1: Address at d45a8808 and value A
HORZ_RES_VC1: Address at d45a8818 and value 7800780
VERTICAL_RES_VC1: Address at d45a8810 and value 1E0
FRAME_LINE_INT_MASK_VC1: Address at d45a8820 and value 0
FRAME_LINE_INT_STAT_VC1: Address at d45a8828 and value F0
FRAME_COUNT_READ_VC1: Address at d45a8830 and value 0
```

5.4.3 DSI Validation

5.4.3.1 Test case_1: DSI colour pattern display

This test case is validated by displaying the colour pattern on the DSI. For this need to initialize the DSI through the u-boot commands.

Peripheral Instance	Command Used	Test Status	Remarks
DSI	dsi_enable, dsi_colour_pattern	Passed	Tested using U-Boot

Execution of test case using Commands:

- dsi_enable
- dsi_colour_pattern

These are test commands used to validate. **dsi_enable** will initialize the dsi and **dsi_colour_pattern** will displays the colour patterns on the dsi.

Console Output:

```
shikhara-uboot>dsi_enable
```

```
DPHY Module initilized:Success

DSI bridge is initialized, Enable Bridge using dsi_bridge_enable command on u-boot prompt

Shikhara Display controller initilized successfully

TRULY LCD Panel initialization is done properly

shikhara-uboot>dsi_color_pattern

Going to display Blue colour on MIPI DSI Panel.....

Going to display Green colour on MIPI DSI Panel.....

Going to display Red colour on MIPI DSI Panel.....

Going to display Green,Blue,Red colour bar pattern on MIPI DSI Panel....
```

5.4.3.2 Test Case_2: Display Image on DSI

This test case is validated by displaying the image on DSI. Before, the image required to display should be copied on to the SD card. And then by using **fatload** commands copy the image to the RAM location. Then by using the **bmp dsi_display** command image will be displayed on the dsi.

Peripheral Instance	Command Used	Test Status	Remarks
DSI	mmc_start, mmc dev fatload mmc, bmp dsi_display	Passed	Tested using U-Boot

Execution of test case using Commands:

- **mmc_start**
- **mmc dev 0**
- **fatls mmc 0**
- **fatload mmc 0 <Addr> <file_name>**
- **bmp dsi_display <Image (RAM) address>**

These are the commands used to display the image on the DSI. This **bmp dsi_display** command will display the image from the location specified as a third argument in the command will copy the image to buffer and will display on to the DSI.

Console Output:

```
shikhara-uboot>mmc_start
```

```
shikhara-uboot>mmc dev 0

shikhara-uboot>fatls mmc 0

shikhara-uboot>fatload mmc 0 0x01500000 rsz_red_street_ferrari_ferrari_enzo_76877_800x1280.bmp
reading rsz_red_street_ferrari_ferrari_enzo_76877_800x1280.bmp

3072138 bytes read

shikhara-uboot>bmp dsi_display 0x01500000

Display-Panel: w=800 x h=1280

Display-bmp: w=800 x h=1280 with 24 colors

Display-bmp: w=-1051972528 x h=1278 with 24 colors
```

5.4.4 LCD Validation

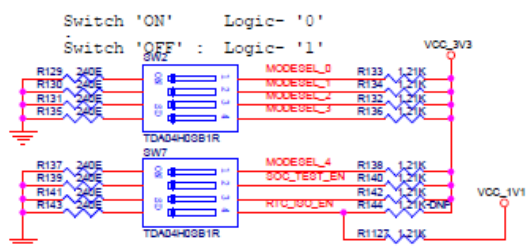
- ➔ LCD Test_1 – Display colours on LCD
- ➔ LCD Test_2 – Display Image on LCD

5.4.4.1 LCD Test_1 – Display colours on LCD

This test case is validated by displaying some colour patterns on LCD. By using the test commands will display the colours on LCD. To validate these test case need to change the mode to (sw2) **mode -1** before and power on the board and test using the following commands. Refer the following images.

SW2

Mode Selection	SOC_TEST_EN	MODE_SEL_4	MODE_SEL_3	MODE_SEL_2	MODE_SEL_1	MODE_SEL_0
NORMAL_MODE_0	0	0	x	x	0	0
NORMAL_MODE_1	0	0	x	x	0	1
NORMAL_MODE_2	0	0	x	x	1	0
NORMAL_MODE_3	0	0	x	x	1	1
TB_MODE	0	1	x	0	0	0
TRACE_MODE	0	1	x	0	0	1



Peripheral Instance	Command Used	Test Status	Remarks
LCD	lcd_enable, lcd_color_pattern	Passed	Tested using U-Boot

Execution of test case using Commands:



- **lcd_enable**
- **lcd_colour_pattern**
- **lcd_disable**

These are the test commands used, **lcd_init** initialize the lcd and and executing the **lcd_colour** will start displaying the colours on LCD. The colour patterns like green,blue,yellow,magenta,cyan,white and black will be displayed. Then after the combination of RGBY colours will be displayed on LCD at a time.

Console Output:

```
shikhara-uboot>lcd_enable
Going for PL111(CLCD) Panel configuration...
DEBUG_LCD: clcd base address is d4554000
DEBUG_LCD: lcd parameter tim0 is 2dd227c4
DEBUG_LCD: lcd parameter tim1 is 16164ddf
DEBUG_LCD: lcd parameter tim2 is 31f000a
DEBUG_LCD: lcd parameter tim3 is 0
DEBUG_LCD: lcd parameter cntl is 1002b
DEBUG_LCD: lcd parameter UPBASE is 1e00000
DEBUG_LCD: lcd parameter cntl is 1082b

shikhara-uboot>lcd_color_pattern
Going to Display/Show Green Color
Going to Display/Show Blue Color
Going to Display/Show Yellow Color
Going to Display/Show Magenta Color
Going to Display/Show Cyan Color
Going to Display/Show White Color
Going to Display/Show Block Color
Going to Display/Show Color: Red,Green,Blue, Yellow
lcd color bar test has done

shikhara-uboot>lcd_disable
Lcd Powered off
```

5.4.4.2 LCD Test_2 – Display Image on LCD

This test case is validated by displaying the image on LCD. Before need to copy the .bmp format image on to the RAM location. Then by test command will display the image on lcd.

Peripheral Instance	Command Used	Test Status	Remarks
---------------------	--------------	-------------	---------

LCD	lcd_enable , bmp lcd_display	Passed	Tested using U-Boot
-----	---------------------------------	--------	---------------------

Execution of test case using Commands:

- lcd_enable
- bmp lcd_display <Img_Addr> { x y }

This are the test commands used to display the image on LCD. **bmp lcd_disply** is the command in which the second argument is the address at which the image is copied and third argument is the x and y positions. This will pick the image from the specified address and will display on LCD.

Console Output:

```
shikhara-uboot>fatload mmc 0 0x0 atp1.bmp
Dwmci_init Host base address = 0xd4555000
Buswidth = 1, clock: 25000000
Buswidth = 1, clock: 400000
Buswidth = 4, clock: 400000
Buswidth = 4, clock: 25000000
reading atp1.bmp
921654 bytes read
shikhara-uboot>bmp lcd_display oxo 0 0
Display-Panel: w=800 x h=480
Display-bmp: w=640 x h=480 with 16777216 colors
Display-bmp: w=640 x h=480 with 16777216 colors
```

5.5 TEST CASES FOR H/W ACCELERATED PERIPHERALS

5.5.1 GPU Validation

The Mali Test suite is a group of tests that exercises the GPU. It takes predefined input data and process it and compares the result with predefined result data that is supplied to verify the correct functionality if the MALI GPU.

The path for the source codes for these GPU validation are available under the firmware release folders, the path is, -> **firmware\gpu_validation**

Peripheral Instance	Command Used	Test Status	Remarks
GPU	gputest	Passed	Tested using U-Boot

Execution of test case using Commands:

- gputest <core> <test_number>



- **gputest <version>**

These are test commands used to validate the GPU. In **gputest** command second argument specifies which core you want to test, either Geometry Processor (GP) or Pixel Processor (PP).

To validate this test case need to enable the “**CONFIG_SHIKHARA_GPU**” macro in the anusoc.h file.

Console Output:

```
shikhara-uboot>gputest version

-----GPU(Mali-400) Version-----

Geometric Processor(GP): 0xB070101
L2 Cache Controller   : 0xCAC20101
Pixel Processor-0(PP0) : 0xCD070101
Pixel Processor-1(PP1) : 0xCD070101
Pixel Processor-2(PP2) : 0xCD070101
Pixel Processor-3(PP3) : 0xCD070101

shikhara-uboot>
```

5.5.1.1 Geometry Processor (GP)

For testing this GP core with the **gputest** command , second argument will be “gp” and third argument will be test number which would be from 0 to 15.

Peripheral Instance	Command Used	Test Status	Remarks
GPU	gputest gp	Passed	Tested using U-Boot

Execution of test case using Commands:

- **gputest gp <test_number>**

Console Output:

```
shikhara-uboot>gputest gp 0

TEST PASSED

shikhara-uboot>gputest gp 1

interrupt request from do_irq
```


Done Processing Interrupt

Performing data check of 256 bytes

TEST PASSED

shikhara-uboot>gputest gp 2

interrupt request from do_irq

Performing data check of 256 bytes

TEST PASSED

shikhara-uboot>gputest gp 4

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 256 bytes

TEST PASSED

shikhara-uboot>gputest gp 5

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 128 bytes

TEST PASSED

shikhara-uboot>gputest gp 6

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 256 bytes

TEST PASSED

shikhara-uboot>gputest gp 7

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 304 bytes

TEST PASSED

shikhara-uboot>gputest gp 8

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 256 bytes

TEST PASSED

shikhara-uboot>gputest gp 9

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 128 bytes

TEST PASSED

shikhara-uboot>gputest gp 10

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 128 bytes

TEST PASSED

shikhara-uboot>gputest gp 11

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 1536 bytes

TEST PASSED

shikhara-uboot>gputest gp 12

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 64 bytes

TEST PASSED

shikhara-uboot>gputest gp 13

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest gp 14

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 40 bytes

TEST PASSED

shikhara-uboot>gputest gp 15

```

interrupt request from do_irq
Performing data check of 16 bytes
TEST PASSED
shikhara-uboot>gputest gp 16
Please GP test number below 16
shikhara-uboot>

```

5.5.1.2 Pixel Processor (PP)

For testing this PP core with the “**gputest**” command, second argument will be “**pp0**”, “**pp1**”, “**pp2**” or “**pp3**” and third argument will be test number which would be from 0 to 34 for each pp test cases.

Peripheral Instance	Command Used	Test Status	Remarks
GPU	gputest pp0, gputest pp1, gputest pp2, gputest pp3	Passed	Tested using U-Boot

Execution of test case using Commands:

- **gputest <pp0/1/2/3> <test_number>**

Console Output:

```

shikhara-uboot>gputest pp0 0
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp0 1
interrupt request from do_irq
Done Processing Interrupt

```

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp0 2

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 2048 bytes

TEST PASSED

shikhara-uboot>gputest pp0 3

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp0 4

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp0 5

interrupt request from do_irq

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp0 6

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp0 7

interrupt request from do_irq

Done Processing Interrupt0

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp0 8

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp0 9

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp0 10

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp0 11

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp0 12

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 4096 bytes

TEST PASSED

shikhara-uboot>gputest pp0 13

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 49152 bytes

TEST PASSED

shikhara-uboot>gputest pp0 14

```
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 16384 bytes
TEST PASSED
shikhara-uboot>gputest pp0 15
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 8192 bytes
TEST PASSED
shikhara-uboot>gputest pp0 16
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 9216 bytes
TEST PASSED
shikhara-uboot>gputest pp0 17
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 512 bytes
TEST PASSED
shikhara-uboot>gputest pp0 18
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 8192 bytes
TEST PASSED
shikhara-uboot>gputest pp0 19
interrupt request from do_irq
Done Processing Interrupt
TEST PASSED
shikhara-uboot>gputest pp0 20
interrupt request from do_irq
Done Processing Interrupt
```

Reading performance counters

TEST PASSED

shikhara-uboot>gputest pp0 21

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

TEST PASSED

shikhara-uboot>gputest pp0 22

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp0 23

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp0 24

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp0 25

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp0 26

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp0 27

TEST PASSED

shikhara-uboot>gputest pp0 28

TEST PASSED

```
shikhara-uboot>gputest pp0 29
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp0 30
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 8192 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp0 31
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 4096 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp0 32
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 20608 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp0 33
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 32768 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>
```

```
shikhara-uboot>gputest pp1 0
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 32768 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp1 1
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```


Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp1 2

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 2048 bytes

TEST PASSED

shikhara-uboot>gputest pp1 3

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp1 4

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp1 5

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp1 6

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp1 7

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp1 8

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp1 9

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp1 10

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp1 11

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp1 12

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 4096 bytes

TEST PASSED

shikhara-uboot>gputest pp1 13

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 49152 bytes

TEST PASSED

shikhara-uboot>gputest pp1 14

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp1 15

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp1 16

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 9216 bytes

TEST PASSED

shikhara-uboot>gputest pp1 17

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 512 bytes

TEST PASSED

shikhara-uboot>gputest pp1 18

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp1 19

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 20

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 21

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 22

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 23

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 24

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 25

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 26

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 27

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp1 28

TEST PASSED

shikhara-uboot>gputest pp1 29

TEST PASSED

shikhara-uboot>gputest pp1 30

TEST PASSED

shikhara-uboot>gputest pp1 31

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp1 32

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 4096 bytes

TEST PASSED

shikhara-uboot>gputest pp1 33

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 20608 bytes

TEST PASSED

shikhara-uboot>gputest pp1 34

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>

shikhara-uboot>gputest pp2 0

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp2 1

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp2 2

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 2048 bytes

TEST PASSED

shikhara-uboot>gputest pp2 3

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp2 4

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp2 5

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp2 6

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp2 7

```
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 8192 bytes
TEST PASSED
shikhara-uboot>gputest pp2 8
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 8192 bytes
TEST PASSED
shikhara-uboot>gputest pp2 9
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 16384 bytes
TEST PASSED
shikhara-uboot>gputest pp2 10
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp2 11
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 16384 bytes
TEST PASSED
shikhara-uboot>gputest pp2 12
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 4096 bytes
TEST PASSED
shikhara-uboot>gputest pp2 13
interrupt request from do_irq
```

Done Processing Interrupt

Performing data check of 49152 bytes

TEST PASSED

shikhara-uboot>gputest pp2 14

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp2 15

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp2 16

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 9216 bytes

TEST PASSED

shikhara-uboot>gputest pp2 17

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 512 bytes

TEST PASSED

shikhara-uboot>gputest pp2 18

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp2 19

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 20

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 21

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 22

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 23

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 24

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 25

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 26

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp2 27

TEST PASSED

```
shikhara-uboot>gputest pp2 28
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp2 29
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp2 30
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 8192 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp2 31
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 4096 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp2 32
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 20608 bytes
```

```
TEST PASSE
```

```
shikhara-uboot>gputest pp2 33
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 32768 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp2 34
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
```

```
Performing data check of 32768 bytes
```

```
TEST PASSED
```

```
shikhara-uboot>gputest pp2 35
```

```
interrupt request from do_irq
```

```
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>
shikhara-uboot>gputest pp3 0
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp3 1
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp3 2
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 2048 bytes
TEST PASSED
shikhara-uboot>gputest pp3 3
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp3 4
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp3 5
interrupt request from do_irq
```

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp3 6

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp3 7

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp3 8

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 9

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp3 10

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest pp3 11

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp3 12

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 4096 bytes

TEST PASSED

shikhara-uboot>gputest pp3 13

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 49152 bytes

TEST PASSED

shikhara-uboot>gputest pp3 14

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest pp3 15

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp3 16

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 9216 bytes

TEST PASSED

shikhara-uboot>gputest pp3 17

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 512 bytes

TEST PASSED

shikhara-uboot>gputest pp3 18

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp3 19

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 20

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 21

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 22

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 23

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 24

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 25

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 26

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 27

interrupt request from do_irq

Done Processing Interrupt

TEST PASSED

shikhara-uboot>gputest pp3 28

TEST PASSED

shikhara-uboot>gputest pp3 29

TEST PASSED

shikhara-uboot>gputest pp3 30

TEST PASSED

shikhara-uboot>gputest pp3 31

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest pp3 32

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 4096 bytes

TEST PASSED

shikhara-uboot>gputest pp3 33

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 20608 bytes

TEST PASSED

```
shikhara-uboot>gputest pp3 34
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp3 35
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp3 36
interrupt request from do_irq
Done Processing Interrupt
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest pp3 37
Please PP3 test number below 37
shikhara-uboot>
```

5.5.1.3 Test Cases for MMU-1 (MP-1):

For testing this MMU-1 core with the “**gputest**” command, second argument will be “**mp1**” and third argument will be test number which would be from 0 to 5.

Peripheral Instance	Command Used	Test Status	Remarks
GPU	gputest mp1	Passed	Tested using U-Boot

Execution of test case using Commands:

- **gputest mp1 <test_number>**



Console Output:

shikhara-uboot>gputest mp1 0

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest mp1 1

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest mp1 2

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

Performing data check of 2048 bytes

TEST PASSED

shikhara-uboot>gputest mp1 3

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest mp1 4

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest mp1 5

Please MP1 test number below 5

5.5.1.4 Test Cases for MMU-2 (MP-2):

For testing this MMU-2 core with the “**gputest**” command, second argument will be “**mp2**” and third argument will be test number which would be from 0 to 8.

Peripheral Instance	Command Used	Test Status	Remarks
GPU	gputest mp2	Passed	Tested using U-Boot

Execution of test case using Commands:

- **gputest mp2 <test_number>**

Console Output:

```
shikhara-uboot>gputest mp2 0
interrupt request from do_irq
Done Processing Interrupt
Reading performance counters
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest mp2 1
interrupt request from do_irq
Done Processing Interrupt
Reading performance counters
Performing data check of 32768 bytes
TEST PASSED
shikhara-uboot>gputest mp2 2
interrupt request from do_irq
Done Processing Interrupt
Reading performance counters
Performing data check of 8192 bytes
```

TEST PASSED

shikhara-uboot>gputest mp2 3

interrupt request from do_irq

Done Processing Interrupt

Performing data check of 8192 bytes

TEST PASSED

shikhara-uboot>gputest mp2 4

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest mp2 5

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

Performing data check of 32768 bytes

TEST PASSED

shikhara-uboot>gputest mp2 6

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

Performing data check of 16384 bytes

TEST PASSED

shikhara-uboot>gputest mp2 7

interrupt request from do_irq

Done Processing Interrupt

Reading performance counters

Performing data check of 4096 bytes

TEST PASSED

shikhara-uboot>gputest mp2 8

Please MP2 test number below 8



5.5.1.5 GPU Full data path test

This test verifies that the complete data path for Mali GPU using the test command. To validate this need to enable these macro's "CONFIG_GPU_SCENE" and "HDMI_720_480" in anusoc.h file.

Peripheral Instance	Command Used	Test Status	Remarks
GPU	hdmi_enable gpu_scene	Passed	Tested using U-Boot

Execution of test case using Commands:

- **hdmi_enable**
- **gpu_scene**

This is the test command used to validate the GPU.

Console Output:

```
shikhara-uboot>hdmi_enable
shikhara-uboot>gpu_scene
copying Pagetable...
done copying Pagetable!
copying data...
done copying data!
starting test
configuring performace counters to GP,PP0,PP1..
configuring GP...
configuring PP0..
configuring PP1..
interrupt request from do_irq
Configured MALI..
```

5.5.2 AV417 Video Subsystem Validation

- **AV417V Test_1:** To test MPEG4 Encoder functionality.
- **AV417V Test_2:** To test MPEG4 Decoder functionality.
- **AV417V Test_3:** To test H264 Encoder functionality.
- **AV417V Test_4:** To test H264 Decoder functionality.
- **AV417V Test_5:** To test alpha blending.
- **AV417V Test_6:** To test IPC.

5.5.2.1 AV417V Test_1: To test MPEG4 Encoder functionality.

To validate this test case, first need to copy all the necessary files from the SD card to the ram memory location. Execution of this test commands and procedure is as follows.

- For All AV417 validation, First Use **mmc_start** command.
- Enable the AV417 by executing this command **av417enable** for every test case.
- Load jump application binary onto sram address using below command.
 - `fatload mmc 0 0xD41F0000 app_jump.bin`
- Load the encoder application onto the ddr using below command.
 - `fatload mmc 0 0x01D00000 app_mpeg4_encoder.bin`
- Load input.yuv file onto ddr.
 - `fatload mmc 0 0x01500000 input.yuv`
- Reset the arc processor using "**av417reset**" arm uboot command.
- After reset, encoder application starts executing.
- After execution of application, load the output video into a file using fatwrite command.
 - `fatwrite mmc 0 0x01B00000 encoded.m4v 0x1dc00`

Peripheral Instance	Command Used	Test Status	Remarks
AV417V	av417enable av417reset	Passed	Tested using U-Boot

Execution of test case using Commands:

- **av417enable**

- `mmc_start`
- `fatls mmc 0`
- `fatload mmc 0 <Address> <file>`
- `av417reset`
- `fatwrite mmc 0 <Addr> <File> <Len>`

Console Output:

```
shikhara-uboot>av417enable
Enabled Power to the AV417 Video subsystem
shikhara-uboot>fatload mmc 0 0xD41F0000 app_jump.bin

shikhara-uboot>fatload mmc 0 0x01D00000 app_mpeg4_encoder.bin

shikhara-uboot>fatload mmc 0 0x01500000 input.yuv

shikhara-uboot>av417reset
Releasing Reset
shikhara-uboot>fatwrite mmc 0 0x01B00000 encoded.m4v 0x1dc00
writing encoded.m4v
121856 bytes written
```

5.5.2.2 AV417V Test_2: To test MPEG4 Decoder functionality

To validate this test case, first need to copy all the necessary files from the SD card to the ram memory location. Execution of this test commands and procedure is as follows.

- For All AV417 validation, First Use **mmc_start** command.
- Load jump application binary onto sram address using below command.
 - `fatload mmc 0 0xD41F0000 app_jump.bin`
- Load the encoder application onto the ddr using below command.
 - `fatload mmc 0 0x01D00000 app_mpegdec.bin`
- Load encoded.m4v file onto ddr.
 - `fatload mmc 0 0x01B00000 encoded.m4v`
- Reset the arc processor using "av417reset" arm u-boot command.
- After reset, decoder application starts executing.
- After execution of application, load the output video into a file using fatwrite command.



- `fatwrite mmc 0 0x01500000 decoded.yuv 0x567000`

Peripheral Instance	Command Used	Test Status	Remarks
AV417V	av417reset	Passed	Tested using U-Boot

Execution of test case using Commands:

- `mmc_start`
- `fatls mmc 0`
- `fatload mmc 0 <Address> <file>`
- `av417reset`
- `fatwrite mmc 0 <Addr> <File> <Len>`

Console Output:

```
shikhara-uboot>fatload mmc 0 0xD41F0000 app_jump.bin
shikhara-uboot>fatload mmc 0 0x01B00000 encoded.m4v
shikhara-uboot>fatload mmc 0 0x01D00000 app_mpegdec.bin
reading encoded.m4v
121856 bytes read
shikhara-uboot>av417reset
system already got reset: 5
Releasing Reset
Jumping to App
Different NAL Type
Decoding :
Started decoding a frame
completed decoding
Finished ... bye!

shikhara-uboot>fatwrite mmc 0 0x01500000 decoded.yuv 0x567000
writing decoded.yuv
```

5.5.2.3 AV417V Test_3: To test H264 Encoder functionality

To validate this test case, first need to copy all the necessary files from the SD card to the ram memory location. Execution of this test commands and procedure is as follows.

- For All AV417 validation, First Use **mmc_start** command.
- Load jump application binary onto sram address using below command.
 - **fatload mmc 0 0xD41F0000 app_jump.bin**
- Load the encoder application onto the ddr using below command.
 - **fatload mmc 0 0x01D00000 h264_encoder_app.bin**
- Load input.yuv file onto ddr.
 - **fatload mmc 0 0x01500000 input.yuv**
- Reset the arc processor using "**av417reset**" arm uboot command.
- After reset, encoder application starts executing.
- After execution of application, load the output video into a file using fatwrite command.
 - **fatwrite mmc 0 0x01B00000 encoded.h264 0x13000**

Peripheral Instance	Command Used	Test Status	Remarks
AV417V	av417reset	Passed	Tested using U-Boot

Execution of test case using Commands:

- **mmc_start**
- **fatls mmc 0**
- **fatload mmc 0 <Address> <file>**
- **av417reset**
- **fatwrite mmc 0 <Addr> <File> <Len>**

Console Output:

```
shikhara-uboot>fatload mmc 0 0xD41F0000 app_jump.bin
shikhara-uboot>fatload mmc 0 0x01D00000 h264_encoder_app.bin
shikhara-uboot>fatload mmc 0 0x01500000 input.yuv
shikhara-uboot>av417reset
system already got reset: 5
Releasing Reset
Jumping to App
encoding
Set up input frame memory
```



```

started encoding
Completed encoding
Finished ... bye!

shikhara-uboot>fatwrite mmc 0 0x01B00000 encoded.h264 0x13000
writing encoded.h264
77824 bytes written
shikhara-uboot>fatls mmc 0
      images/
    3304  app_jump.bin
  118960  h264dec.bin
.....
 5664768  decoded.yuv
   77824  encoded.h264

9 file(s), 4 dir(s)

```

5.5.2.4 AV417V Test_4: To test H264 Decoding functionality

To validate this test case, first need to copy all the necessary files from the SD card to the ram memory location. Execution of this test commands and procedure is as follows.

- Load jump application binary onto sram address using below command.
 - **fatload mmc 0 0xD41F0000 app_jump.bin**
- Load the decoder application onto the ddr using below command.
 - **fatload mmc 0 0x01D00000 h264dec.bin**
- Load encoded.m4v file onto ddr.
 - **fatload mmc 0:1 0x01B00000 encoded.h264**
- Reset the arc processor using "**av417reset**" arm uboot command.
After reset, decoder application starts executing.
- After execution of application, load the output video into a file using fatwrite command.
 - **fatwrite mmc 0:1 0x01500000 decoded_h264.yuv 0x567000**
- Play the yuv file using yuvplayer, if the resolution is other than 176*144.

Peripheral Instance	Command Used	Test Status	Remarks
AV417V	av417reset	Passed	Tested using U-Boot

Execution of test case using Commands:

- **mmc_start**
- **fatls mmc 0**
- **fatload mmc 0 <Address> <file>**
- **av417reset**



- **fatwrite mmc 0 <Addr> <File> <Len>**

Console Output:

```
shikhara-uboot>fatload mmc 0 0xD41F0000 app_jump.bin

shikhara-uboot>fatload mmc 0 0x01D00000 h264dec.bin

shikhara-uboot>fatload mmc 0 0x01B00000 encoded.h264

shikhara-uboot>av417reset
system already got reset: 5
Releasing Reset
Jumping to App
uboot> Started H264 Decoding
Started Decoding
.....
End Of Data
.Finished Bye

shikhara-uboot>fatwrite mmc 0 0x01500000 decoded_h264.yuv 0x567000
writing decoded_h264.yuv
5664768 bytes written
shikhara-uboot>fatls mmc 0
      images/
    3304   app_jump.bin
  118960   h264dec.bin
.....
    77824   encoded.h264
  5664768   decoded_h264.yuv
```

5.5.2.5 AV417V Test_5: To test alpha blending.

This test case validation procedure and commands to test are as follows, need to enable the “HDMI_720_480” macro in the anusoc.h file to validate this test case.

- Run **hdmi_enable** command.
- Load the application binary on to on-chip sram address using below command.
 - **fatload mmc 0 0xD41F0000 app_alphablending.bin 3668**
- Load foreground and background images using the following command.
 - **fatload mmc 0 0x01500000 background.raw 1036800**
 - **fatload mmc 0:1 0x015FF200 foreground.raw 1036800**
- After loading the required images, reset the arc processor using "**av417reset**" arm u-boot command.
- After reset, Alpha blending Application runs from on-chip sram start address and blended image will be displayed on HDMI Display.

Peripheral Instance	Command Used	Test Status	Remarks
AV417V	av417reset	Passed	Tested using U-Boot

Execution of test case using Commands:

- **hdmi_enable**
- **mmc_start**
- **fatls mmc 0**
- **fatload mmc 0 <Address> <file>**
- **av417reset**
- **fatwrite mmc 0 <Addr> <File> <Len>**

Console Output:

```
shikhara-uboot> fatload mmc 0 0xD41F0000 app_alphablending.bin 3668
reading app_alphablending.bin

3668 bytes read
shikhara-uboot> fatload mmc 0 0x01500000 background.raw 1036800
reading background.raw

1036800 bytes read
shikhara-uboot> fatload mmc 0 0x015FF200 foreground.raw 1036800
reading foreground.raw

1036800 bytes read
shikhara-uboot> hdmi_enable
shikhara-uboot> av417reset
Releasing Reset
shikhara-uboot> completed
```

5.5.2.6 AV417V Test_6: To test IPC

This test case validation procedure and commands to test are as follows,

- Load the application binary on to on-chip sram address using below command.
 - **fatload mmc 0 0xd41F0000 app_ipc.bin 3452**
- After loading the binary onto sram, run "**arcipc**" command on arm u-boot code.



- "**arcipc**" command, which writes data to ddr memory and generates interrupt to arc, the respective arc interrupt handler is called. Arc processor writes data into ddr memory and in turn generates interrupt to arm, In arm interrupt handler, the respective memory location is compared, if the data is not same, IPC test is successfully completed.

Peripheral Instance	Command Used	Test Status	Remarks
AV417V	arcipc	Passed	Tested using U-Boot

Execution of test case using Commands:

- *fatls mmc 0*
- *fatload mmc 0 <Address> <file>*
- *arcipc*

Console Output:

```
shikhara-uboot> fatload mmc 0 0xd41F0000 app_ipc.bin 3452
reading app_ipc.bin
3452 bytes read
shikhara-uboot> arcipc
Releasing Reset
interrupt request from do_irq
in arm irq handler
ipc test success
```

5.6 TEST CASES FOR MISCELLANEOUS PERIPHERALS

5.6.1 GPIO Validation

5.6.1.1 Test case for Glow LEDs

Test case is validated by Toggle the LEDS on PORT-0 & PORT-1 (PORT-A and PORT-B).Toggle LEDS on PORT-0 & PORT-1 (PORT-A and PORT-B).

Peripheral Instance	Command Used	Test Status	Remarks
GPIO	gpio_led	Passed	Tested using U-Boot

Execution of test case using Commands:

- *gpio_led*

Console Output:

```
shikhara-uboot>gpio_led
```

```
Test is going to Toggle LEDS on PORT-0 & PORT-1 (PORT-A and PORT-B)
```

```
Toggle LEDS on PORT-0 & PORT-1 (PORT-A and PORT-B)..Completed
```

5.6.1.2 Test case for Driving low of GPIO's

Test is going to drive low of GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D). Drive low the GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D).

Peripheral Instance	Command Used	Test Status	Remarks
GPIO	gpio_output_low	Passed	Tested using U-Boot

Execution of test case using Commands:

- *gpio_output_low*

Console Output:

```
shikhara-uboot>gpio_output_low
```

Test is going to drive low of GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D)

Drive low the GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D)..Completed

5.6.1.3 Test case for Driving high of GPIO's

Test is going to drive high of GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D). Drive high the GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D).

Peripheral Instance	Command Used	Test Status	Remarks
GPIO	gpio_output_high	Passed	Tested using U-Boot

Execution of test case using Commands:

- **gpio_output_high**

Console Output:

shikhara-uboot>gpio_output_high

Test is going to drive high of GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D)

Drive high the GPIOs on PORT-2 & PORT-3 (PORT-C and PORT-D)..Completed

5.6.1.4 Test case for reading GPIO's

Test is going to read GPIOs (input) on PORT-2 & PORT-3 (PORT-C and PORT-D).

Peripheral Instance	Command Used	Test Status	Remarks
GPIO	gpio_test_input	Passed	Tested using U-Boot

Execution of test case using Commands:

- **gpio_test_input**

Console Output:

shikhara-uboot>gpio_test_input

Test is going to read GPIOs(input) on PORT-2 & PORT-3 (PORT-C and PORT-D)

PORT-2(PORT-C) GPIO Input value=0x1

PORT-3(PORT-D) GPIO Input value=0x0

GPIO Input test on PORT-2 & PORT-3 (PORT-C and PORT-D)..Completed

5.6.2 Timer Validation

- **Timer Test_1** – DT-0 counter 0 32 BIT interrupt test.
- **Timer Test_2** – DT-0 counter 1 32 BIT interrupt test.
- **Timer Test_3** – DT-1 counter 0 32 BIT interrupt test.
- **Timer Test_4** – DT-1 counter 1 32 BIT interrupt test.
- **Timer Test_5** – Watchdog timer interrupt test.

5.6.2.1 Timer Test_1 – DT-0 counter 0 32 BIT interrupt test

NOTE: The above test can't be done, as timer is being initialized at system start-up and used throughout. Other timers will be tested.

5.6.2.2 Timer Test_2 – DT-0 of counter 1 32-bit interrupt test

This test case validates basic functionality of 32-bit timer. 32-bit timer is configured to generate interrupt. All other timer interrupts are disabled.

Peripheral Instance	Command Used	Test Status	Remarks
TIMER	sysconfig	Passed	Tested using U-Boot

Execution of test case using Commands:

- **sysconfig 0 1**

This is the test command used to validate the timer with second argument as 0 and third argument as 1 which are timer and counter instances (**sysconfig 0 1**). And will print the test result on the console.

Console Output:

```
shikhara-uboot> sysconfig 0 1
DT0 Timer value :0x2faf060
DT0 Timer value :0x2fad6ca
DT0 timer's interrupt is fired of counter1
32 bit Timer interrupt occurred
```

interrupt test is pass

5.6.2.3 Timer Test_3 – DT-1 of counter 0 32 BIT interrupt test

This test case validates basic functionality of 32-bit timer. 32-bit timer is configured to generate interrupt. All other timer interrupts are disabled.

Peripheral Instance	Command Used	Test Status	Remarks
TIMER	sysconfig	Passed	Tested using U-Boot

Execution of test case using Commands:

- **sysconfig 1 0**

This is the test command used to validate the timer with second argument as 1 and third argument as 0 which are timer and counter instances (**sysconfig 1 0**). And will print the test result on the console.

Console Output:

```
shikhara-uboot> sysconfig 1 0
DT1 Timer value :0x2faf060
DT1 Timer value :0x2fad668
DT1 timer's interrupt is fired of counter 0
32 bit Timer interrupt occurred
interrupt test is pass
```

5.6.2.4 Timer Test_4 – DT-1 of counter 1 32 BIT interrupt test

This test case validates basic functionality of 32-bit timer. 32-bit timer is configured to generate interrupt. All other timer interrupts are disabled.

Peripheral Instance	Command Used	Test Status	Remarks
TIMER	sysconfig	Passed	Tested using U-Boot

Execution of test case using Commands:

- **sysconfig 1 1**



This is the test command used to validate the timer with second argument as 1 and third argument as 1 which are timer and counter instances (**sysconfig 1 1**). And will print the test result on the console.

Console Output:

```
shikhara-uboot> sysconfig 1 1
DT1 Timer value :0x2faf060
DT1 Timer value :0x2fad6dd
DT1 timer's interrupt is fired of counter 1
32 bit Timer interrupt occurred
interrupt test is pass
```

5.6.2.5 Watchdog Timer interrupt test

This test case validates basic functionality of WDT timer. WDT timer is configured to generate the interrupt by loading appropriate load value. On count reaching 0, interrupt is generated.

Peripheral Instance	Command Used	Test Status	Remarks
TIMER	wdttest	Passed	Tested using U-Boot

Execution of test case using Commands:

- **wdttest**

Console Output:

```
shikhara-uboot> wdttest
interrupt request from do_irq
watchdog fired
watchdog_timer interrupt occurred
WDT interrupt test pass
```

5.6.3 GNSS Validation

To validate GNSS, follow the below steps:

Install Navika UI in any Windows PC to see the satellite tracking information. Navika UI executable is under firmware release folder path, ->**NavikaUI**

- a. Setting-up the board:



- ➔ Connect RF interface board to the shikhara board.
- ➔ Connect the antenna and it should be visible to the sky.
- ➔ Connect **UART-0** to the serial console of the PC to get u-boot prompt.
- ➔ Connect **UART-1** to any other windows PC where “**Navika UI**” is installed.
- ➔ Open the “**Navika UI**” executable to view the satellite tracking information

Peripheral Instance	Command Used	Test Status	Remarks
GNSS	gnss_test	Passed	Tested using U-Boot

Execution of test case using Commands:

- ***gnss_test***

Console Output:

```
shikhara-uboot> gnss_test

start gps

gps library start = 1408683264, end = 1408683264
Library initialized

Location available
Lat=3041501, Lon=13686463, Alt=499915

Location available
Lat=3041501, Lon=13686463, Alt=499915
Location available
Lat=3041501, Lon=13686463, Alt=499915
Location available
Lat=3041501, Lon=13686463, Alt=499915
```

NAVIKA UI Console Output:

```
$GNRMC,,V,,,,,,,,,N*4D
$GNGSA,A,1,,,,,,,,,00
$GNGSA,A,1,,,,,,,,,00
??`ââ,HDê<<<<<<<î<<< vv|
vi      4d      p      v|p      ë~,      î'

$GPGSV,3,1,00,,,,,,,,,7B
$GPGSV,3,2,00,,,,,,,,,78
$GPGSV,3,3,00,,,,,,,,,79
$GLGSV,3,1,04,67,00,000,20,69,00,000,18,71,00,000,20,76,00,000,15*67
$GLGSV,3,2,04,,,,,,,,,60
$GLGSV,3,3,04,,,,,,,,,61
```

```

$GPZDA,073850.57,21,02,2014,-00,00*46
$GNGGA,,,,,0,,,,M,,M,,*78
$GNRMC,,V,,,,,,,,,N*4D
$GNGSA,A,1,,,,,,,,,,,,,*00
$GNGSA,A,1,,,,,,,,,,,,,*00
??̂PçCpĀ<<<<<<<<<<<<
                                Ñ~  é

qé                                q

é
 ½          ½Đd          é
                ë~ê|øî¾

                                $GPGSV,3,1,00,,,,,,,,,,,,,*7B
$GPGSV,3,2,00,,,,,,,,,,,,,*78
$GPGSV,3,3,00,,,,,,,,,,,,,*79
$GLGSV,3,1,04,67,00,000,21,69,00,000,20,71,00,000,21,76,00,000,16*6F
$GLGSV,3,2,04,,,,,,,,,,,,,*60
$GLGSV,3,3,04,,,,,,,,,,,,,*61
$GPZDA,073851.57,21,02,2014,-00,00*47
$GNGGA,,,,,0,,,,M,,M,,*78
$GNRMC,,V,,,,,,,,,N*4D
$GNGSA,A,1,,,,,,,,,,,,,*00
$GNGSA,A,1,,,,,,,,,,,,,*00
??̂Döfpî<<<<<<<<<<<<  ıÓ²          páwáwé
                                z
                                é
                                ÄİĐd          p|vêöî%

$GPGSV,3,1,00,,,,,,,,,,,,,*7B
$GPGSV,3,2,00,,,,,,,,,,,,,*78
$GPGSV,3,3,00,,,,,,,,,,,,,*79
$GLGSV,3,1,06,66,00,000,,67,00,000,23,69,00,000,20,71,00,000,20*68
$GLGSV,3,2,06,73,00,000,,76,00,000,17,,,,,,,,,*61
$GLGSV,3,3,06,,,,,,,,,,,,,*63
$GPZDA,073852.57,21,02,2014,-00,00*44
$GNGGA,,,,,0,,,,M,,M,,*78

$GPGSV,3,2,03,,,,,,,,,,,,,*7B
$GPGSV,3,3,03,,,,,,,,,,,,,*7A
$GLGSV,3,1,05,67,00,000,19,69,00,000,16,71,00,000,20,75,00,000,11*65
$GLGSV,3,2,05,77,00,000,,,,,,,,,,,,,*51
$GLGSV,3,3,05,,,,,,,,,,,,,*60
$GPZDA,074108.63,21,02,2014,-00,00*42
$GNGGA,,,,,0,,,,M,,M,,*78
$GNRMC,,V,,,,,,,,,N*4D
$GNGSA,A,1,,,,,,,,,,,,,*00
$GNGSA,A,1,,,,,,,,,,,,,*00
¾_Ç
  yc          z<  c
                vj<<<<z<<<<îw¬ó          ñZñòTòT
                                ñĐ''
                                ñø0|èy

$GPGSV,3,1,03,05,00,000,18,08,00,000,15,17,00,000,21,,,,,*4D
$GPGSV,3,2,03,,,,,,,,,,,,,*7B
$GPGSV,3,3,03,,,,,,,,,,,,,*7A
$GLGSV,3,1,04,67,00,000,20,69,00,000,17,71,00,000,21,75,00,000,12*6D

```

```

$GLGSV,3,2,04,,,,,,,,,,,,,*60
$GLGSV,3,3,04,,,,,,,,,,,,,*61
$GPZDA,074109.63,21,02,2014,-00,00*43
$GNGGA,,,,,0,,,,M,,M,,*78
$GNRMC,,V,,,,,,,,,N*4D
$GNGSA,A,1,,,,,,,,,,,,,*00
$GNGSA,A,1,,,,,,,,,,,,,*00
{µdÊ
    je    kä¼f
           ô~·W
           Vl
           X<<G}<<<<iÛªi
                                   (ñZø0   Ä|i«    :èÐöpW

$GPGSV,3,1,02,05,00,000,18,17,00,000,20,,,,,,,,,*71
$GPGSV,3,2,02,,,,,,,,,,,,,*7A
$GPGSV,3,3,02,,,,,,,,,,,,,*7B
$GLGSV,3,1,04,67,00,000,21,69,00,000,18,71,00,000,21,75,00,000,12*63
$GLGSV,3,2,04,,,,,,,,,,,,,*60
$GLGSV,3,3,04,,,,,,,,,,,,,*61
$GPZDA,074110.63,21,02,2014,-00,00*4B
$GNGGA,,,,,0,,,,M,,M,,*78
$GNRMC,,V,,,,,,,,,N*4D
$GNGSA,A,1,,,,,,,,,,,,,*00
$GNGSA,A,1,,,,,,,,,,,,,*00
S;ki
    jg    }m´Ûi
           û<<Q<<<<iw|x    ð`¾
                                   £
                                   ôHñZi«    ñZð`ð`il°    ð`øvè

$GPGSV,3,1,03,05,00,000,15,30,00,000,,17,00,000,16,,,,,*4B
$GPGSV,3,2,03,,,,,,,,,,,,,*7B
$GPGSV,3,3,03,,,,,,,,,,,,,*7A
$GLGSV,3,1,04,67,00,000,20,69,00,000,17,71,00,000,20,75,00,000,14*6A
$GLGSV,3,2,04,,,,,,,,,,,,,*60
$GLGSV,3,3,04,,,,,,,,,,,,,*61
$GPZDA,074111.63,21,02,2014,-00,00*4A
$GNGGA,,,,,0,,,,M,,M,,*78
$GNRMC,,V,,,,,,,,,N*4D
$GNGSA,A,1,,,,,,,,,,,,,*00
$GNGSA,A,1,,,,,,,,,,,,,*00
??¯y±>Ð
    Yjyn¥mâl
W<<<<i£    ð`iw i¥;    ð`ä    ¿    |èn

$GPGSV,3,1,03,05,00,000,15,30,00,000,,17,00,000,14,,,,,*49
$GPGSV,3,2,03,,,,,,,,,,,,,*7B
$GPGSV,3,3,03,,,,,,,,,,,,,*7A
$GLGSV,3,1,04,67,00,000,21,69,00,000,15,71,00,000,21,75,00,000,11*6D
$GLGSV,3,2,04,,,,,,,,,,,,,*60
$GLGSV,3,3,04,,,,,,,,,,,,,*61

```

5.6.4 TZASC Validation

In Shikhara, TZASC are connected to DDR Interface to provide security configurations for external memory. To validate TZASC, we are configuring TZASC with 2- regions. Region- 0 is default configuration, Region-1 with secure access. And try to perform write/read operations to these regions in secure state and non-secure state of processor.

If Processor is in secure state, then this test will perform write/read operations to different regions and switch to non-secure state and run the same test in non-secure state. If Processor is in Non-Secure State then this test is failed, due to secure access violations

Define the flag "CONFIG_TZASC" in configuration file and generate u-boot binary. Setup FPGA board loaded with H/W build and firmware build

Peripheral Instance	Command Used	Test Status	Remarks
TZASC	tzasc	Passed	Tested using U-Boot

Execution of test case using Commands:

- tzasc

Console Output:

```
shikhara-uboot>tzasc
Processor is in Secure world...
Configure TZASC...
Performing writes/reads to secure memory region.. 0x0
Switch to Non Secure World.. processor will go to abort mode..
data abort
```

6. U-BOOT VALIDATION COMMANDS SUMMARY

IP name	Command	Parameters	Description
DT-0 (SP804)	<i>sysconfig</i>	0 0 0 1	DT-0 timer, conter-0 interrupt test DT-0 timer, conter-1 interrupt test
DT-1 (SP804)	<i>sysconfig</i>	1 0 1 1	DT-1 timer, conter-0 interrupt test DT-1 timer, conter-1 interrupt test
WDT (SP805)	<i>wdttest</i>	0 0	WDT timer interrupt generation test
RTC (PL031)	<i>date</i>	No parameters	Prints Epoch time
GIC (PL390)	<i>sysconfig</i> <i>wdttest</i>	Parameters as specified for DT and WDT	Tested Interrupt generation for DT and WDT using GIC.
UART (PL011)	<i>uart_loopbacktest</i>	No Parameters	Tests all UART's and checks to see transmitted & Received data is same, if so, Prints success.
CLCD (PL111)	<i>lcd_enable</i> <i>lcd_disable</i> <i>lcd_color_pattern</i> <i>bmp lcd_display <image address> x y</i> <i>lcd_dump</i>	--	Tests displaying colour pattern and image on CLCD
GPIO (PL061)	<i>gpio_led</i> <i>gpio_input_test</i>	-- --	This test blinks LED on Bank0 & Bank1. This will test the GPIO's in input (BANK2 and BANK3) & output mode.
KMI (PL050)	<i>apkybd</i> <i>apmouse</i>	--	Tests keyboard & Mouse connected to KMI. On success, prints the data from keyboard & Mouse.
DMAC (DMA330)	<i>dmatest</i>		Tests memory to memory DMA transfers.
SMC (PL353)	<i>sram_test</i> <i>nand_start</i> <i>nandtest</i> <i>nand info</i> <i>protect off all</i>	--	Compares write data with read data, and prints success, if same

	<i>flash_start</i> <i>nortest</i>		
TZASC (TZC380)	<i>tzasc</i>		Tests the security permissions for DDR memory.
SPI (PL022)	<i>sf probe</i> <i>sf read</i> <i>sf write</i> <i>sferase</i> <i>ts print</i> <i>ts calibrate</i>	0 0 0 0x100 0x0 0x10 0x100 0x0 0x10 0x0 0x40000 -- --	Tests SPI and serial flash connected over SPI0. Tests SPI touch screen controller and prints the touch positions on LCD panel.
I2C	<i>i2c probe</i> <i>i2c dev</i> <i>i2c md</i> <i>i2c mw</i> <i>i2c md</i>	-- - 0 - <i2c_chip> <dev_addr> <Len> - {i2c_chip} {devaddr} {data} {len}	Check the devices attached to the i2c using i2c probe. Write data from first buffer to the EEPROM device. Read data from EEPROM device, if same, test passes.
I2S	<i>mmc_start</i> <i>mmc part</i> <i>sound init</i> <i>sound sine</i> <i>fatload mmc</i> <i>sound wav</i>	-- -- -- -- 0 <Address> <.wav_file> <Address>	Initialises the MMC subsystem. Shows the number of partitions of MMC card. Initialises the I2S and SGTL5000 Codec controllers. Play a sine tone over SGTL5000 codec. Copy a wave file from MMC card on partition 2 to RAM address Play a wave file copied on a RAM address.
CAN	<i>can_start</i> <i>can init</i> <i>can xmit</i> <i>can rcv</i>	-- -- <id> <dlc> <d0> <d1><d7> --	Initialises the CAN subsystem. Initialises the CAN controller. Transmits a CAN frame consisting of an ID, DATA LENGTH and DATA(Bytes) Receives a CAN frame which is being transmitted by other CAN controller.
HDMI	<i>Hdmi_dvi_min</i> <i>hdmi_enable</i>	-- --	Fills HDMI panel with a color.

	<i>fatload mmc bmp display bmpslideshow coninfo set stdout hdmi</i>	0 <Addr> <file_name> <Addr> {x y}	Enables HDMI. Load the file. Display on HDMI console. Runs the slideshow on HDMI TV. Set the hdmi as a console.
MIPI CSI2, DSI	<i>dsi_enable dsi_dump lcd_enable dsi_colour_pattern bmp display caminit preview_on_hdmi capture_frame csi_dump csi_bridge_dump bmp dsi_display</i>	-- -- -- -- <Image Addr> -- -- -- <Addr> -- -- -- <Image Addr>	Display registers dump values of csi and dsi. Displays the DSI colour patterns on the LCD. Displays the CSI image on the hdmi.
USB OTG and Host	HOST VALIDATION: <i>usb start usb info env export fatwrite usb fatls usb fatload usb md usb part fatls usb fatload usb md</i> DEVICE VALIDATION: <i>fut sudo lsusb sudo ./smdk-usbdll md</i>	-- -- <Addr> bootdelay 0 <Addr> <File> <Size> 0 0 <Addr> <File> <Addr> -- 0 0 <Addr> <File> <Addr> -- <Addr> -v more -f bulk.bin -a <Addr> <Addr>	Starts USB controller. Display the device enumerated. Creates a file in the memory. Creates file & copies to USB. Format a USB pendrive with FAT filesystem and create file Write Some characters inside the file created. Load the file ‘xyz.txt’ onto RAM of board by using command fsload. Perform control transfer on Host-PC. Send a file from Host-PC to Shikara board by using dltool utility. Perform Bulk-Out Transfer by using command fut .
AV417	<i>fatload mmc fatload mmc mw mw</i>	0 <Addr> <.raw_file> 0 <Addr> <.raw_file> <Addr> <Data> <Size>	Write input raw files to memory using fatload command. This command loads input raw files from

	<i>fatload mmc</i> <i>md</i> <i>mw</i> <i>md</i> <i>fatwrite mmc</i> <i>fatload mmc</i> <i>arcipc</i> <i>md</i>	0 <addr> <File>	mmc to RAM address. Using IrfanView software (any raw image viewer s/w) view raw output data. If output image is blended image of inputs then blending test is successfully completed.
MALI 400 GPU	<i>gputest version</i> <i>gputest</i> <i>gputest</i> <i>gputest</i> <i>gputest</i>	-- gp 0 pp0 0 pp1 0 mp1 0	Prints GP and PP version. Runs geometric processor test0. Runs pixel processor test0. Runs pixel processor test1. Runs MMU tests along with GP & PP.
SD/MMC	<i>mmc_start</i> <i>fatls mmc</i> <i>md</i> <i>md</i> <i>env export</i> <i>md</i> <i>fatwrite mmc</i> <i>fatls mmc</i> <i>fatls mmc</i> <i>md.</i> <i>mw.</i> <i>md.</i> <i>md</i> <i>fatload mmc</i> <i>md</i>	-- 0 <Addr> <Addr> <Addr> serverip <Addr> 0 <Addr> <File> <size> 0 0 <Addr> <Addr> <data> <Len> <Addr> <Addr> 0 <Addr> <File> <Addr>	Export environment to known address. Check if environment is copied to the specified location. Write data to a file in file system on sd card using fatwrite. Check if file has been written to the fat filesystem on the sd/mmc card. Read the dumped file to memory location on RAM using fatload.
GNSS	<i>gnss_test</i>	--	This test starts GPS. Keep the tests running till some satellite data is obtained on GPS uart. (UART-1).

APPENDIX - I

➤ *ENVIRONMENT VARIABLE SETTING IN U-BOOT*

In Shikhara U-boot, environment variables are stored in QSPI Flash. To configure environment settings we need to provide following details:

```
#define CONFIG_ENV_SIZE          (64<<10)          /* Maximum total size of env variables*/
#define CONFIG_ENV_SECT_SIZE    CONFIG_ENV_SIZE
#define CONFIG_ENV_IS_IN_SPI_FLASH /* ENV is stored in spi flash (QSPI)*/
#define CONFIG_ENV_OFFSET       0Xc0000           /* ENV starting address in memory */
#define CONFIG_ENV_OVERWRITE     /* supports to overwrite default values*/
#define CONFIG_CMD_SAVEENV       /* Command to save ENV to Flash */
```

In Shikhara U-boot, there are some default environment variables are available in flash. To overwrite or print the environment variables present in flash we can use standard u-boot commands save and print.

Setting Environment Variables:

To change or create new environment variables in Shikhara U-boot, we use generic command “**setenv**”. Using this command we can set own environment variables or overwrite already existed environment variables. After setting if we want these variables to be maintained after power off, we need to save these variables in flash. To save them we can use u-boot generic command “**saveenv**”.

Usage:

```
shikhara-uboot> set bootdelay 3
shikhara-uboot> saveenv
Saving Environment to SPI Flash...
Erasing SPI flash...Writing to SPI flash...done
```

Printing Environment Variables:

To know all environment variables present in current U-boot, we can use generic command “**printenv**”.

Usage:

```
shikhara-uboot> printenv
baudrate=115200
bootcmd=run modeboot
bootdelay=3
stderr=serial
stdin=serial
stdout=serial
Environment size: 150/65532 bytes
```

APPENDIX - II

➤ *U-Boot Modifications:*

The following are the files which are modified or newly added to the opensource u-boot version u_boot-2012-04 to make it work on Shikhara board.

Opensource U-Boot version used for Shikhara: **u_boot-2012-04.**

>>>Shikhara_Uboot/arch/arm/cpu/armv7/

- Makefile
- config.mk
- cpu.c
- load_uboot.c(This file Added newly)
- start.S

>>>Shikhara_Uboot/arch/arm/include/asm/

- **arch-anuboard/**
 - dma.h
 - dma_code.h
 - gpu_test.h
 - serial_pl01x.h
 - serial_xpsuart.h
 - shikhara-usbgadget.h
 - shikhara_clock.h
 - shikhara_dwmmc.h
 - shikhara_gic.h
 - shikhara_hdmi.h
 - shikhara_hdmi_bridge.h
 - shikhara_i2c.h
 - shikhara_i2s.h
 - shikhara_irq.h
 - shikhara_map.h
 - shikhara_mipi_csi.h
 - shikhara_mipi_csi_bridge.h
 - shikhara_mipi_dsi_bridge.h
 - shikhara_mipi_dsi_hal.h
 - shikhara_mipi_dsi_local.h
 - shikhara_nvm.h
 - shikhara_pmu_sysctrl.h
 - shikhara_rtc.h
 - shikhara_sgtl5000.h

- shikhara_spi.h
- shikhara_tc.h
- shikhara_xhci.h
- slcr.h
- start_gps.h
- usrapi.h
- hardware.h
- nand.h
- sev_segment.h
- shikhara_can.h
- shikhara_gpio.h
- shikhara_jack.h
- shikhara_kmi.h
- shikhara_kmi_pc_keyb.h
- shikhara_max98089.h
- shikhara_pmu.h
- shikhara_smc.h
- shikhara_sound.h
- shikhara_wdt.h
- xparameters_ps.h
- **arch-anuboard/bluetooth (Added newly)**
 - shikhara_bt_defines.h
 - shikhara_bt_irq_defines.h
 - shikhara_bt_register_defines.h
- **arch-anuboard/dma/**
 - dma.h
 - pl330.h
 - shikara-dma-pl330.h
 - shikara-pl330-pdata.h

>>Shikhara_Uboot/arch/arm/include/asm/arch-armv7/

- sysctrl.h
- systimer.h
- wdt.h

>> Shikhara_Uboot/bl0_spl and Shikhara_Uboot/bl1_spl

>> Shikhara_Uboot/board/shikhara_anu/

- anusoc/
 - > **This directory and all files under this are newly added**
 - shikhara_helper.c
 - shikhara_pmu.c
 - shikhara_pmu_sysctrl.c
 - zynq_timer.c
 - Makefile*

- shikhara_board.c
- shikhara_gic.c
- shikhara_timer.c
- config.mk
- coretile_ddr_init.ds
- ddr_init.S
- pl341_ddr_init.S
- shikhara_pmu_a9.S

- validation/

-> This directory and all files under this are newly added

- cmd_apkybd.c
- cmd_apmouse.c
- cmd_arc_ipc.c
- cmd_atp.c
- cmd_av417reset.c
- cmd_bt.c
- cmd_bttest.c
- cmd_bttest_2lp.c
- cmd_clcd_hw_cursor.c
- cmd_clcd_preview.c
- cmd_dma_uartapi.c
- cmd_dmatest.c
- cmd_dsi_preview.c
- cmd_dwmmc.c
- cmd_gicdump.c
- cmd_gnss.c
- cmd_gnss_intr_min.c
- cmd_gpio_led.c
- cmd_gpu_scene.c
- cmd_hdmi_audio_min.c
- cmd_hdmi_dvi_min.c
- cmd_hdmi_preview.c
- cmd_i2c_mintest.c
- cmd_i2cdump.c
- cmd_i2sdump.c
- cmd_mem_dma_min.c
- cmd_mincan.c
- cmd_nandmintest.c
- cmd_nandtest.c
- cmd_nortest.c
- cmd_ov5640_preview.c
- cmd_pmutest.c
- cmd_post.c
- cmd_raw_display.c
- cmd_raw_videoplayer.c
- cmd_sev_segment.c
- cmd_sevseg_test.c

- cmd_shikhara_sw_cursor.c
- cmd_spi_loopbacktest.c
- cmd_spi_minitest.c
- cmd_spidump.c
- cmd_spislave.c
- cmd_tzasc.c
- cmd_usb_intr.c
- cmd_usbdev_dump.c
- cmd_usbdump.c
- cmd_usbmintest.c
- cmd_yuvtobmp.c
- Makefile
- cmd_bmp_slideshow.c
- cmd_ddrtest.c
- cmd_dttest.c
- cmd_gpio_input_test.c
- cmd_gpioloopback.c
- cmd_gpu.c

>>>Shikhara_Uboot/common/

-> **This files are added newly**

- cmd_can.c
- cmd_custom_lcd.c
- cmd_flyer.c
- cmd_powerdown_cpu.c
- cmd_sound.c
- cmd_usb3.c

-> **And modified few files also.**

- cmd_bedbug.c
- cmd_bmp.c
- cmd_bootm.c
- cmd_eeprom.c
- cmd_i2c.c
- cmd_mem.c
- cmd_usb.c
- cmd_version.c
- command.c
- console.c
- env_common.c
- image.c
- main.c
- lcd.c
- usb.c
- usb_hub.c
- usb_storage.c

>>>Shikhara_Uboot/drivers/

-> This files are added newly under drivers directory.

- /can
 - can.c
 - shikhara_can.c
- /dma330
 - pl330.c
 - shikhara_pl330.c
- /gpu
 - MaliFns-generic.c
 - MaliFns-m400.c
 - MaliFns-m400perf.c
 - MaliFns-sysintf.c
 - MaliTop.c
 - shikhara_gpu.c

-> This files are modified.

- /block
 - systemace.c
- /gpio
 - pl061.c
- /i2c
 - shikhara_i2c.c
- /input
 - tc_calibrate.c
 - keyboard.c
 - shikhara_kmi.c
 - shikhara_kmi_pc_keyb.c
 - shikhara_tc.c
- /mmc
 - dw_mmc.c
 - shikhara_dw_mmc.c
 - shikhara_dw_mmc_core.c
 - zynq_sdhci.c
 - arm_pl180_mmci.c
 - arm_pl180_mmci.h
 - mmc.c
 - sdhci.c
- /mtd/nand/
 - shikhara_nand.c
 - nand.c
 - nand_base.c
 - nand_bbt.c
 - nand_util.c
- /mtd/spi/

- atmel.c
- spansion.c
- spi_flash.c
- stmicro.c
- **/rtc**
 - shikhara_rtc.c
- **/serial**
 - serial_pl01x.c
- **/sev_segment**
 - sev_segment.c
- **/sound/**
 - shikhara_hdmiaudio_new3.c
 - shikhara_hdmichannels.c
 - shikhara_max98089.c
 - shikhara_sgtl5000.c
 - shikhara_i2s.c
 - shikhara_jack.c
 - sound.c
- **/spi/**
 - slcr.c
 - spi.c
 - shikhara_spi.c
 - shikhara_spi_dma.c
 - zynq_qspi.c
- **/usb/gadget/**
 - regs-usb-device.h
 - usbd3-ss.h
 - usbd3-ss.c
- **/usb/host/**
 - xhci-shikhara.h
 - xhci.h
 - xhci-mem.c
 - xhci-ring.c
 - xhci-shikhara.c
 - xhci.c
 - regs-usb3-shikhara-drd-phy.h
- **/video**
 - shikhara_fb.c
 - shikhara_hdmi_bridge.c
 - shikhara_hdmi_helper.c
 - shikhara_hdmi_new.c
 - shikhara_mipi_csi.c
 - shikhara_mipi_csi_bridge.c
 - shikhara_mipi_dsi.c
 - shikhara_mipi_dsi_bridge.c
 - shikhara_mipi_dsi_hal.c
 - shikhara_csi_dsi_helper.c

- /watchdog
 - sp805_wdt.c

>>Shikhara_Uboot/include/

- /configs
 - anusoc.h -> This file is added newly
 - mipi_display.h
 - MaliFns.h
 - Page_Table_PP0_PP1_GP_13MB.h
 - amba_clcd.h
 - can.h
 - common.h
 - config_cmd_all.h
 - config_cmd_default.h
 - div64.h
 - dwmmc.h
 - flyer.h
 - hw_can.h
 - i2c.h
 - keyboard.h
 - keymap.h
 - lcd.h
 - mmc.h
 - nand.h
 - netdev.h
 - ns9750_bbus.h
 - ns9750_mem.h
 - ns9750_ser.h
 - ns9750_sys.h
 - part.h
 - sdhci.h
 - shikhara_mali.h
 - sound.h
 - spi.h
 - spi_flash.h
 - systemace.h
 - usb.h
 - usb_defs.h
 - usbdescriptors.h

APPENDIX – III

➤ *Porting U-Boot to Shikhara board*

If we want to port a uboot to the custom board, need to create a new set of directories and files to the existing configuration. The necessary directories and files are:

```
-> u-boot/board/<board_name>
-> u-boot/include/configs/<boardname>.h
-> u-boot/Makefile
```

Adding New Command

Commands are added to U-Boot by creating a new command structure. This is done by first including command.h, then using the U_BOOT_CMD() macro to fill in a cmd_tbl_t structure.

U_BOOT_CMD(name,maxargs,repeatable,command,"usage","help")

name : is the name of the command. THIS IS NOT a string.
maxargs : the maximum number of arguments this function takes
repeatable : either 0 or 1 to indicate if autorepeat is allowed
command : Function pointer (*cmd)(struct cmd_tbl_s *, int, int, char *[]);
usage : Short description. This is a string
help : Long description. This is a string

Steps to be follow to add a new command :

- Need to create a new file under the common command folder under u-boot (u-boot/common/) with your own suitable name.
- And need to make a entry in the Makefile in common folder with the newly added file.

For example:

We have added a sample test command in the u-boot under the common command directory,
--> u-boot/cmd/test_cmd.c

This is just an test command and just prints as just a test command in the console after running this command. After creating a file, have to make a entry in the makefile. We can make a entry with flag macro should be defined or else you can define as,

--> obj-y += test_cmd.o

The above example explains the procedure to add an new command to the u-boot.