StarterWare User Guide

ADAS StarterWare

User Guide

1 Table of Contents

1. Introduction

StarterWare provides no-OS platform support for TI ADAS series of SoCs: TDA1Mxx, TDA2xx, TDA2Ex and TDA3xx. The StarterWare package contains Device Abstraction Layer libraries and peripheral/board level sample/demo examples that demonstrate the capabilities of the peripherals on TDA1Mxx, TDA2xx, TDA2Ex and TDA3xx.

TDA1Mxx device family is a derivative of TMS320DM8148 that supports Advanced Driver Assistance Systems (ADAS) applications. For more information about the TDA1Mxx device family, please contact your local TI sales representative. For more information about TMD320DM814x, please visit: http://www.ti.com/product/tms320dm8148.

TDA2xx and TDA2Ex are high-performance, automotive vision application devices based on enhanced OMAP™ architecture integrated on a 28-nm technology. The architecture is designed for Advanced Driver Assistance applications, including Vision Analytics for Single/Dual Front Camera, LVDS/Ethernet Surround View, Night Vision, Blind Spot Detection, Sensor Fusion and LIDAR, among others, and best-in-class CPU performance, video, image, and graphics processing sufficient to support
– Streaming video up to full high definition (Full-HD) (1920×1080p, 60 fps)
– 2-dimensional (2D) and 3-dimensional (3D) graphics.

TDA3x is an ADAS application device based on enhanced OMAP™ architecture integrated on a 28-nm technology. TDA3x complements the TDA2x ADAS device family by using a common architecture, enabling scalability from entry to high performance for a broad range of applications. The device family is targeted at ADAS applications including Front Camera, Intelligent Rear Camera, Radar and Mirror Replacement.

The purpose of this *User Guide* is to provide more detailed information regarding the software elements and infrastructure provided with StarterWare. The software provided is intended to be used as a reference when starting application software development.

1.1 Acronyms and Definitions

The following acronyms are used throughout this document.

Acronym	Meaning
CCS	Texas Instruments Code Composer Studio
DDR	Double Data Rate
DSP	Digital Signal Processor
EDMA	Enhanced Direct Memory Access
EVM	Evaluation Module, hardware platform containing the Texas Instruments DSP
IPC	Texas Instruments Inter-Processor Communication Development Kit
JTAG	Joint Test Action Group
RAM	Random Access Memory
DHCP	Dynamic Host Configuration Protocol
TI	Texas Instruments
UART	Universal Asynchronous Receiver/Transmitter

CG Tools	Code Generation Tools
SBL	Secondary Boot Loader
XDAIS	eXpressDSP Algorithm Interface Standard

1.2 Supported Devices/Platforms

TDA1Mxx: This release supports TDA1Mxx [1] (PG2.1, PG3.0).
TDA2xx: This release supports TDA2xx series of SoC (PG1.0, PG1.1 and PG2.0).
TDA2Ex: This release supports TDA2Ex series of SoC (PG1.0).
TDA3xx: This release supports TDA3xx series of SoC (PG1.0)

1.3 Other Documentation

Document	**Description**
StarterWare_ReleaseNotes.pdf	Contains latest information on the release including what's changed, known issues and compatibility information.
StarterWare_API_Reference.chm	API guide for the StarterWare.
SBL_UserGuide.pdf	SBL user-guide describes the SBL support features, boot modes, build & flash the bootloader & multicore application image. It is located in <install_dir>/bootloader/ SBL_UserGuide.pdf
StarterWare_DataSheet.pdf	Data sheet for StarterWare. Contains memory statistics for the different libraries.

2. Hardware Overview

TDA2SEDX VPS Hardware Introduction
Video Processing system comprises of VIP for video capture, VPE for processing on video data, and DSS for video display output.

Video Input Port (VIP)
Video Input Port (VIP) is the video capture interface on TDA2SEDX. There are 3 instances of VIP in TDA2SEDX. Each instance of VIP comprises of 2 slices and 1 VPDMA. Each slice in turn has 2 input ports Port A and Port B. Port A can be configured for 24bit (YUV444/RGB888), 16bit (YUV422), 8bit time interleaved (YUV422). Port B is only 8bit time interleaved (YUV422). This implies that on Vayu, 12 8bit captures is possible using 3 VIPs, 6slices and 2 ports (A and B).

Supported Features:
- Support for embedded (BT.656/BT.1120 16/24b, BT.656 8b) or discrete (BT.601 style) sync
- Embedded Sync data interface mode (can support multiplexed sources)
- Discrete Sync data interface mode (only supports single source/non-multiplexed inputs)
- 24b data input plus discrete syncs, can be configured to include:
  - 8b YUV422 (Y and U/V time interleaved)
  - 16b YUV422 (CbY and CrY time interleaved)
  - 24b YUV444
  - 16b RGB656
  - 24b RGB888
  - 8b RAW Capture
  - 16b RAW Capture
  - 24b RAW Capture
- Discrete sync modes include:
  - VSYNC + HSYNC (FID determined by FID signal pin or HSYNC/VSYNC skew)
  - VSYNC + ACTVID + FID
  - VBLANK + ACTVID (ACTVID toggles in VBLANK) + FID
  - VBLANK + ACTVID (no ACTVID toggles in VBLANK) + FID
- Multichannel parser (embedded syncs only)
  - Pixel (2x or 4x) or Line multiplexed modes supported
  - Performs de multiplexing and basic error checking
  - Supports maximum of 9 channels in Line Mux (8 normal + 1 split line)
- Ancillary data capture support
  - For 16b or 24b input, ancillary data may be extracted from any single channel
  - For 8b time interleaved input, ancillary data can be chosen from the Luma channel, the Chroma channel, or both channels
  - Horizontal blanking interval data capture only supported when using discrete syncs (VSYNC + HSYNC or VSYNC + HBLANK)
  - Ancillary data extraction supported on multichannel capture as well as single source streams
- Format conversion and scaling
  - Programmable color space conversion
  - 422 => 444 conversion
  - 444 => 422 conversion
  - 422 => 420 conversion
  - YUV444 Source:
    YUV444 =>YUV444, YUV444 =>RGB888, YUV444 =>YUV422, YUV444 =>YUV420
  - RGB888 Source:
    RGB888 =>RGB888, RGB888 =>YUV444, RGB888 =>YUV422, RGB888 =>YUV420
  - YUV422 Source:
    YUV422 =>YUV422, YUV422 =>YUV420, YUV422 =>YUV444, YUV422 =>RGB888
  - Supports RAW to RAW (no processing)
  - Scaling and format conversions do not work for multiplexed input
- Support for 1080P input with scaling and Chroma up/down sampling (1920 wide line buffers)

- VPDMA can transfer the capture data into the buffer in external memory or OCMC memory.
  - VPDMA output buffer size restriction feature ensures that writes not exceed allocated memory buffer size
  - Support for Tiled (2D) and raster addressing without bandwidth penalty
  - Dual clients per channel allows for capture of scaled and non-scaled versions of the data stream (non-multiplexed mode only)
  - Start on new frame capability
  - Interrupt every X number of frames
  - Interrupt every X lines (synced to frame start)

TDA3XX VPS Hardware Introduction
Video Processing system comprises of VIP for video capture and DSS for video display output.

Video Input Port (VIP)

Video Input Port (VIP) is the video capture interface on TDA2SEDX. There are 3 instances of VIP in TDA2SEDX. Each instance of VIP comprises of 2 slices and 1 VPDMA. Each slice in turn has 2 input ports Port A and Port B. Port A can be configured for 24bit (YUV444/RGB888), 16bit (YUV422), 8bit time interleaved (YUV422). Port B is only 8bit time interleaved (YUV422). This implies that on Tda3xx, 4 8bit captures is possible using 1 VIP, 2 slices and 2 ports (A and B).

Supported Features:
- Support for embedded (BT.656/BT.1120 16/24b, BT.656 8b) or discrete (BT.601 style) sync
- Embedded Sync data interface mode (can support multiplexed sources)
- Discrete Sync data interface mode (only supports single source/non-multiplexed inputs)
- 24b data input plus discrete syncs, can be configured to include:
  - 8b YUV422 (Y and U/V time interleaved)
  - 16b YUV422 (CbY and CrY time interleaved)
  - 24b YUV444
  - 16b RGB656
  - 24b RGB888
  - 8b RAW Capture
  - 16b RAW Capture
  - 24b RAW Capture
- Discrete sync modes include:
  - VSYNC + HSYNC (FID determined by FID signal pin or HSYNC/VSYNC skew)
  - VSYNC + ACTVID + FID
  - VBLANK + ACTVID (ACTVID toggles in VBLANK) + FID
  - VBLANK + ACTVID (no ACTVID toggles in VBLANK) + FID
- Multichannel parser (embedded syncs only)
  - Pixel (2x or 4x) or Line multiplexed modes supported
  - Performs de multiplexing and basic error checking
  - Supports maximum of 9 channels in Line Mux (8 normal + 1 split line)
- Ancillary data capture support
  - For 16b or 24b input, ancillary data may be extracted from any single channel
  - For 8b time interleaved input, ancillary data can be chosen from the Luma channel, the Chroma channel, or both channels
  - Horizontal blanking interval data capture only supported when using discrete syncs (VSYNC + HSYNC or VSYNC + HBLANK)
  - Ancillary data extraction supported on multichannel capture as well as single source streams
- Format conversion and scaling
  - Programmable color space conversion
  - 422 => 444 conversion
  - 444 => 422 conversion
  - 422 => 420 conversion
  - YUV444 Source:

YUV444 =>YUV444, YUV444 =>RGB888, YUV444 =>YUV422, YUV444 =>YUV420

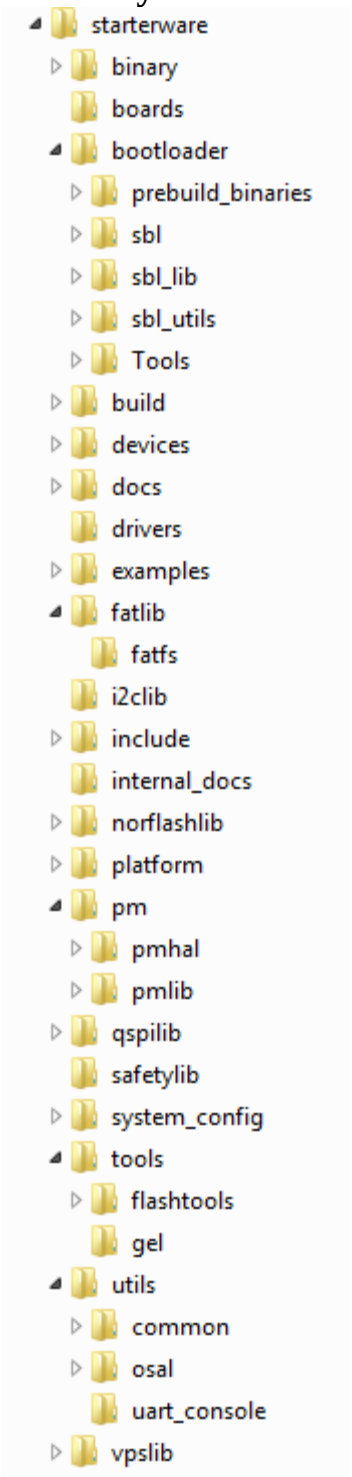
- RGB888 Source:
  RGB888 =>RGB888, RGB888 =>YUV444, RGB888 =>YUV422, RGB888 =>YUV420
- YUV422 Source:
  YUV422 =>YUV422, YUV422 =>YUV420, YUV422 =>YUV444, YUV422 =>RGB888
- Supports RAW to RAW (no processing)
- Scaling and format conversions do not work for multiplexed input

- Support for 1080P input with scaling and Chroma up/down sampling (1920 wide line buffers)
- VPDMA can transfer the capture data into the buffer in external memory or OCMC memory.
  - VPDMA output buffer size restriction feature ensures that writes not exceed allocated memory buffer size
  - Support for Tiled (2D) and raster addressing without bandwidth penalty
  - Dual clients per channel allows for capture of scaled and non-scaled versions of the data stream (non-multiplexed mode only)
  - Start on new frame capability
  - Interrupt every X number of frames
  - Interrupt every X lines (synced to frame start)

3. StarterWare Overview

StarterWare provides a no-OS platform support for multi-core SoCs. StarterWare provides Device Abstraction Layer (DAL) libraries and peripheral/board level sample/demo examples that demonstrate the capabilities of the peripherals. This package includes the following components:

Directory Structure



- **Binary -** The entire executables are placed in this directory. The generated library or executable/binary is placed in an appropriate path under tool specific directory.
- **Boards -** This contains list of on-board devices/daughter card such as Vision Application board, Custom board, etc. Contains routines to auto-detect the daughter card/board type. Any special programming needed by the board is also added here.
- **Bootloader –** This contains the secondary bootloader specific files.
    - SBL library layer: This layer contains APIs needed for parsing and loading multicore App Image and booting the slave cores. This library is used by both bootloader application and master application.
    - SBL utility layer: This layer is used only by the bootloader application and contains APIs for communicating between boot media and master core
    - SBL application: This contains the bootloader application that loads and boots the application image.
- **Build -** This folder will contain the Makefile required to build libraries (drivers, system_config, platform, utils, i2clib, etc.) and example applications (Uart, mcspi, hsi2c etc.). The StarterWare only supports Makefile based build.
- **Devices -** This contains the I2C API files and APIs to control IO expanders. I2C could be used to read / write into any of the video on-board devices such as Sii9022a, TVP7002, IO Expanders or sensors such as MT9V022, etc.
- **Docs** – This folder contains various collaterals like StarterWare user guide, release notes, data sheet, etc.
- **Drivers -** This directory contains the device abstraction layer API source files for supported peripherals.
- **Examples -** Examples provided as part of the package showcase some (not all) functionality of the peripheral. This depends on the availability of a peripheral instance and its association, Pin Multiplex settings etc. on the EVM. This folder contains application level utility functions which could be used as is or with minimal modification.
- **Fatlib -** This folder contains source code for FAT32 file system library. This provides a thin file system interface for the applications to access the devices having FAT32 file system and provides APIs for the file based access. However there are a few limitations with fatlib that are listed below:
    - Tested and enabled on IPU1-0 only (in theory it can run on IPU1-1 and A15 also but not tested)
    - When MMCSD filesystem is enabled, networking is disabled (due to pinmux limitations)
    - Run-time card removal/insertion wont re-detect the SD card
    - Works on 1 partition only
- **I2clib -** This folder contains source files for the I2C library.
    - I2C lib can be configured to operate either in POLLED, INTERRUPT or DMA mode(FIFO or NON-FIFO mode)
    - In polled mode the transfer status is returned from the transfer API (lld_i2c_transfer() API).

- In Interrupt and DMA mode transfer API (lld_i2c_transfer()) returns just the transfer initialization status and actual transfer status is returned in the application callback function registered using lld_i2c_open() API. lld_i2c_transfer() API is not a blocking call and application has to wait till the application callback function is called after transfer and take appropriate action depending on the transfer status returned in the callback function.
- I2C can be operated in FIFO mode. API lld_i2c_SetFifoThreshold() is used to set FIFO threshold valve. This API should be called before calling lld_i2c_open() and after calling lld_i2c_init(). I2C LIB is validated with 8 and 16 byte FIFO.
- Changing I2C clock on demand: Added lld_i2c_clockConfig() API in I2C LIB which can be used to change I2C clock on demand. This API has to be called after calling lld_i2c_open().
- Optimized I2C lib: The I2C lib is optimized to depend only on the I2C status and unwanted delays to check the status is removed. This may lead to behavioral change in I2C lib and it may require more timeout parameter to be passed from the application. If slave device needs some time to process internal data, it has to be taken care in application (Ex: EEPROM requires 5ms of time to flash data internally after the I2C transfer is complete).

- **Include -** The header files for inclusion are all placed under include/.
  - The include files are classified as, User interface driver headers: For example include/mcspi.h, which contain macro definitions used by peripheral APIs and prototypes of the peripheral APIs.
  - Processor Family and SoC specific files: Certain processor family specific files like include/armv7a/cpu.h.
  - include/armv7a/mmu.h and SoC specific files like include/am335x/interrupt.h are present.
  - Register Layer files: Files like include/hw/hw_mcspi.h which contain the peripheral register offset macros and register field token macros.
- **Norflashlib -** This folder contains source code for NOR flash library. Generic API's are provided top configure, read and write to the NOR flash.
  - Supports AMD and Intel based NOR flash commands.
- **Platform -** Every supported EVM is called as a (hardware) platform inside the package. This exports functions specific to an EVM that usually do (as required) Peripheral Pin Multiplexing settings, Peripheral Clock settings, EVM Profile settings, I/O expander settings etc. to enable a peripheral operation on the platform. Peripheral Pin Multiplexing though depends only on the SoC Silicon Package; the availability of the external ports for communication is decided by the EVM. The code in platform is maintained as an entity distinct from the applications/examples to provide a simpler look at the first level.
- **PM:** The power management (PM) software enables optimal power consumption and thermal management scalable across TDA2xx, TDA2ex and TDA3xx. It is divided into three layers namely:
  - **PRCM Database (DB)**: abstracts the SoC specific PRCM details regarding the registers, the partitioning of the device and the clock tree details.

  - **Power Manager Hardware Abstraction Layer (PMHAL)**: abstracts the programming sequence of atomic power management actions like configuring a PD, CD, PLL, bandgap thermal sensors etc.
  - **Application Interface Layer (PM Library - PMLIB):** abstracts all PRCM architecture details to the application where the developer provides top level requests of switching on or off a module or configuring a certain module clock to a certain desired frequency and putting CPUs to low power.

- **Qspilib -** This folder contains source code for QSPI flash library. This provides API's to configure and access the QSPI flash. The initialization API takes in device type as parameter and configures the flash to be read in one bit or 4 bit mode.
  - API's are provided to read and write from the flash in cfg port mode, memory mapped mode (using CPU and EDMA).
  - API's are provided to erase the flash (single block or entire flash).
- **Safetylib –** This folder contains source files of safety library. This provides higher level APIs written on top of safety IP drivers (DAL).
  - Safety lib can be used to configure L3 Firewall.
  - Safety lib can be used to configure CRC to calculate CRC signature of given data.
- **System_config -** The system configuration and initialization code like the start-up code, interrupt vector initialization, low level CPU specific code etc. are provided here. Since this may involve assembly level coding, such code is placed under tools specific directory.
- **Tools -** This directory contains the various flash tools and a few gel files containing small snippets of code that are needed to run the Starterware examples.
- **Utils -** This contains user utilizable code. As an example, utils/uartStdio.c contains wrapper functions which use UART APIs to interface with the user through the serial console.
- **Vpslib -** This folder contains all the source files for DSS and VIP. It has three layers HAL, Core and a thin driver layer for VIP and DSS IPs.
  - **HAL:** HAL is hardware abstraction layer which has API to program the registers. Core layer will sequence the register writes required for a particular operation. Driver layer logically groups the Core API's so that application will be simpler.
  - **VIP:** VIP is a hardware block in TDA2xx to capture data from external video source like video decoders (example, TVP5158, TVP7002) or sensors (example, MT9V022). The video data is captured from the external video source by the VIP Parser sub-block in the VIP block. The VIP Parser then sends the captured data for further processing in the VIP block which can include color space conversion, scaling, Chroma down sampling and finally writes the video data to external DDR memory or an OCMC buffer.

    Note: Currently VIP init will do initializations for all the three instances of VIP present in TDA2XX platform, if only one or two instances are used and have limitations on the memory then number of VIP's to be initialized can be configured in soc_tda2xx.h

file.CSL_TDA2XX_VPS_VIP_PER_CNT macro needs to be set to number of VIP's to be initialized.

- **DSS:** DSS module will take the video/Graphics buffer from the memory and displays the video/Graphics on the video encoder (VENC) at specified frame rate and resolution.
  In TDA2XX DSS has three video pipelines and one graphics pipeline. It has got four overlays, 3 DPI outputs and one HDMI output.
- **ISS:** Imaging Sub System – Provides ability to receive video streams (via CSI2, Parallel & LVDS interfaces), perform operations such as WDR (Wide Dynamic Range) merge, convert RAW / Bayer streams to YUV streams, correct distortions introduced by lens, Noise filtering, etc.

4. Installation

To install ADAS StarterWare on your PC run the StarterWare installer (starterware_setupwin32_xx_xx_xx_xx.exe). The installer allows you to choose the installation directory. The StarterWare includes several sub-components and all the components will be installed in the same location (e.g., " C:/ti/ starterware_xx_xx_xx_xx").

5. Tool Chain versions

Refer to corresponding Release notes for Tool chain's information.

6. Build Steps

This section describes how to build the StarterWare package. The package is built using the gmake from Cygwin. Make sure that the Cygwin tools location is added to the PATH variable. Also Makefile internally uses some binaries like rm, mkdir echo from Cygwin.

Edit the "env.mk" file present in "build\makerules" to give the appropriate paths for the tool chains. Make sure that UTILS_INSTALL_DIR and the CODEGEN_PATH_<CORE> are updated with the proper values. E.g. If A15 Linaro tool chain is present at a different location; update the CODEGEN_PATH_A15 in this file.

To build all the examples and libraries you can give the following command.

$*gmake all PLATFORM=<platform>*

<platform> can be 'ti814x', 'tda2xx', 'tda2ex' and 'tda3xx'. This would build all the libs and examples in release mode.

Command to build all the libraries:

$*gmake libs PLATFORM=ti814x/tda2xx/tda2ex/tda3xx*

Command to build the StarterWare hal library:

$*gmake starterware_hal PLATFORM==<platform>*

Command to build an application given in examples folder:

$*gmake <app_name> PLATFORM=<platform>*

Command to clean all targets:

$*gmake -s clean PLATFORM=<platform>*

Command to clean individual targets:

$*gmake -s $(target)_clean PLATFORM=<platform>*

app_name can be found in the make file of the respective examples folder.

The build system requires the RTS library present as part of the TI CG tools. If the library is not already built make reports an error while linking. You can build the library by opening command prompt at the <CG tools directory>/libs and run the following command.

$mklib –pattern=<RTS library name> (i.e. rtsv7A8_A_le_n_eabi.lib)

Gmake and <CG tools directory>/bin should be added to system PATH environment variable to run above command.

6.1. A8 Multi Tool chain Support

A8 libraries/binaries can be built either using TMS470 tool chain or GCC tool chain. User can specify the tool chain to use by setting parameter TOOLCHAIN_a8 while building.
TOOLCHAIN_a8 can take the following values:

- cgt for TMS470 tool chain
- gcc for GCC tool chain

If TOOLCHAIN_a8 is not set during build, A8 libraries/binaries will be built using TMS470 tool chain by default.
To build all A8 libraries/binaries using GCC tool chain, use the below command:
$*gmake all PLATFORM=ti814x TOOLCHAIN_a8=gcc*
This would build all the libs and examples in release mode.

Command to build all the libraries:
$*gmake libs PLATFORM=ti814x TOOLCHAIN_a8=gcc*

Command to build the StarterWare hal library:
$*gmake starterware_hal PLATFORM=ti814x TOOLCHAIN_a8=gcc*

Command to build an application given in examples folder:
$*gmake <app_name> PLATFORM=ti814x TOOLCHAIN_a8=gcc*

Command to clean all targets:
$*gmake clean PLATFORM=ti814x TOOLCHAIN_a8=gcc*

Command to clean individual targets:
$*gmake $(target)_clean PLATFORM=ti814x TOOLCHAIN_a8=gcc*

6.2. Debug Mode

StarterWare package can be built in two modes:

- Release Mode
- Debug Mode

User can build in either mode by setting parameter PROFILE while building.

$*gmake all PLATFORM=<platform> PROFILE=release/debug*

PROFILE can take the following values:

- release for Release Mode
- debug for Debug Mode

If PROFILE is not set during build, starterware will be built for release profile by default.

To build all the examples and libraries in debug mode, you can give the following command.

$*gmake all PLATFORM=<platform>PROFILE=debug*

Similarly, PROFILE can be set in all the above build commands.

7. Secondary Boot loader and Boot Utilities

The bootloader and boot utilities are located in the path <install_path>/bootloader/.
The Secondary Bootloader does the following:

- Program the ADPLLJM & ADPLLM at OPP NOM frequency
- Force wake-up all clock domain
- Switch the module mode to Auto-enable state
- Configure the PAD
- Initialize DDR 2/DDR3
- Configure the CPU MMUs required for boot-up
- Parse & load the multi-core application image
- Boot-up the slave cores

SBL in this release supports TDA2xx, TDA2Ex & TDA3xx platform. For more information on SBL refer the SBL user-guide located under <install_path>/bootloader/ SBL_UserGuide.pdf

7.1. Hardware Setup

List of hardware required for TDA1Mxx:

1. TDA1Mxx EVM (PG2.1)
2. Video Vision Application Board (Beta / Rev B)
3. Video Security Application Board(For VIP examples)
4. OV10630 Sensor (Sensor Config Application)
5. Composite cable
6. Spectrum Digital XDS560 / XDS510 JTAG emulator

List of hardware required for TDA2xx:

1. TDA2xx EVM
2. Video Vision Application Board (For VIP examples)
3. OV10630 Sensor (Sensor Config, Video Loopback Application)
4. Mini USB Cable (for UART Console logs)
5. Spectrum Digital XDS560 v2 JTAG/PCIE JTAG/other EVM supported emulator

List of hardware required for TDA2Ex:

1. TDA2Ex EVM

2. Video Vision Application Board (For VIP examples)
3. OV10630 Sensor (Sensor Config, Video Loopback Application)
4. Mini USB Cable (for UART Console logs)
5. Spectrum Digital XDS560 v2 JTAG/PCIE JTAG/other EVM supported emulator

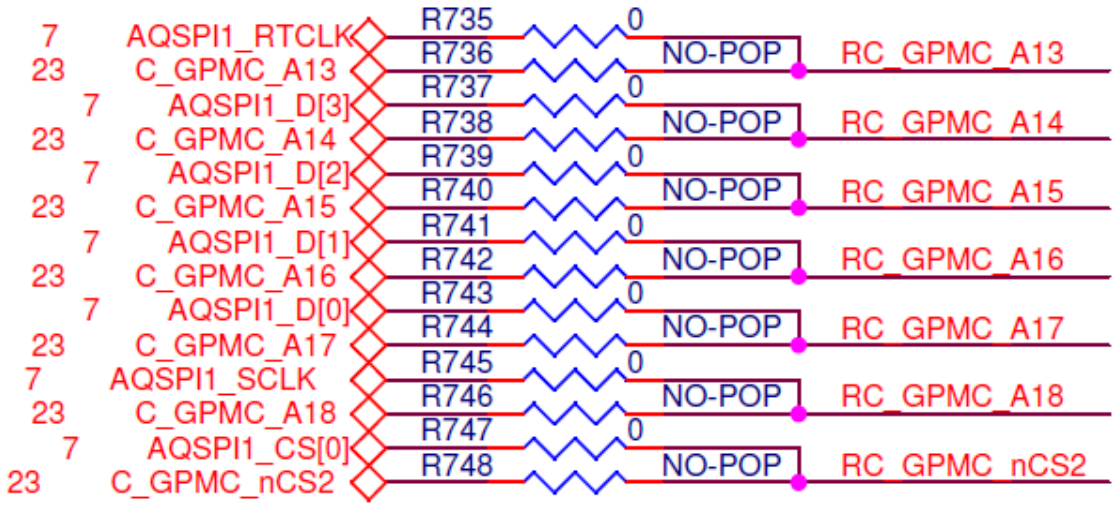
List of hardware required for TDA3xx:

1. TDA3xx EVM
2. OV10630 Sensor (Sensor Config, Video Loopback Application)
3. Mini USB Cable (for UART Console logs)
4. Spectrum Digital XDS560 v2 JTAG/PCIE JTAG/other EVM supported emulator
5. LCD with cable

7.2. Board Modification

TDA2xx Board Modification for NOR BOOT Mode:

By default QSPI's Zero ohm resistor are connected and GPMC lines are opened. To get NOR working, required to remove the QSPI Zero Ohm resistor & pop-up GPMC resistors as stated below.

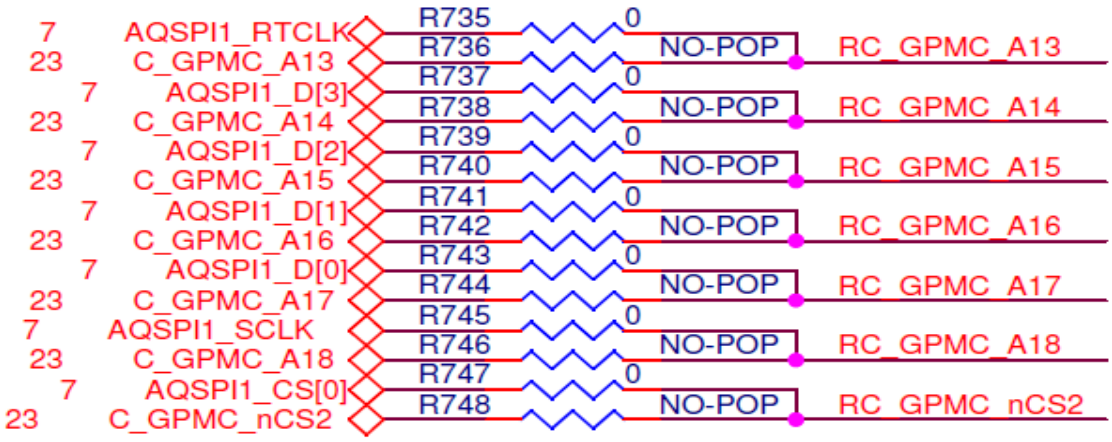


- Remove the R735 resistor and popup at R736 resistor => C_GPMC_A13
- Remove the R737 resistor and popup at R738 resistor => C_GPMC_A14
- Remove the R739 resistor and popup at R740 resistor => C_GPMC_A15
- Remove the R741 resistor and popup at R742 resistor => C_GPMC_A16
- Remove the R743 resistor and popup at R744 resistor => C_GPMC_A17
- Remove the R745 resistor and popup at R746 resistor => C_GPMC_A18
- Remove the R747 resistor and popup at R748 resistor => C_GPMC_nCS2

QSPI will not work on Tda2xx Board which is modified for NOR BOOT Mode. For NOR, EVM switch setting SW5[1:10] - 0100100000

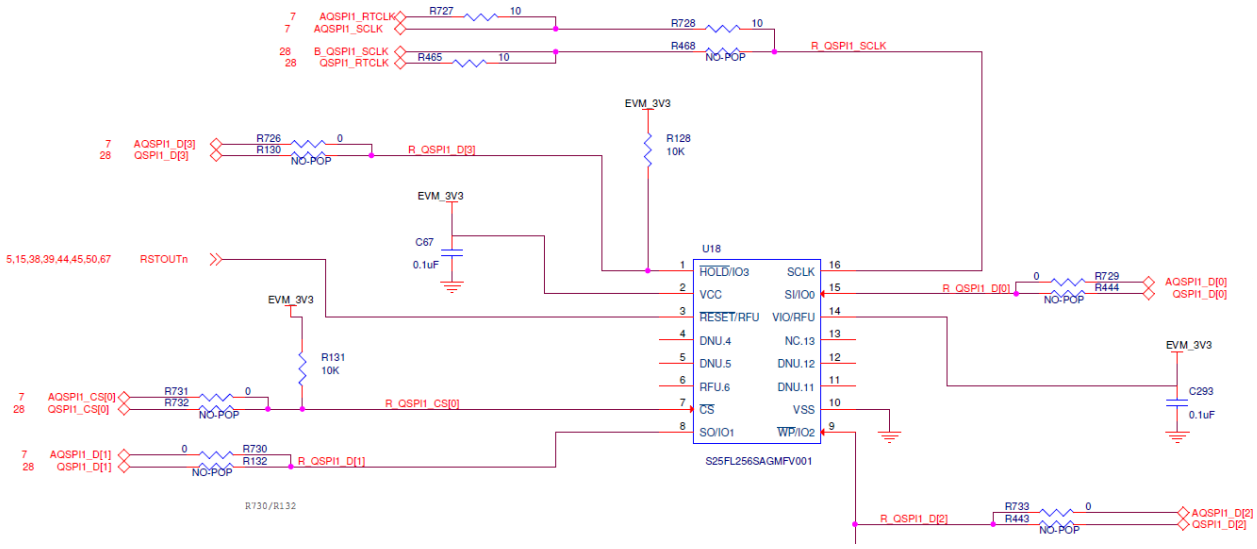
TDA2Ex Board Modification for NOR and QSPI BOOT Mode:

By default QSPI's Zero ohm resistor are connected and GPMC lines are opened. To get NOR working, required to remove the QSPI Zero Ohm resistor & pop-up GPMC resistors as stated below.



- Remove the R735 resistor and popup at R736 resistor => C_GPMC_A13
- Remove the R737 resistor and popup at R738 resistor => C_GPMC_A14
- Remove the R739 resistor and popup at R740 resistor => C_GPMC_A15
- Remove the R741 resistor and popup at R742 resistor => C_GPMC_A16
- Remove the R743 resistor and popup at R744 resistor => C_GPMC_A17
- Remove the R745 resistor and popup at R746 resistor => C_GPMC_A18
- Remove the R747 resistor and popup at R748 resistor => C_GPMC_nCS2

For NOR, EVM switch setting SW5[1:10] – 0100100000. For SPI Flash, by default A-QSPI path resistor are connected and B-QSPI path resistor are opened. To get QSPI working on the TDA2EX Board which is modified for NOR, required to remove the A-QSPI path resistor & pop-up B-QSPI path resistor as stated below.



- Remove the R728 resistor and popup at R468 resistor => B_QSPI1_SCLK, QSPI1_RTCLK
- Remove the R731 resistor and popup at R732 resistor => QSPI1_CS[0]
- Remove the R729 resistor and popup at R444 resistor => QSPI1_D[0]
- Remove the R730 resistor and popup at R132 resistor => QSPI1_D[1]
- Remove the R733 resistor and popup at R443 resistor => QSPI1_D[2]

- Remove the R726 resistor and popup at R130 resistor => QSPI1_D[3]

For QSPI, EVM switch setting SW5[1:10] - 0001100000

7.3. Running the examples

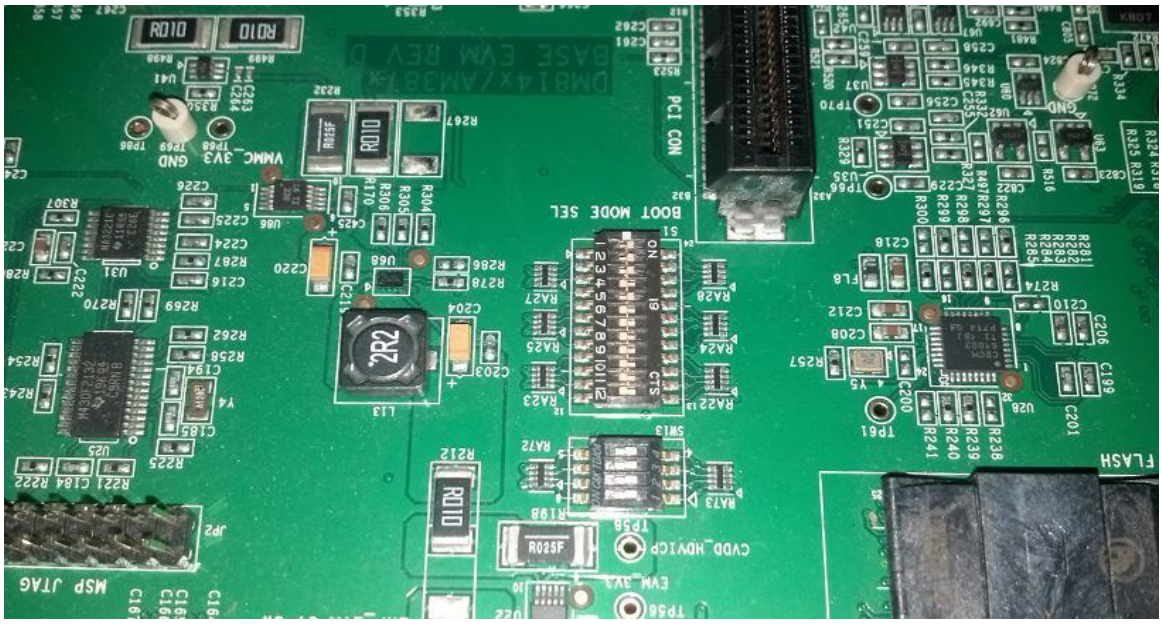
For TI814x:

1. Copy the MLO file in MMCSD card. Insert the card in the MMC/SD slot.
2. Boot the board in SD Boot Mode. The switch S1 settings needs to be as below:
   DIP_SW1 - DIP_SW3 = ON

   DIP_SW4 = OFF

   DIP_SW5 = ON

   DIP_SW6 - DIP_SW12 = OFF



3. Connect the EVM to CCS through JTag.
4. Connect to specific core (Cortex a8 or VideoM3 or DSP C674x).
5. Load the binary.
6. After loading binary on Cortex A8 core run the gel file script “SetCPSRsuper” from the gel file provided at location “$(starterware)/tools/gel/switch_cpu_mode.gel” (Not required for running on VideoM3) once you run to symbol main.
7. Run the application

For TDA2xx:

1. Copy the MLO file in MMCSD card. Insert the card in the MMC/SD slot.
2. Boot the board in SD Boot Mode.
3. Connect the EVM to CCS through JTag.
4. Connect to specific core (Cortex a15/M4/C66x).
5. Load the binary.
6. Run the application

For TDA2Ex:

7. Copy the MLO file in MMCSD card. Insert the card in the MMC/SD slot.
8. Boot the board in SD Boot Mode.

9. Connect the EVM to CCS through JTag.
10. Connect to specific core (Cortex a15/M4/C66x).
11. Load the binary.
12. Run the application

For TDA3xx:

1. Flash the SBL image on QSPI flash using QSPI flash writer.
2. Boot the board in QSPI Boot Mode.
3. Connect the EVM to CCS through JTag.
4. Connect to specific core (M4/C66x).
5. Load the binary.
6. Run the application

NOTE: Many StarterWare examples print messages on the UART Serial Console running on the host. Hence, a serial terminal application (like Tera Term/HyperTerminal/minicom) should be running on the host. The host serial port is configured at 115200 baud, Data Bits 8, no parity, 1 stop bit and no flow control. Please ensure that the local echo setting for the terminal is turned off. The serial port (DB9 connector P2) on the baseboard of the EVM is to be connected to the host serial port via a NULL modem cable.

7.4. EDMA Test

This example is in <install_path>/examples/edma_test

The EDMA Test example runs from M3VPSS, 674 and CORTEX A8 core of TI814x, M4, Cortex A15 core of TDA2xx and TDA2Ex and M4 of TDA3xx. Following are the test cases executed as explained below. To run on M4 Core of TDA2xx You need to do the AMMU configuration to access the edma registers. For this load the gel file <install_path>/tools/gel/VayuIPC.gel and run the gel function program_IPU_AMMU_FOR_EDMA from the scripts.

DMA_TEST: This test case transfers 64 bytes of data from _srcBuff1 to _dstBuff1.

DMA_CHAIN_TEST: This test case transfers 64 bytes of data from _srcBuff1 to _dstBuff1 and with the chaining feature enabled it triggers another 64 bytes of transfer from _srcBuff2 to _dstBuff2.

DMA_LINK_TEST: This test case transfers 64 bytes of data from _srcBuff1 to _dstBuff1 and with the linking feature it loads the second set of EDMA parameter set and on second trigger it transfers another 64 bytes from _srcBuff2 to _dstBuff2 after this on one more trigger it performs third set of transfer depending on the third set of EDMA parameter set.

QDMA_TEST: This test case transfers 64 bytes of data from _srcBuff1 to _dstBuff1 using a QDMA channel.

QDMA_LINK_TEST: This test case transfers 64 bytes of data from _srcBuff1 to _dstBuff1 and with the linking feature it loads the second set of EDMA parameter set. On QDMA channel when written to the trigger word the transfer starts. Hence while copying the second set of EDMA parameter set, the trigger word is also written hence triggers a transfer on second channel. The transfer on second channel transfers 64 bytes from _srcBuff2 to _dstBuff2.

DMA_POLLED_TEST: This test case transfers 64 bytes of data from _srcBuff1 to _dstBuff1 but the interrupt is not enabled hence we poll for the IPR bit to be set after triggering the transfer.

DMA_PING_PONG_TEST: This is test to show a real world scenario where the EDMA will be transferring first buffer while the CPU can processes second buffer, once these are done CPU can work on first buffer while EDMA can transfer on Second buffer. There are two big buffers of size (PING_PONG_NUM_COLUMNS * PING_PONG_NUM_ROWS).Both are present in DDR and are known as pingpongSrcBuf and pingpongDestBuf. There are two small buffers of size (PING_PONG_L1D_BUFFER_SIZE). They are known as ping buffer and pong buffer. The pingpongSrcBuf is divided into chunks, each having size of PING_PONG_L1D_BUFFER_SIZE. Data is being transferred from pingpongSrcBuf to either ping or pong buffers, using EDMA3. Logic behind using two ping pong buffers is that one can be processed by DSP while the other is used by EDMA3 for data movement. So ping and pong are alternately used by EDMA3 and DSP. Also, to simulate the real world scenario, as a part of DSP processing, I am copying data from ping/pong buffers to pingpongDestBuf. In the end, I compare pingpongSrcBuf and pingpongDestBuf to check whether the algorithm has worked fine.

This example is tested on TDA2xx and TDA2Ex EVM PG1.0 on Cortex A15 and IPU core and TDA3xx EVM PG1.0 on IPU core.

7.5. GPIO Output Test

This example is in <install_path>/examples/gpio/gpio_output

The GPIO Test example is a simple test running from a15 core of tda2xx and tda2ex EVM and M4 core of tda3xx EVM to show how to toggle a GPIO pin from a CPU. Below are the pins to be used for probing. These pins can be probed to get a square wave when the app is running continuously. This example does not use any interrupt**.**

On TDA2xx and TDA2Ex EVM GPIO1 IP's pin 14 is used. On the EVM this comes out on a DCAN1 TX pin with a mux mode of DCAN1 pins.

OnTDA3xx EVM GPIO4 IP's pin 9 is used. On the EVM this comes out on a DCAN1 TX pin with a mux mode of DCAN1 pins.

NOTE: Due to EVM limitation examples showing generation of interrupt to the core from an external GPIO toggle cannot be created for TDA1Mxx release. Also there is no visible indication of GPIO write on the EVM.

7.6. GPIO Input Interrupt Test

This example is in <install_path>/examples/gpio/gpio_input_interrupt

The GPIO Test example is a simple test running from a15 core of tda2xx EVM and m4 core 0f tda3xx EVM to show how to read from GPIO pin from a CPU and to demonstrate the GPIO interrupt usage when the input state changes. The application provides user interface through Uart console to select the type of interrupt change. The application provides user to configure GPIO, to detect four types of input state change (LOW, HIGH, RISING EDGE and FALLING EDGE). Below are the pins to be used in the example. A15 core prints the status of the pin on the Uart depending on the status of the pin and the type, GPIO is configured to detect input.

On TDA2xx EVM GPIO1 IP's pin 15 is used. On the EVM this pin comes out on a DCAN1 Rx pin with a mux mode of DCAN1 pins.

On TDA3xx EVM GPIO1 IP's pin 10 is used. On the EVM this pin comes out on a DCAN1 Rx pin with a mux mode of DCAN1 pins.

7.7. I2C Driver LED Test

This example is in <install_path>/examples/i2c/i2c_driver_led

The I2C LED test is an example running either from M3VIDEO, CORTEXA8, CORTEXA15 or CORTEXM4 core to write to an I2C slave device on the EVM and getting a visible indication by LEDs going on and off. Below are the I2C slave devices connected to 4 LEDs called USER_LED1/2/3/4. When a value of 0x00 and 0xF0 is written to this device over I2C0 the LEDs glow on to off. This example is implemented using I2C driver.

On TI814x, TDA2xx and TDA2Ex EVM there is a PCF8575 Remote 16 bit I2C Expander (U14) for TI814x and (U58) for tda2xx and TDA2Ex EVM PG1.0, connected to LEDs.

On TDA3xx there is a TCA6424 24 bit Expander (U8006) connected to LEDs.

7.8. A15 MMU Data Validation Test

This example is in <install_path>/examples/mmu/a15

The mmu data validation test is an example that runs on A15 core and demonstrates A15 Cache and MMU operations. This application is build and tested on TDA2xx platform.

The application first enables the L1 and L2 cache for A15. Then the application does the MMU configuration and writes data at the virtual address 0xD0000000. After this the MMU is disabled and the data is read from the physical address 0x90000000 and the data integrity is checked. Application prints a success if the data matches and failure otherwise.

7.9. Mailbox Test

This example is in <install_path>/examples/mailbox/mailbox_a15

The mailbox test is an example running from A8, M3VPSS and DSP cores for TI814x and A15, M4 and DSP cores to show a mailbox communication between the multiple cores. This example will use the SYSTEM Mailbox's Queue 0 for TI814x and Mailbox2 instance 1 for TDA2xx, TDA2Ex and TDA3xx EVM. This example works in two modes: interrupt mode and polled mode.

In each of the platform one core will run as receiver and all other cores run as sender. The receiver will initialize the mailbox and waits for the new message in polled mode or gets an interrupt when new message is sent. The sender will waits for the queue not full interrupt and sends a message to mailbox. When sender sends the message the receiver receives from the mailbox and prints on UART console.

On TI814x, M3VPSS and DSP will run sender app and A8 core will receiver app.

On TDA2xx and TDA2Ex, DSP, M4 run sender app and A15 core will run receiver app.

On TDA3xx, DSP run sender app and M4 core will run receiver app.

7.10. MMU Test

This example is in <install_path>/examples/mmu

There are two sets of example as explained below.

MMU TLB and TWL test: The MMU example runs from DSP core. In this example DSP writes the pattern data at physical address 0xB0000000 and 0x82000000 and then configures MMU TLB (Translation Look aside Buffer) and MMU TWL(Table walk through logic) to map these physical addresses to virtual addresses 0x81000000 and 0xA0000000 as preserved entries to protect them against global flush. Then DSP then does the global flush and reads the data from virtual address. If the pattern matches as written to physical address then the MMU is configured successfully and entries are protected against global flush. On success the message is printed on CCS console. If the pattern does not match test failure message is displayed on the CCS console. This example is tested on TDA2xx EVM PG1.0, TDA2Ex EVM and TDA3xx EVM PG1.0.

MMU translation fault handle: The MMU example runs from DSP core and A15 (on TDA2xx/TDA2Ex) / M4 (on TDA3xx) and requires A15 (on TDA2xx /TDA2Ex) / M4 (on TDA3xx) binary to be running before running DSP binary. This example demonstrates the case of handling translation fault. Here DSP configures MMU for required memory mapping and tries to access the memory which is not configured either in TLB or TWL. This results in translation fault sending an interrupt to cortex A15 (on TDA2xx/TDA2Ex) / M4 (on TDA3xx) and DSP core halts. Cortex A15(on TDA2xx/TDA2Ex) / M4 (on TDA3xx) on receiving the interrupt resets the DSP core and bring DSP core out of reset. DSP core then prints the test successful message on the CCS console. This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM PG1.0.

7.11. OCMC Test

This example is in <install_path>/examples/ocmc

There are six sets of tests as explained below. In the application a frame of 48 bytes is divided into 3 sub frames each of size 16 bytes and CBUF is configured with the size of sub frame. The size of virtual buffer is set to 48 bytes (Full frame). These tests are tested on TDA2xx, TDA2Ex and TDA3xx EVM PG1.0.

OCMC basic test: This test runs from M4 and CORTEXA15 (on TDA2xx & TDA2Ex) / DSP (on TDA3xx). M4 gets the virtual address pointer and writes a sub frame and sends a message to CORTEXA15 (on TDA2xx & TDA2Ex) / DSP (on TDA3xx) using mailbox indicating the sub frame write completion and keeps waiting till it gets a read completion message from CORTEXA15 (on TDA2xx & TDA2Ex) / DSP (on TDA3xx). CORTEX-A15 (on TDA2xx & TDA2Ex) / DSP (on TDA3xx) then reads the sub frame and sends a message to M4 with last word read as the message. The same continues for other two sub frames and the M4 prints the test pass or failure message on the UART terminal. This test also gives interface for the user using UART to configure OCMC ECC mode.

OCMC address sequence error test: This test runs from M4 core and demonstrates read and write address sequence error case. In this example write and read access are not done in positive raster scan order (i.e., address should increase except for frame start condition) leading to read and write address sequence error. When error occurs an interrupt is generated and error status is cleared in the ISR call. On success a success message is printed on the UART terminal. This application also gives interface for the user using UART to configure OCMC ECC mode.

OCMC overflow mid test: This test runs from M4 core and demonstrates overflow mid error case and handling. An overflow mid interrupt is generated when the difference between write and read pointer is greater than CBUF size with write pointer being ahead of read pointer. On interrupt, error status is cleared and read is performed to catch up with write pointer in the ISR call. On success a success message is printed on the UART terminal. This application also gives interface for the user using UART to configure OCMC ECC mode.

OCMC overflow wrap test: This test runs from M4 core and demonstrates overflow wrap error case and handling. An overflow wrap interrupt is generated when a new frame write event occurs with reader still reading the previous frame and the difference between write and read pointer is greater than CBUF size. In this case writing is blocked until a new frame start write and read event occurs in this order. On interrupt, error status is cleared and

reading of the previous frame is completed in the ISR call. The overwritten frame is corrupted and on next new frame start write and read event a valid frame will be available. On success a success message is printed on the UART terminal. This application also gives interface for the user using UART to configure OCMC ECC mode.

OCMC underflow test: This test runs from M4 core and demonstrates underflow error case and handling. An underflow interrupt is generated when reader goes ahead of writer. On interrupt, error status is cleared and write is performed to catch up with read pointer in the ISR call. On success a success message is printed on the UART terminal. This application also gives interface for the user using UART to configure OCMC ECC mode.

OCMC short frame test: This test runs from M4 core and demonstrates short frame error case. A short frame interrupt is generated when a new frame write event occurs with write pointer not having written the complete previous frame. On interrupt, error status is cleared in the ISR call. On success a success message is printed on the UART terminal. This application also gives interface for the user using UART to configure OCMC ECC mode.

7.12. QSPI Test

This example is in <install_path>/examples/qspi_test

This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM PG 1.0. This example reads the QSPI flash device ID and Manufacturer ID and prints on UART and then erases the flash and writes a pattern in the QSPI flash memory, and reads back the data in memory mapped mode and checks if the patterns match. Repeat the test with different clock frequency and different read commands. Supported combination are 12 MHz 4 pin Normal read, 48 MHz 4 pin Fast read, 64 MHz 4 pin Fast read, 64 MHz 4 pin Dual read, 64 MHz 6 pin QUAD read.

7.13. McSPI Flash Test

This example is in <install_path>/examples/mcspi

This example builds binary for both A8 and M3Vpss cores. On execution of the example, it writes 256 bytes to spi flash memory, reads it back and verifies the read data.To make the chip select to reach the spi flash, switch SW2 (NAND/SPI boot) needs to be switched ON.

SW2 (NAND/SPI Boot)	
1	2
•	•
ON	

The example works in DMA mode.

7.14. McSPI master/slave Test (EVM to EVM)

This example is in <install_path>/examples/ mcspiMasterSlave/master and <install_path>/examples/ mcspiMasterSlave/slave. This example will generate two binaries, one of them is for master and other one is for slave. Binary will be generated for both A8 and M3vpss cores.

The two boards have to be connected and connections are as shown below:

EVM 1 Connector	EVM 2 - Connector
SPI1_SCLK (Pin 35 of J17)	SPI1_SCLK (Pin 35 of J17)
SPI1_MISO (Pin 37 of J17)	SPI1_MOSI (Pin 41 of J17)
SPI1_MOSI (Pin 41 of J17)	SPI1_MISO (Pin 37 of J17)
SPI1_nCS0 (Pin 43 of J17)	SPI1_nCS0 (Pin 43 of J17)

Note: Make sure there is a common GND connection between the EVMs.

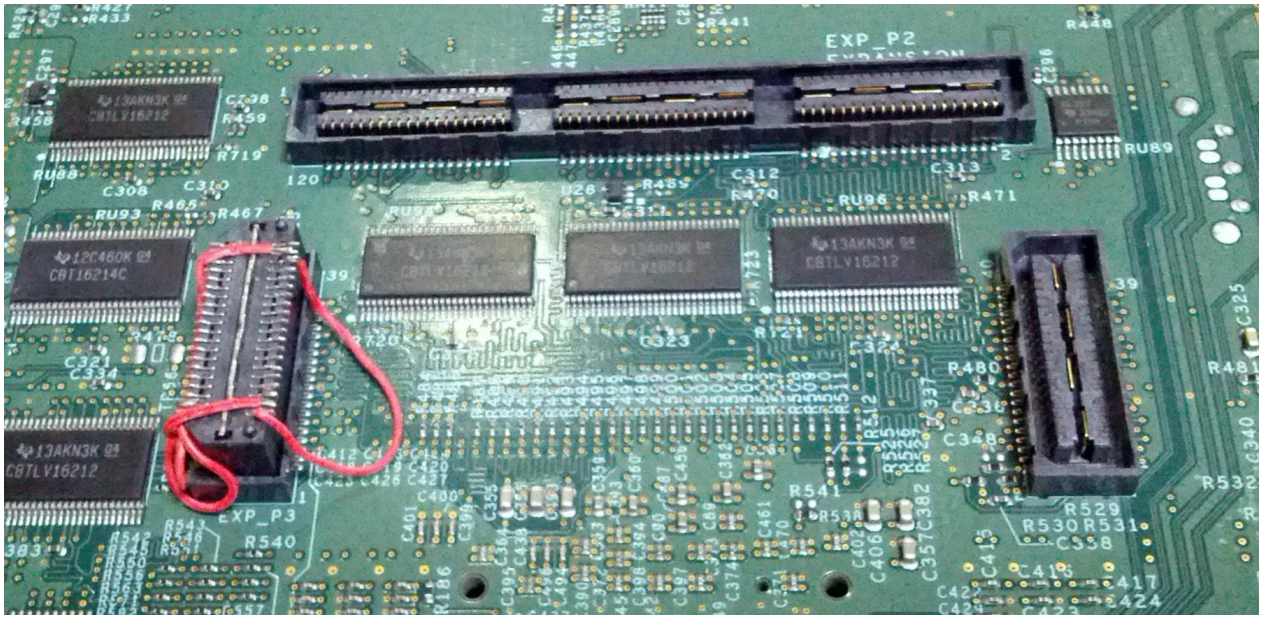
The application configures the Dat0 and Dat1 line as per the connection shown above. One of the EVM should be the master and other one slave. Load the binaries appropriately and execute the slave first. The slave will be waiting for data, start the master application. Once after receiving the data, slave will verify it and then displays the result on the CCS console.

This application works in DMA mode. This application is not validated on the TDA2xx EVM.

7.15. McSPI master/slave Test (With in EVM SPI1 to SPI2)

This example is in <install_path>/examples/ mcspiMasterSlave/ masterslave

This example will run on a15 core on TDA2xx, TDA2Ex and on M4 core on TDA3xx. Connect the McSPI1 and McSPI2 pins. Connect pins of visibility connector as shown below:



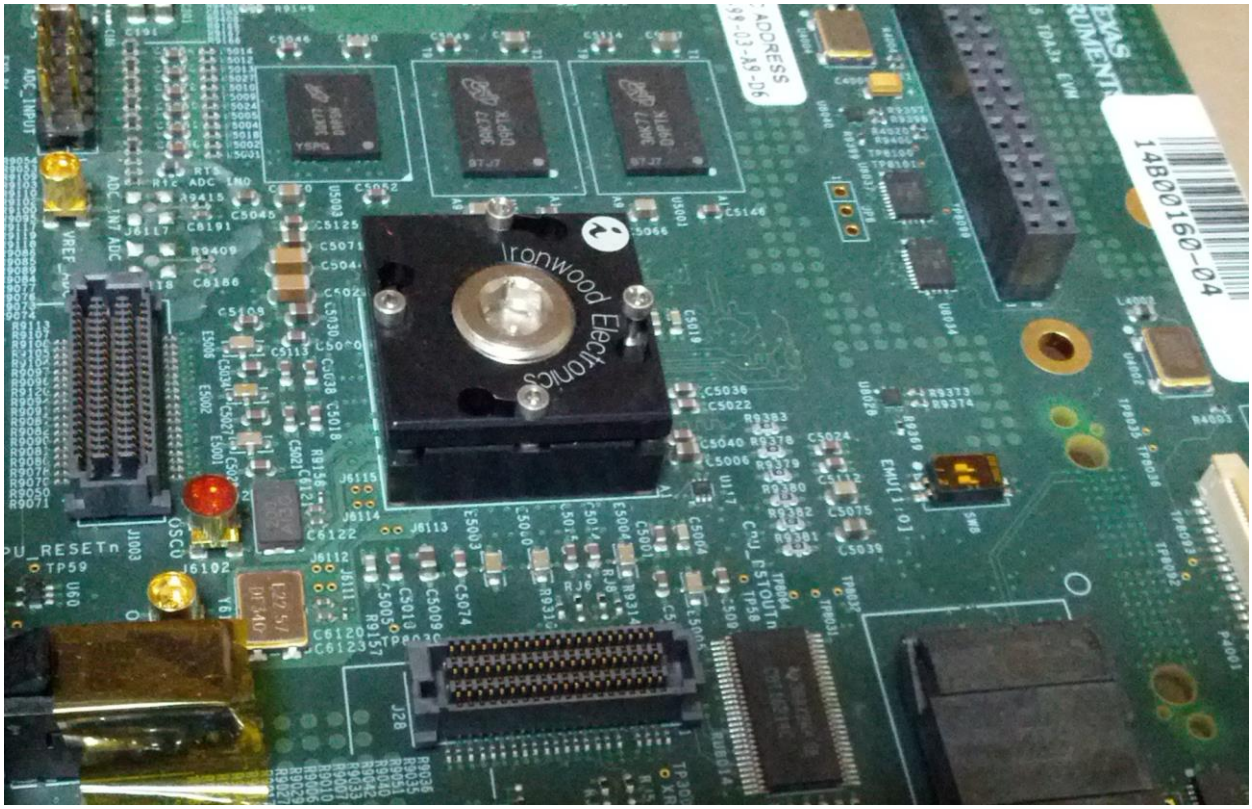
For TDA2xx and TDA2Ex please find the images below for connections. SPI pins to be connected to Daughter card connector EXP_P3 present at bottom side of the EVM.

SPI1	SPI2
SPI1 CS0 (Pin 36)	SPI2 CS0 (Pin 2)
SPI1 SCLK (Pin 35)	SPI2 SCLK (Pin 1)
SPI1 MOSI (Pin 38)	SPI2 MISO (Pin 3)
SPI1 MISO (Pin 37)	SPI2 MOSI (Pin 4)

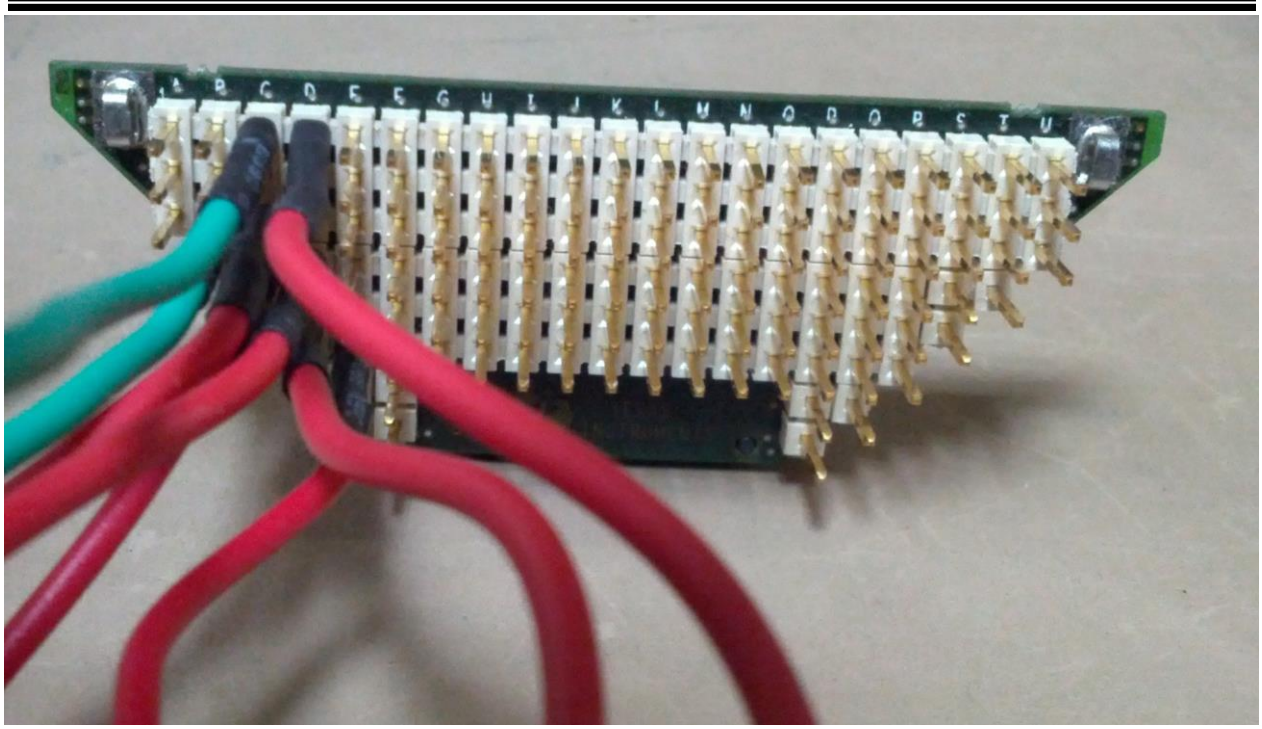
For TDA3xx please find the images below for connections. SPI pins to be connected to Visibility Connector J28 present beside SOC.

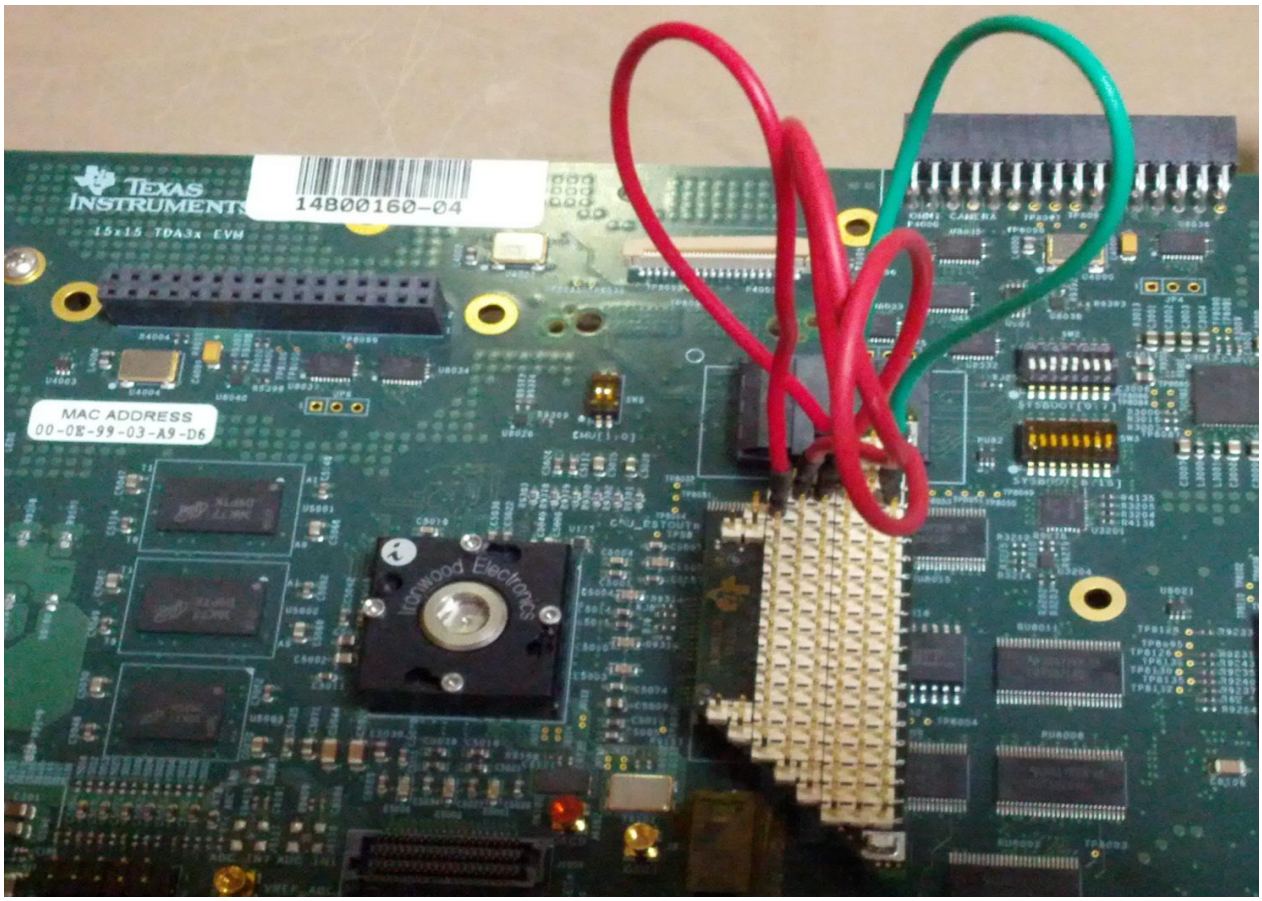
Please note that as UART3 pin is muxed with SPI1 SCLK for TDA3xx, UART1 terminal must be connected to give user input.

SP1	SPI2
SPI1 CS0 (Pin D4)	SPI2 CS0 (Pin D1)
SPI1 SCLK (Pin D5)	SPI2 SCLK (Pin C3)
SPI1 MOSI (Pin B3)	SPI2 MISO (Pin C1)
SPI1 MISO (Pin C4)	SPI2 MOSI (Pin E6)



Pins to be connected:





In this example data is transferred between spi1 to spi2 through external wires connecting McSPI pins.

7.16. NOR Flash Writer (GPMC Test)

This example is in <install_path> tools\flashtools\nor_flash_writer

This is a more complex example which shows how to write to a NOR flash through the GPMC IP. For this example to work a Vision Daughter Card needs to be connected to the Base EVM. The NOR part on the Vision Daughter Card is a Spansion NOR S29GL01GP90TFC02. The example has 2 logical parts which shows different parts of StarterWare being used.

1. First we configure the NOR Flash via GPMC.
2. Then we execute commands based on user input i.e. Erase Regions, Erase Whole Flash etc. and burn the flash image to NOR.

For TDA2xx and TDA2Ex, NOR Flash writer details are mentioned in section 5.4 NOR Boot Mode of TDA1Mxx_TDA2xx_SBL_UserGuide in Bootloader.

7.17. DSS Display Example

This example is in *<install_path>/examples/DssApp*. This is supported on **m4vpss** core of the **tda2xx, tda2ex and tda3xx** platform.
This can be built using the command "*gmake -s DssApp PLATFORM=tda2xx*" for tda2xx, "*gmake -s DssApp PLATFORM=tda2ex*" for TDA2Ex and "*gmake -s DssApp PLATFORM=tda3xx*" for tda3xx

This example show how the DSS starterware API as exposed in *<install_path>/include/vps_dssDrv.h* can be used to display data using DSS module.

Tda2xx/Tda2ex/Tda3xx EVM – This example will display different test patterns on the LCD connected the base EVM.

Example overview:

1. All Display APIs (VpsDrv_xxxx functions) are called from the file:
   <install_path>/examples/DssApp/src/app_dssConfigure.c
   The open(), close(), start(), stop(), setParams() APIs need to be called multiple times if based on number of capture ports used.
2. DSS PRCM/PLL configuration is done in *<install_path>/examples/DssApp/src/app_clk_prcm.c*
3. Templates for display driver init(), open() and setParams() configurations are provided in *<install_path>/examples/DssApp/src/app_dssConfigure.c*
4. Frame completion callback function in the example is App_deQueBufs() in *<install_path>/examples/dssApp/src/main.c*

The DSS example depends on four libraries – *starterware_vpslib, starterware_common* i2clib and *sys_config*. These will be automatically built as part of the build command

7.18. VIP Capture Example

This example is in *<install_path>/examples/vipCapt*. This is supported on **m3vpss** core of the ti814x and m4vpss core of the **tda2xx, tda2ex** and **tda3xx** platform.
This can be built using the command "*gmake -s vipCapt PLATFORM=tda2xx*", "*gmake -s vipCapt PLATFORM=tda2ex*", "*gmake -s vipCapt PLATFORM=tda3xx*", and "*gmake -s vipCapt PLATFORM=ti814x*"

This example show how the VIP starterware API as exposed in *<install_path>/include/vps_vipDrv.h* can be used to capture data using VIP module. This example assumes that a video decoder is configured to provide discrete-sync data. VIP is configured to capture using HSYNC style line capture.

The example does not configure any decoder and, therefore, will not work on any EVM setup as is. Need to run sensor_config_app before running capture app.

Example overview:

1. All capture APIs (VpsDrv_xxxx functions) are called from the file:
   <install_path>/examples/vipCapt/src/app_vipConfigure.c
   The open(), close(), start(), stop(), setParams() APIs need to be called multiple times if based on number of capture ports used.

2. VIP PRCM/PLL configuration is done in *<install_path>/examples/vipCapt/src/app_clk_prcm.c*

3. Templates for capture driver init(), open() and setParams() configurations are provided in *<install_path>/examples/vipCapt/src/app_vipConfigure.c*

4. Frame completion callback function in the example is App_deQueBufs() in *<install_path>/examples/vipCapt/src/main.c*

5. Sub-frame callback function in the example is App_SubFrmCbFxn() in *<install_path>/examples/vipCapt/src/app_vipConfigure.c.*
   Please note that in the current example version, sub-frame callback is generated only on the N-th line of the input video for each frame. The line number is specified using *subFrmPrms[0].numLinesPerSubFrame* and interrupt mode is specified using *subFrmPrms[0].interruptMode*

6. The VIP example depends on three libraries – *starterware_vpslib, starterware_common* and *sys_config*. These will be automatically built as part of the build command.

7.19. Sensor Configuration Example

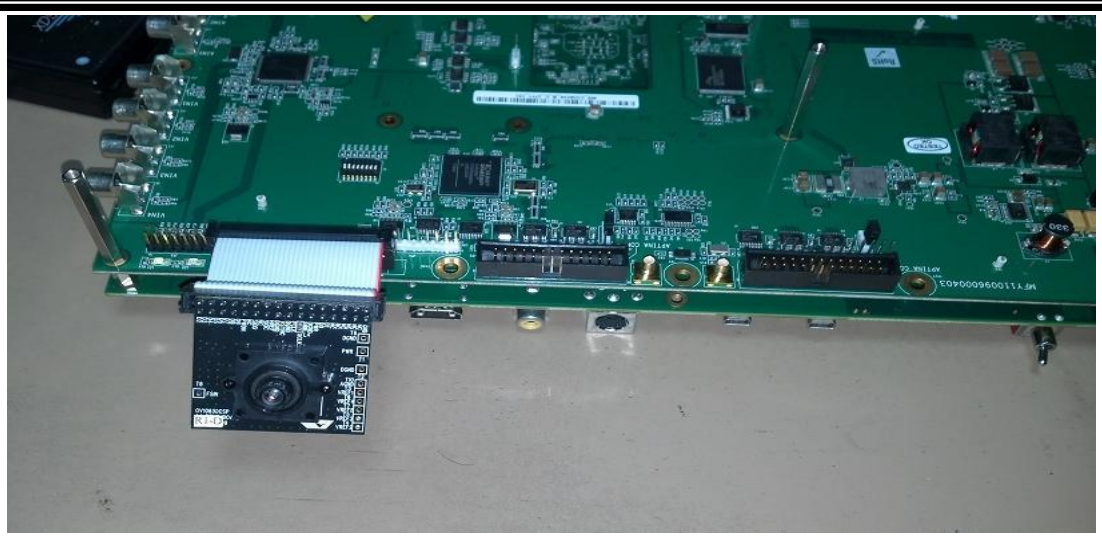
This example is in <install_path>/examples/ ov10630_sensor.

This application configures the OV10630 sensor for demonstrating YUV422 capture.

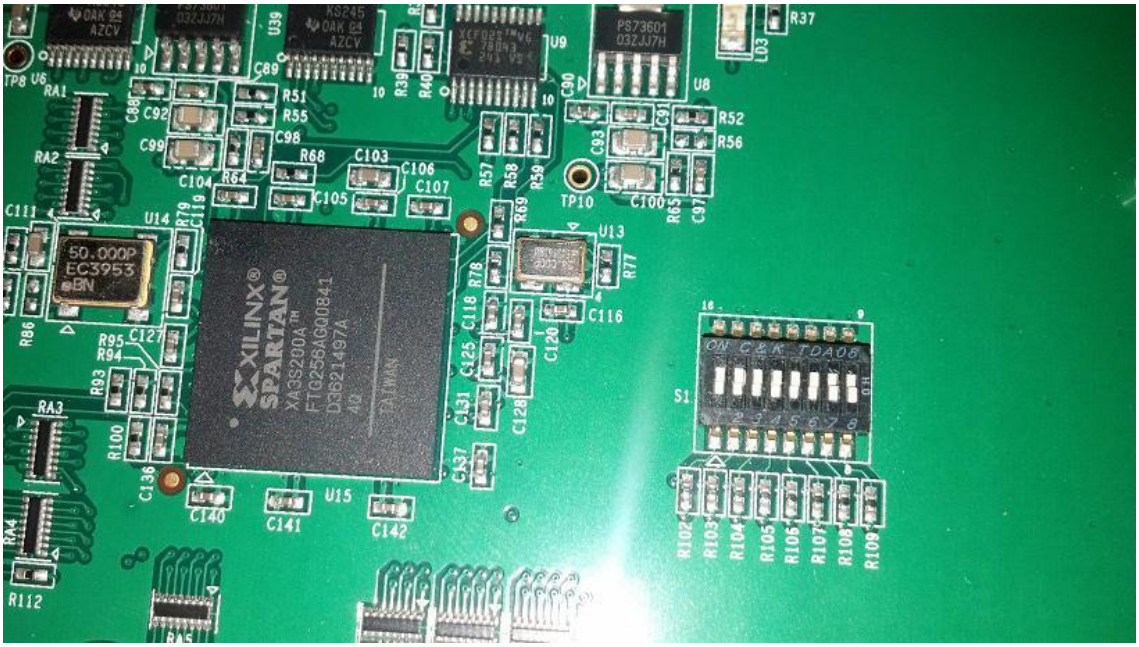
This example uses the I2C Lib module and INTC module to communicate to the sensor.

Following are the steps to run sensor_config_app on DM814x:

1. Requires TDA1Mxx or DM814x EVM connected to Video Vision Application Board (REV C).
2. OV10630 Sensor should be connected to OV7962 CONN port of Vision daughter card using 32/34 pin parallel IDE cable. First 32 pins of OV sensor need to be connected to this cable.



3. Configure switch S1 of the Video Vision Application Board as follows:
    a. DIP_SW1 - DIP_SW3 = X (Don't Care)
    b. DIP_SW4 - DIP_SW6 = ON
    c. DIP_SW7 - DIP_SW8 = X (Don't Care)



Following are the steps to run sensor_config_app on TDA2xx/TDA2Ex:

1. Requires TDA2xx/TDA2Ex base board to be connected to TDA2xx Vision Application board.
2. OV10630 Sensor should be connected to OMNIVISION CAMERA (P1) port of TDA2xx Vision Application board.
3. Configure switch S3 of the Video Vision Application Board as follows for TDA2xx.
    a. SW3 setting should be 01010101(PIN1 -> PIN8)

Following are the steps to run sensor_config_app on TDA3xx:

1. Requires TDA3xx base board.
2. OV10630 Sensor should be connected to OMNIVISION CAMERA port (P4000) of TDA3xx base board.

7.20. McASP Transmit Test

This example is in <install_path>/examples/mcasp/mcasp_transmit

The mcasp test is an example running on ti814x A8 and M3 core , tda2xx/tda2ex A15 and M4 core and tda3xx M4 Core.. This example demonstrates the mcasp as master to run at 10MHz bit clock frequency performing specific functional requirements. The mcasp outputs two different patterns on both the serializers. The output can be viewed on the CRO. Serializer 0 and Serializer 1 should yield the output frequency equal to (1/8th) and (1/4th) of Bit Clock respectively. These frequency ratios depend on the data patterns that are sent to particular serializers. In this case, for Serializer 0, data pattern is '0xFF' and data pattern for Serializer 1 is '0xCC'.

Following pins on the base board can be probed to view the output on the CRO

For tda2xx / tda2ex:

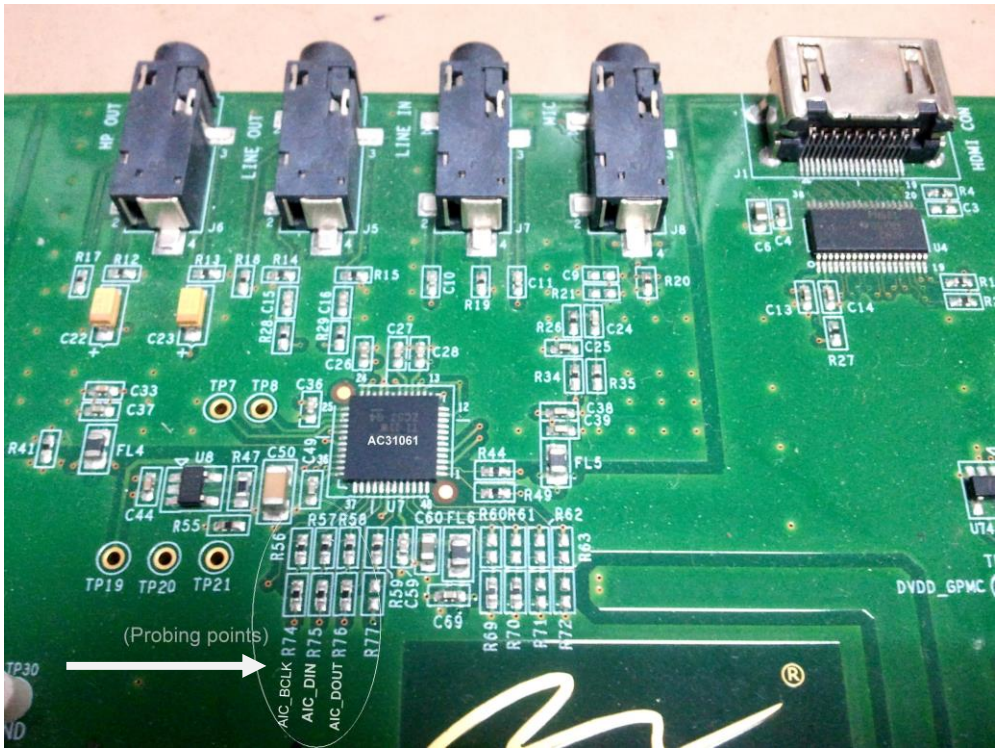
Probing Points	Pin	Pin Description
R253	AIC_BCLK	Bit Clock
R251	AIC_DIN	Serializer 0
R250	AIC_DOUT	Serializer 1
R252	AIC_WCLK	Frame Sync

For tda3xx:

Probing Points	Pin	Pin Description
R9118	AIC_BCLK	Bit Clock
R9093	AIC_DIN	Serializer 1
R9094	AIC_DOUT	Serializer 0
R9119	AIC_WCLK	Frame Sync

For ti814x:

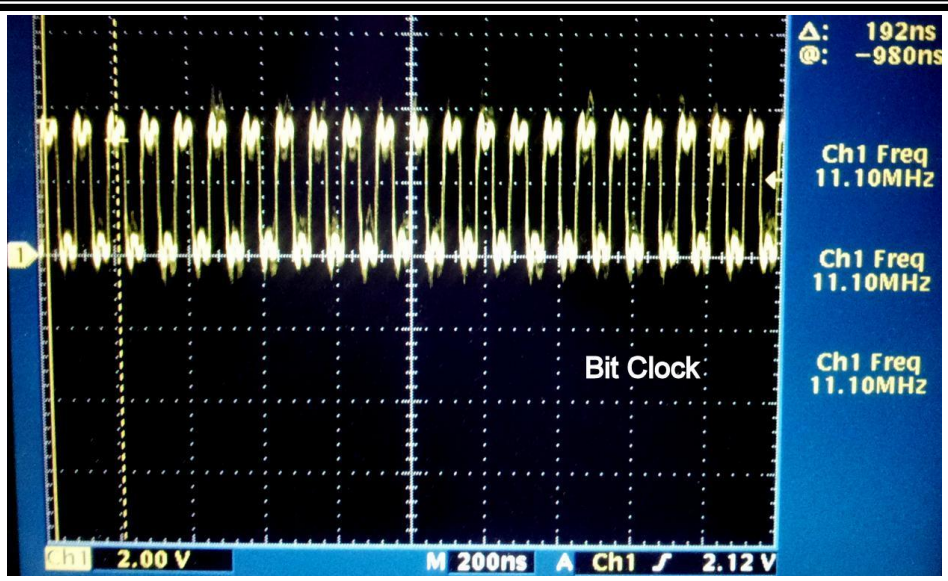
Probing Points	Pin	Pin Description
R74	AIC_BCLK	Bit Clock
R75	AIC_DIN	Serializer 1
R76	AIC_DOUT	Serializer 0
R57	AIC_WCLK	Frame Sync

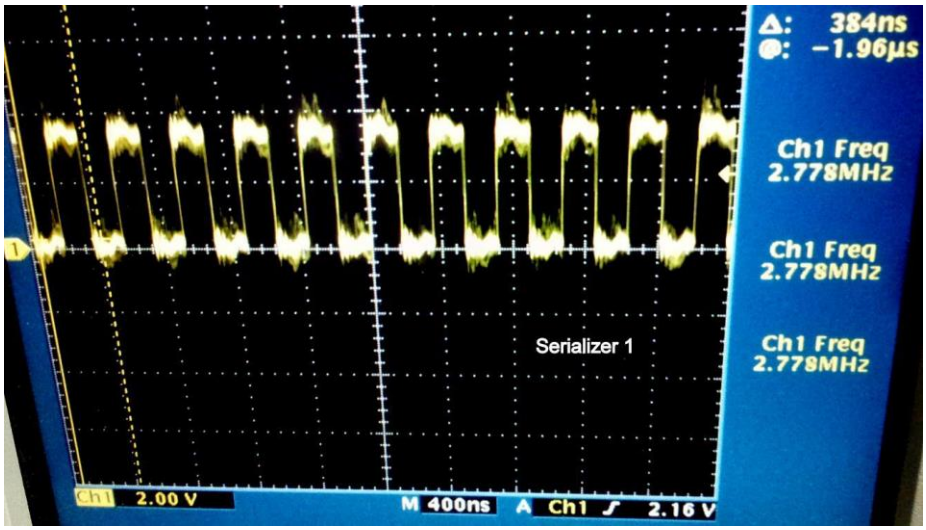


The example configures the Serializer 0 and Serializer 1 as output. The Bit Clock frequency can be achieved by setting appropriate divisor values in the example. Above figure shows probing points on ti814x-EVM.

This application works in DMA mode.

Waveforms –





This application is validated on the TI814x, TDA2xx, TDA2Ex and TDA3xx EVM.

7.21. McASP BurstModeTest

This example is in <install_path>/examples/mcasp/mcaspburst_transmit.

This Application demonstrates Burst Mode Support on A15, M4 using EDMA Manual Trigger Mode, Burst Transfer in Single Burst and no intermediate transfers, Synchronization using Interrupt and Polled Mechanism between EDMA Transfers and Buffers, HW Ping Pong Buffer Mechanism, Data Packets to be sent at different user configurable Delays, Update User Configurable Buffer Params.
We need to have probe configurations as mentioned in section 8.19 McASP Test, output needs to have data, frame sync pulses at varying delays. This is validated on A15 and M4 on tda2xx-EVM and tda2ex EVM.

7.22. SPINLOCK Test

This example is in <install_path>/examples/spinlock_test

This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM PG1.0. This test runs on CortexA15, DSP and M4 core for TDA2xx and TDA2Ex and runs on DSP and M4 core for TDA3xx. Here the core which is run first acquires the lock and other cores ran after this keep waiting for the lock to get release. Each core prints the status of the lock on the UART terminal

7.23. I2C EEPROM Test

This example is in <install_path>/examples/ i2c/i2c_eeprom_app

This app runs from Cortex A15 for TDA2XX and TDA2EX and from M4 for TDA3XX and is used test I2C in different configurations. I2C is used to read/write to EEPROM flash. This application writes 64 bytes of data to EEPROM starting from second page, reads 64 bytes of same data and compares whether the data written is same as data read.

In this example I2C is configured successively for all combinations of modes supported, bus speeds, FIFO status. Modes supported are Polled, DMA and Interrupt. Bus speeds supported are 100Kbps and 400 Kbps.

Note:

1. For tda2xx-EVM and tda2ex-EVM power ON bit 10 of userconfig switch (SW5) to provide write access to EEPROM.

2. For tda3xx-EVM power ON bit 2 of switch (SW8001) to provide write access to EEPROM.

7.24. UART Test

This example is in <install_path>/examples/ uart/uart_test

This app runs from Cortex A15 (on TDA2xx and TDA2Ex)/ M4 (on TDA3xx) and is used test UART working for different line characteristics configurations (baud rate, parity, stop bit and word length). Parity is set to “None” in the app. Please select parity as “None” while opening Terminal. UART is configured depending on the test case Id. User is asked to enter some data on UART console which is echoed back.

This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM PG1.0.

7.25. UART Interrupt Test

This example is in <install_path>/examples/ uart/uart_intr

This app runs from Cortex M4 and is used test UART working in the interrupt mode. UART will first send data to UART console using THR interrupt. Then the data entered by user will be echoed back using the RHR interrupt.

This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM PG1.0.

7.26. UART DMA Test

This example is in <install_path>/examples/ uart/uart_edma

This app runs from Cortex (on TDA2xx and TDA2Ex)/ M4 (on TDA3xx) and is used test UART working in the dma mode. The application echoes the characters that user types on the console. UART will send the data to Console using edma and ask user to enter 8 bytes of data. Then the data entered by user will be echoed back to the user using edma. This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM PG1.0.

7.27. Timer Test

This example is in <install_path>/examples/ timer

This app runs from Cortex A15, Cortex M4 & C66x on TDA2xx and TDA2Ex, runs from Cortex M4 & C66x on TDA3xx and runs from Cortex A8, Cortex M3 and C674x on TI814x.

On TDA2xx, TDA2Ex and TDA3xx this app is used to test Timer for different IRQ XBAR instances. The app will be run in a loop for particular core. Firstly the interrupt will be configured for first IRQ XBAR instance. Then Timer is configured for overflow interrupt. Timer will generate interrupt 10 times and count value will become zero. Then the interrupt will be configured for next IRQ XBAR instance and so on. On TI814x Timer is configured for overflow interrupt. Timer will generate interrupt 10 times and count value will become zero. Then success message is printed on UART console.

This example is tested on TDA2xx EVM PG1.0, TDA2Ex EVM PG1.0, TDA3xx EVM PG1.0 and TI814x-EVM.

7.28. WatchDog Timer Test

This example is in <install_path>/examples/ wdtimer

This app runs from Cortex A15 and is used test watchdog timer. This example will use the WDTimer instance 2 for TDA2xx. WDTimer setup is done for reset period of around 4s. In order to prevent reset user should continuously enter data from UART terminal with gap less than 4s. If no data is entered for reset period, the A15 core will be reset. This example is tested on TDA2xx and TDA2Ex EVM PG1.0.

7.29. Nor Edma Read Test

This example is in path <install_path>/examples\nor\nor_edma_read

This app runs on Cortex M4 Core of TDA2xx and TDA3xx and Cortex a8 core of TI814x. GPMC configuration is done to access the NOR flash. The NOR flash can be accesses in memory mapped mode. An EDMA transfer is setup to copy the content of NOR flash to a DDR location.

This example is tested on TDA2xx and TDA3xx EVM PG1.0.

7.30. PCIe Write Loopback Test

This example is in <install_path>/examples/ pcie/write_loopback

This app does board to board communication and runs from Cortex A15 on both boards. First both the boards wait to establish the link. When link gets established, RC will write data on EP side. EP will then read the data written and write it back to RC. RC will then compare the data which is written back with the data it wrote on EP. If the data matches then test is reported as success otherwise failure is reported.

This example is tested on TDA2xx EVM PG1.0 and TDA2Ex EVM

7.31. HDMI EDID Programmer

This example is in <install_path>/examples/ i2c_diag_test/edid_programmer

This app verifies and programs the EDID information to the EEPROM present in the vision application board. This is required by the HDMI receiver present in the Vision application board (on TDA2xx and TDA2Ex) / TDA3xx base board (on TDA3xx) to properly send EDID information to the HDMI source. For this application to program the EDID, the SW1 switch settings should be set to 11xx (1 to 4) for TDA2xx and TDA2Ex and SW80000 switch settings should be set to 11xx on TDA3xx base board. This will select I2C1 for programming and set the WP pin of the EEPROM to high so that new EDID could be programmed. After programming change the SW1 (on TDA2xx and TDA2Ex) / SW8000 (on TDA3xx) switch setting to original 00xx so that the EEPROM I2C is connected back to the HDMI I2C lines.

Note: The I2C address of the EDID EEPROM present in Vision daughter card (on TDA2xx and TDA2Ex) / TDA3xx Base Board (on TDA3xx) and the EEPROM present in the Multi-deserializer are the same (0x50). Hence while programming EDID, the Multi-deserializer board should be disconnected to avoid I2C conflict and incorrect EDID programming.

This example is tested on TDA2xx and TDA2Ex Vision EVM and TDA3xx EVM.

7.32. RTI Test

This example is in <install_path>/examples/rti

This app runs from Cortex M4 (on TDA3xx) and is used test RTI configured as DWWD. The application needs to be run twice to see full DWWD functionality. In first run, application programs RTI to generate reset to the system after specific time out period which is 10 sec. In second run, RTI is configured to generate reset after 10 sec but reset will not be generated as RTI is serviced before the timeout occurs. Application prints UART message when 10 sec are elapsed. However RTI will generate reset after 18 sec as it is not serviced again.

Note: Before loading app binary second time, disconnect and connect CortexM4_IPU1_C0 core. It will run Gel script under TDA3xx Misc Module configurations-> OnTargetConnect_API which is needed for second run.

This example is tested on TDA3xx EVM and should be run with gel.

7.33. CRC Test

This example is in <install_path>/examples/ crc

This app runs from Cortex M4 and C66x (on TDA3xx) and is used test CRC configured in semi-cpu mode. The application programs CRC to generate signature of pre-determined data pattern stored in memory. After generation of signature, application then compares it with pre-calculated signature value to check data integrity.

Note: The application uses 32-kHz Synchronized Timer (COUNTER_32K) for calculating performance. Hence while running application on C66x_DSP1 core, put CortexM4_IPU1_C0 core in free run mode.

This example is tested on TDA3xx EVM.

7.34. DCAN Loopback Test

This example is in <install_path>/examples/dcan/dcanLoopback

For Tda3xx, this example has three sets of tests: DCAN External Loopback Test, DCAN Internal Loopback Test and DCAN ECC Test and for Tda2xx and Tda2ex, this example has two tests- DCAN External Loopback Test and DCAN Internal Loopback Test. This app runs from Cortex M4 (on TDA3xx, Tda2ex & TDA2xx) and Cortex A15 (on TDA2xx)

DCAN External Loopback Test: This application tests DCAN external loopback test mode. DCAN external loopback test mode tests the internal feedback from Tx output to Rx input with external transceiver. Transmitted messages are treated as received messages and can be stored into message objects if they pass acceptance filtering. This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM.

DCAN Internal Loopback Test: This application tests DCAN internal loopback test mode. DCAN internal loopback test mode tests the internal feedback from Tx output to Rx input without external transceiver. Transmitted messages are treated as received messages and can be stored into message objects if they pass acceptance filtering. This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM.

DCAN ECC Test: This application tests DCAN ECC mode. In this mode, it enables Single Error Correction and Double Error Detection (SECDED) Mechanism. The application will corrupt single bit of the DCAN Message RAM data in RDA Test Mode and checks whether single bit error is detected and corrected. This verifies DCAN ECC Mode. This example is tested on TDA3xx EVM.

7.35. DCAN EVM Loopback Test(EVM to EVM)

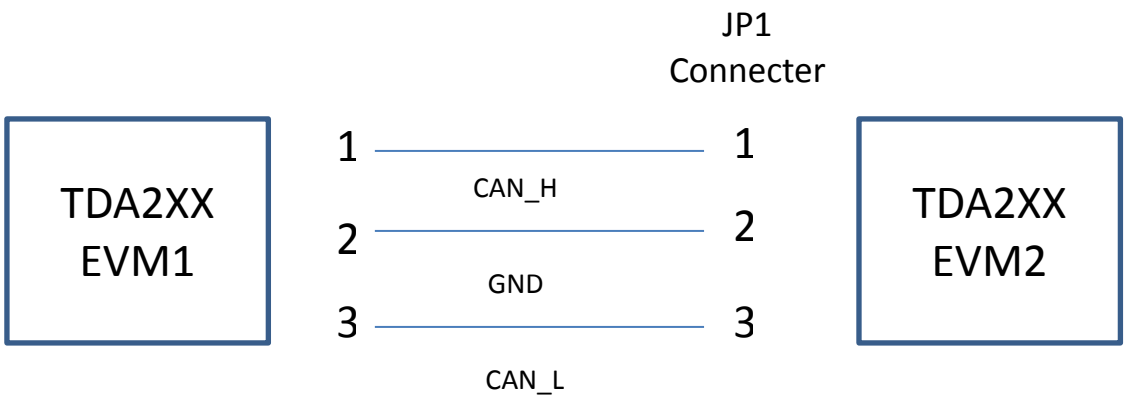
This example is in <install_path>/examples/dcan/dcanEvmLoopback

This application runs from Cortex M4 (on TDA3xx, TDA2Ex & TDA2xx) and Cortex A15 (on TDA2xx & TDA2Ex). This application tests DCAN external loopback mode. DCAN external loopback mode tests the board 1(Rx input) will receive the data transmitted from board 2 (Tx output) or vice versa. This application also prints the performance of the DCAN external loopback mode.

This example is tested on TDA2xx, TDA2Ex and TDA3xx EVM.

Make sure CAN1 Transreceiver (CAN1 Bus) of TDA2Ex/TDA2xx/TDA3XX Board1 is connected to CAN2 Transreceiver (CAN2 Bus) of TDA2Ex/TDA2xx/TDA3XX Board2 of the base EVM before running the example as shown in below diagram for Tda2xx

DCAN CABLE SETUP



For Tda3xx, connector is J6108, for Tda2ex connector is JP1 and for Tda2xx connector is JP2. For Tda2xx/Tda2ex, while running application on M4 core, put A15 core in free run mode.

One of the EVM should be the Transmitter and other one Receiver. Load the binaries appropriately and execute the Receiver first. The Receiver will be waiting for data, start the Transmitter application. Once after receiving the data, Receiver will verify it and then displays the result on the UART Console.

Following are the steps to run dcan_app_evm_loopback:

1. Load the generated Xem4 file in CCS connected to Tda2xx/Tda2ex/Tda3xx Board1 and Tda2xx/ Tda2ex/Tda3xx Board2
2. Run the application on Board1, Select an option 2 (DCAN_RX_TEST option), then
    a. Enter the expected number of messages to be received : 10000 (say 10000) then
    b. Enter the data length to be transmitted from 1 to 8 : 3 (say 3) then
    c. Enter the expected data sequence to be transmitted in Hex : (say)
        i. dataByte 0 : 0xFF
        ii. dataByte 1 : 0x44
        iii. dataByte 2 : 0x55 then
3. Run the application on Board 2, Select an option 1 (DCAN_TX_TEST option), then
    a. Enter the expected number of messages to be received : 10000 (say 10000) then
    b. Enter the data length to be transmitted from 1 to 8 : 3 (say 3) then
    c. Enter the data sequence to be transmitted in Hex : (say)
        i. dataByte 0 : 0xFF
        ii. dataByte 1 : 0x44
        iii. dataByte 2 : 0x55

7.36. ADC Test

This example is in <install_path>/examples/ adc_app

This app runs from Cortex M4 (on TDA3xx) and is used test ADC configured in single shot mode. The application programs ADC module to start conversion of input analog signals given through channels to digital. After complete conversion, application then prints converted digital values on the UART console. ADC module takes input form

channels numbered - 3, 4, 5 and 6. User should provide inputs for Analog to digital conversion through pins ADC_IN2, ADC_IN3, ADC_IN4 and ADC_IN5.

This example is tested on TDA3xx EVM.

7.37. DCC Test

This example is in <install_path>/examples/ dcc_app

This app runs from Cortex M4 (on TDA3xx) and is used test DCC configured in single shot mode. The application programs DCC module to use System Clock (SYS_CLK1) as a reference clock and DPLL_GMAC_H12 as a test clock. First, application configures DCC module for 5% allowed drift and generates DONE interrupt. After getting DONE interrupt, application changes the test clock frequency by programming respective DPLL while keeping rest of the configurations same, forcing test clock to drift more than 5%. This generates ERROR interrupt. This concludes the application and status is printed on the UART console.

This example is tested on TDA3xx EVM.

7.38. ESM Test

This example is in <install_path>/examples/ esm_app

This app runs from Cortex M4 (on TDA3xx) and is used test ESM configured in normal mode. The application programs ESM module to generate interrupt when EVE is given a reset. First, application configures ESM module for generation of interrupt to IPU core on detection of EVE CPU reset. Then it waits for EVE CPU reset to happen. It prints status of the application on the UART console. This example is tested on TDA3xx EVM.

Note: Before running the binary loaded on CortexM4_IPU1_C0, make sure to connect to the EVE core.

7.39. L3 FIREWALL (L3FW) Test

This example is in <install_path>/examples/ l3fw_app

This app runs from Cortex M4 (on TDA3xx and TDA2Ex) and Cortex A15 (on TDA2xx). The application programs L3FW module to generate interrupt/violation when target memory i.e. OCMC RAM is accessed from respective core/master for Write purposes. First, application configures L3FW module for generation of interrupt to IPU/MPU core on detection of violation. Then it waits for violation i.e. Write access to OCMC RAM to happen. It prints status of the application on the UART console.

This example is tested on TDA3xx, TDA2xx and TDA2Ex EVM.

7.40. ECC Test

This example is in <install_path>/examples/ecc_app

This example has three sets of tests: EMIF ECC Test, OCMC ECC Test and IPU ECC Test. This app runs from Cortex M4 (on TDA3xx, Tda2xx and Tda2ex) and from Cortex A15 (on Tda2xx and Tda2ex)

EMIF ECC Test: This application tests ECC feature of EMIF. ECC is enabled for EMIF defined address and tests if the EMIF data is corrupted for 1bit or 2bit EMIF data and then it checks for 1-bit and 2bit ECC errors are occurred or not. If I access non quanta aligned data or address and then it checks for Address ECC errors are occurred or not. This verifies EMIF ECC Feature.

OCMC ECC Test: This application tests ECC feature of OCMC. ECC is enabled for OCMC defined address and tests if the OCMC data is corrupted for 1bit or 2bit OCMC data and then it checks for 1-bit and 2bit ECC errors are occurred or not. If I corrupt ECC bits for the defined OCMC address and then it checks for Address ECC errors are occurred or not. This verifies OCMC ECC Feature.

IPU ECC Test: This application tests ECC feature of IPU L2RAM and L1Data. ECC is enabled for IPU L2RAM/ L1Data for defined address and tests if the data of IPU L2RAM/ L1Data is corrupted for 1bit or 2bit data and then it checks for 1-bit and 2bit ECC errors are occurred or not. This verifies IPU L2RAM and L1Data ECC Feature.

EMIF ECC Test and OCMC ECC Test are tested on TDA3xx, Tda2xx and Tda2ex EVM. IPU ECC Test is tested on TDA3xx EVM.

Note:

1. For EMIF ECC Test on Tda2xx, Before loading binary on Cortex A15 and Cortex M4 core run the gel file script "TDA2xx_set_lisa_maps_MEMMAP_512MB_SINGLE_EMIFX1" from the gel file provided at location "$starterware_rootdir/tools/gel/ECC_Cfg.gel"

2. For IPU ECC Test , Before loading binary on Cortex M4 core run the gel file script "AMMU_config_IPU_ECC_Test" from the gel file provided at location "$starterware_rootdir/tools/gel/ECC_Cfg.gel"

7.41. C66x XMC and MPU Test

This example is in <install_path>/examples/xmc_mpu_app

This app runs from C66x DSP1 on TDA3xx. This application sets up two sections each in DDR, OCMC and DSP L2RAM – one section is given read-write permissions for all CPU modes and other sections is given read-write permissions for supervisor mode and read-only for user mode. The application verifies whether the XMC and MPU modules are able to provide access permissions as required to different memory regions.

This example is tested on TDA3xx EVM.

8. Board Diagnostics

The examples provided in the board diagnostics section can be used to validate the basic connections and functionality of the hardware peripherals on the board. Section 8.1 gives a brief overview of the list of example test cases provided with the starterware and section 8.2 gives detailed description on how to run the example test case and the results expected.

8.1. List of Examples

Below is the list of examples provided for the board diagnostics test for TDA2xx

S. No.	Module	Exe Name	Path of the binary	Description
1.	PMIC Test	pmic_app_a15host_release.xa15fg	<install_path>/binary/ pmic_app/bin/tda2xx	Read the product Id and set power rail to 1.12v.
2.	DDR3 Test	ddr_test_app_a15host_release.xa15fg	<install_path>/binary/ddr_test_app/bin/tda2xx	Tests the DDR3 interface with A15 core. Does a memory read/write of data pattern and checks if the patterns match.
3.	GPIO Expander Test	gpio_test_app_a15host_release.xa15fg	<install_path>/binary/gpio_test_app/bin/tda2xx	Tests gpio expander on base board and JAMR3 board by reading expander register and writing back the same value.
4.	QSPI Flash Test	qspi_test_app_a15host_release.xa15fg	<install_path>/binary/ qspi_test_app/bin/tda2xx	Tests the QSPI flash interface. This test does a read/write of data pattern to QSPI flash memory.
5.	EEPROM Test	eeprom_app_a15host_release.xa15fg	<install_path>/binary/eeprom_app/bin/tda2xx	Tests eeprom interface on base board and JAMR3 board by writing one page of data to a block, read the same block and verifies the data written and read.
6.	UART3 Test	uart3_test_app_a15host_release.xa15fg	<install_path>/binary/ uart3_test_app/bin/tda2xx	Tests UART3 interface

7.	UART1 Test	uart1_test_app_a15host_release.xa15fg	<install_path>/binary/uart1_test_app/bin/tda2xx	Tests UART1 interface
8.	Temperature Sensor Test	temp_sensor_app_a15host_release.xa15fg	<install_path>/binary/temp_sensor_app/bin/tda2xx	Tests the temperature sensor interface by reading the temperature from the sensor.
9.	Boot-up Test	boot_app_a15host_release.xa15fg	<install_path>/binary/boot_app/bin/tda2xx	Tests the boot mode by reading the boot configuration switch in different boot modes.
10.	I2c all	i2c_test_app_a15host_release.xa15fg	<install_path>/binary/i2c_test_app/bin/tda2xx	Tests the i2c interface slave devices (IO expander, PMIC, EEPROM and temperature sensor) present on base board.
11.	LCD Test	DssApp_m4_release.xem4	<install_path>/binary/DssApp/bin/tda2xx	Tests the LCD interface
12.	SD Card Test	mmcsd_fileIO_app_a15host_release.xa15fg	<install_path>/binary/mmcsd_fileIO_app/bin/tda2xx	Tests the SD Card interface on base board and JAMR3 board. Does a card detect and uses FAT file system to read and write files to SD Card.
13.	NOR Read Write Test	nor_read_write_a15host_release.xa15fg	<install_path>/binary/nor_read_write/bin/tda2xx	Tests the NOR flash interface. This test does a read/write of data pattern to NOR flash memory.
14.	videoLoopback Test	videoLoopback_m4_release.xem4	<install_path>/binary/videoLoopback/bin/tda2xx	It configures the OV sensor then captures the images from the sensor and displays it on LCD

15.	McASP sinetone test	mcasp_sinetone_app_a15host_release.xa15fg on a15 and mcasp_sinetone_app_c66x_release.xe66 on DSP	<install_path>/binary/mcasp_sinetone_app/bin/tda2xx	Demonstrates sinetone generation using mcasp as master

Below is the list of examples provided for the board diagnostics test for TDA2Ex

S. No.	**Module**	**Exe Name**	**Path of the binary**	**Description**
1.	PMIC Test	pmic_app_a15host_release.xa15fg	<install_path>/binary/pmic_app/bin/tda2ex	Read the product Id and set power rail to 1.12v.
2.	DDR3 Test	ddr_test_app_a15host_release.xa15fg	<install_path>/binary/ddr_test_app/bin/tda2ex	Tests the DDR3 interface with A15 core. Does a memory read/write of data pattern and checks if the patterns match.
3.	QSPI Flash Test	qspi_test_app_a15host_release.xa15fg	<install_path>/binary/qspi_test_app/bin/tda2ex	Tests the QSPI flash interface. This test does a read/write of data pattern to QSPI flash memory.
4.	EEPROM Test	eeprom_app_a15host_release.xa15fg	<install_path>/binary/eeprom_app/bin/tda2ex	Tests eeprom interface on base board by writing one page of data to a block, read the same block and verifies the data written and read.
5.	UART1 Test	uart1_test_app_a15host_release.xa15fg	<install_path>/binary/uart1_test_app/bin/tda2ex	Tests UART1 interface
6.	Temperature Sensor Test	temp_sensor_app_a15host_release.xa15fg	<install_path>/binary/temp_sensor_app/bin/tda2ex	Tests the temperature sensor interface by reading the temperature from the sensor.

7.	LCD Test	DssApp_m4_release.xem4	<install_path>/binary/ DssApp/bin/tda2ex	Tests the LCD interface
8.	SD Card Test	mmcsd_fileIO_app_a15host_release.xa15fg	<install_path>/binary/mmcsd_fileIO_app/bin/tda2ex	Tests the SD Card interface on base board. Does a card detect and uses FAT file system to read and write files to SD Card.
9.	NOR Read Write Test	nor_read_write_a15host_release.xa15fg	<install_path>/binary/nor_read_write/bin/tda2ex	Tests the NOR flash interface. This test does a read/write of data pattern to NOR flash memory.
10.	videoLoopback Test	videoLoopback_m4_release.xem4	<install_path>/binary/videoLoopback/bin/tda2ex	It configures the OV sensor then captures the images from the sensor and displays it on LCD

Below is the list of examples provided for the board diagnostics test for TDA3xx

S. No.	**Module**	**Exe Name**	**Path of the binary**	**Description**
1.	PMIC Test	pmic_app_m4_release.xem4	<install_path>/binary/ pmic_app/bin/tda3xx	Read the product Id and set power rail to 1.12v.
2.	DDR3 Test	ddr_test_app_m4_release.xem4	<install_path>/binary/ddr_test_app/bin/tda3xx	Tests the DDR3 interface with M4 core. Does a memory read/write of data pattern and checks if the patterns match.
3.	QSPI Flash Test	qspi_test_app_m4_release.xem4	<install_path>/binary/ qspi_test_app/bin/tda3xx	Tests the QSPI flash interface. This test does a read/write of data pattern to QSPI flash memory.
4.	EEPROM Test	eeprom_app_m4_release.xem4	<install_path>/binary/eeprom_app/bin/tda3xx	Tests eeprom interface on base board by writing one page of data to a

				block, read the same block and verifies the data written and read.
5.	UART3 Test	uart3_test_app_m4_release.xem4	<install_path>/binary/ uart3_test_app/bin/tda3xx	Tests UART3 interface
6.	UART1 Test	uart1_test_app_m4_release.xem4	<install_path>/binary/ uart1_test_app/bin/tda3xx	Tests UART1 interface
7.	Temperature Sensor Test	temp_sensor_app_m4_release.xem4	<install_path>/binary/temp_ sensor_app/bin/tda3xx	Tests the temperature sensor interface by reading the temperature from the sensor.
8.	Boot-up Test	boot_app_m4_release.xem4	<install_path>/binary/boot_ app/bin/tda3xx	Tests the boot mode by reading the boot configuration switch in different boot modes.
9.	LCD Test	DssApp_m4_release.xem4	<install_path>/binary/ DssApp/bin/tda3xx	Tests the LCD interface
10.	SD Card Test	mmcsd_fileIO_app_m4_release.xem4	<install_path>/binary/mmcs d_fileIO_app/bin/tda3xx	Tests the SD Card interface on base board. Does a card detect and uses FAT file system to read and write files to SD Card.
11.	NOR Read Write Test	nor_read_write_m4_release.xem4	<install_path>/binary/nor_r ead_write/bin/tda3xx	Tests the NOR flash interface. This test does a read/write of data pattern to NOR flash memory.
12.	videoLoopback Test	videoLoopback_m4_release.xem4	<install_path>/binary/videoL oopback/bin/tda3xx	It configures the OV sensor then captures the images from the sensor and displays it on LCD

8.2. Description of the Examples

8.2.1. PMIC Test

This example is in <install_path>/examples/i2c_diag_test/pmic_app

The pmic(power management IC) test is an example running from CORTEXA15 core(on TDA2xx and TDA2Ex) and M4 core (on TDA3xx) to read the product id and manufacture id along with setting the power rail to 1.12v with the I2C slave device(TPS659039 for TDA2xx / TPS65917 for TDA3xx and TDA2Ex) on the EVM. On the EVM the TPS659039 (U45 for TDA2xx) and TPS65917 (for TDA3xx and TDA2Ex) power management IC is connected to i2c instance zero and has slave id 0x58 to set the power rails and 0x59 to read the product id and manufacture id.

To run pmic test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and TDA2Ex and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx.
3) Connect the UART terminal & launch Uart console.
4) Load the “pmic_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\pmic_app\bin\tda2xx (for tda2xx) / “pmic_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\pmic_app\bin\tda2ex (for TDA2Ex) / “pmic_app_m4_release.xem4” and present in “<starterware_rootdir>\binary\pmic_app\bin\tda3xx (for tda3xx) and execute it from CCS.
5) After the completion see the logs on the UART console which prints the product id and manufacture id, voltage to which the power rail is set and also the test success or failure message.

Please see the example log from the console for the PMIC test for TDA2xx

```
PMIC Data:
 VENDOR_ID_LSB: 0x51
 VENDOR_ID_MSB: 0x4
 PRODUCT_ID_LSB: 0x35
 PRODUCT_ID_MSB: 0xC0
Value 0x44 written to PMIC register 0x23 to set voltage to 1.12V
PMIC voltage set to 1.12V
PMIC test  successful
```

Please see the example log from the console for the PMIC test for TDA3xx and TDA2Ex

```
PMIC Data:
VENDOR_ID_LSB: 0x51
 VENDOR_ID_MSB: 0x4
 PRODUCT_ID_LSB: 0x17
 PRODUCT_ID_MSB: 0x9
Value 0x44 written to PMIC register 0x23 to set voltage to 1.12V
PMIC voltage set to 1.12V
PMIC test  successful
```

8.2.2. DDR3 Stress Test

This example is in <install_path>/examples/ddr_test

The DDR stress test example is a test running from a15 core of tda2xx and tda2ex EVM and M4 core of tda3xx EVM. This example runs four test cases to test the entire DDR.

1) The first test case does a full memory read write test. It writes a pattern to the memory reads back the data in memory and checks if the patterns match.

2) The second test case does a random memory read/write test. The user has to enter the starting address, number of times to generate the random memory and the number of times the test has to be repeated. It writes a pattern to the memory, reads back the data in memory and checks if the patterns match.

3) The third test case does a sequential memory read/write test where the user has to enter the starting address of the DDR memory, the size, the pattern to be tested and the number of times the test has to be repeated. It writes the pattern to the memory reads back the data in memory and checks if the patterns match.

4) The fourth test case does a sequential increment pattern memory read/write test where the user has to enter the starting address of the DDR memory, the size, the pattern to be tested and the number of times the test has to be repeated. It writes the incremented pattern to the memory reads back the data in memory and checks if the patterns match.

To run the DDR stress test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and TDA2Ex and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx .
3) Connect the UART terminal & launch Uart console.
4) Load the “ddr_test_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\ddr_test_app\bin\tda2xx for TDA2xx / “ddr_test_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\ddr_test_app\bin\tda2ex for TDA2Ex / “ddr_test_app_m4_release.xem4”, present in “<starterware_rootdir>\binary\ddr_test_app\bin\tda3xx for TDA3xx and execute it from CCS.
5) Enter the EMIF interface to test. Enter “1” for EMIF_1 interface or “2” for EMIF_2 interface for TDA2xx EVM.
6) Enter the size of the DDR memory to be tested in hex from the UART console. The size has to match the size of the DDR memory on the EVM.
7) After the completion of the first test case, similarly enter the relevant data for the second, third and fourth test case from the UART console as prompted on the console.
8) The example will finish testing the DDR and prints the appropriate message on the UART console if the test passes or fails.

Please see the example log from the console for the DDR test for TDA2xx

```
DDR Stress Test App

Enter the EMIF number to test:

Enter 1 for EMIF1 and 2 for EMIF2:1

*******************************************************

Running DDR test case 1-- Full memory read/write

Size of DDR is 0x1FFFFFFF

dst address 0x81FFF000

---------------------

----------------------

dst address 0x9FFF0000

DDR-- Full memory read/write test is complete

*************************************************************

Running DDR test case 2 -- Random memory read/write

How many times should the test be repeated(enter a number):1

Enter the start address of the memory to be tested in hex(without 0x):85000000

How many times random mem test is to be repeated(enter a number):10

Random DDR address 0x8377A000 size :0x1000Test count number 0x0

---------------------

Random DDR address 0x845C2000 size :0x1000

Test count number 0x1

DDR-- Random memory read/write test is complete

*************************************************************

Running DDR test case 3 -- Sequential memory read/write

How many times should the test be repeated(enter a number):2

Enter the start address of the memory to be tested in hex(without 0x):82000000

Enter the size of the DDR memory to be tested in hex(without 0x): 1100000
```

```
Enter the pattern used to be tested in hex(without 0x):aa

Test count number 0x0

Test count number 0x1

DDR-- Sequential memory read/write test is complete

************************************************************

Running DDR test case 4 -- Sequential increment pattern

How many times should the test be repeated(enter a number):2

Enter the start address of the memory to be tested in hex(without 0x):80000000

Enter the size of the DDR memory to be tested in hex(without 0x): 1100000

Enter the pattern used to be tested in hex(without 0x):11

Test count number 0x0

Test count number 0x1

DDR-- Sequential increment pattern test is complete

============================================================
```

Please see the example log from the console for the DDR test for TDA3xx

```
DDR Stress Test App

******************************************************

Running DDR test case 1-- Full memory read/write

Size of DDR is 0x1FFFFFFF

dst address 0x81FFF000

--------------------

----------------------

dst address 0x9FFF0000

DDR-- Full memory read/write test is complete

************************************************************

Running DDR test case 2 -- Random memory read/write

How many times should the test be repeated(enter a number):1
```

```
Enter the start address of the memory to be tested in hex(without 0x):85000000

How many times random mem test is to be repeated(enter a number):10

Random DDR address 0x8377A000 size :0x1000Test count number 0x0

---------------------

Random DDR address 0x845C2000 size :0x1000

Test count number 0x1

DDR-- Random memory read/write test is complete

************************************************************

Running DDR test case 3 -- Sequential memory read/write

How many times should the test be repeated(enter a number):2

Enter the start address of the memory to be tested in hex(without 0x):82000000

Enter the size of the DDR memory to be tested in hex(without 0x): 1100000

Enter the pattern used to be tested in hex(without 0x):aa

Test count number 0x0

Test count number 0x1

DDR-- Sequential memory read/write test is complete

************************************************************

Running DDR test case 4 -- Sequential increment pattern

How many times should the test be repeated(enter a number):2

Enter the start address of the memory to be tested in hex(without 0x):80000000

Enter the size of the DDR memory to be tested in hex(without 0x): 1100000

Enter the pattern used to be tested in hex(without 0x):11

Test count number 0x0

Test count number 0x1

DDR-- Sequential increment pattern test is complete

============================================================
```

8.2.3. GPIO Expander Test

This example is in <install_path>/examples/i2c_diag_test/i2c_gpio_expander

The gpio expander test is an example running from CORTEXA15 core to verify IO expander connection on base board and JAMR3 board. This test reads prints and writes back the data on IO expander. On the base board the PCF8575 gpio_expander (U119) is connected to i2c instance one with I2C slave address as 0x26 and on JAMR3 board PCF8575 gpio_expander (U14) is connected to i2c instance four with I2C slave address as 0x21. Application gives the UART interface to select the board to be tested.

To run gpio expander test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Connect base board to JAMR3 board.
3) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card.
4) Power OFF the bit one of switch SW2 to select I2C4 for accessing IO expander.
5) Connect the UART terminal & launch Uart console.
6) Load the "gpio_test_app_a15host_release.xa15fg" present in "<starterware_rootdir>\binary\gpio_test_app\bin\tda2xx and execute it from CCS.
7) On the UART console select the option to test the board. For the board selected slave address of the device probed, data read from the IO expander and test success or failure message is printed on the UART console.

Please see the example log from the console for the GPIO Expander test.

```
**** IO EXPANDER TEST ****
Menu:
1. Base  board I2C1 test
2. JAMR3 board I2C4 test
a. Test  All
x. Exit
Select test board : 1
Probing IO Expander with slave address 0x26 successful
Data read from IO expander: Port0 = 0xFF Port1 = 0xFF
IO expander test Successful

**** IO EXPANDER TEST ****
Menu:
1. Base  board I2C1 test
2. JAMR3 board I2C4 test
a. Test  All
x. Exit
Select test board : 2
Probing IO Expander with slave address 0x21 successful
Data read from IO expander: Port0 = 0xFC Port1 = 0xFF
IO expander test Successful

**** IO EXPANDER TEST ****
Menu:
1. Base  board I2C1 test
2. JAMR3 board I2C4 test
```

```
a. Test  All
x. Exit
Select test board : 3
Enter Valid option


**** IO EXPANDER TEST ****
Menu:
1. Base  board I2C1 test
2. JAMR3 board I2C4 test
a. Test  All
x. Exit
Select test board : a

IO Expander base board test...
Probing IO Expander with slave address 0x26 successful
Data read from IO expander: Port0 = 0xFF Port1 = 0xFF
IO expander test Successful

IO Expander JAMR3 board test...
Probing IO Expander with slave address 0x21 successful
Data read from IO expander: Port0 = 0xFC Port1 = 0xFF
IO expander test Successful

**** IO EXPANDER TEST ****
Menu:
1. Base  board I2C1 test
2. JAMR3 board I2C4 test
a. Test  All
x. Exit
Select test board : x
IO Expander board diagnostic test exiting...
```

8.2.4. QSPI Flash Test

This example is in <install_path>/examples/ qspi_test

The QSPI Flash test example is a test running from A15 core of tda2xx and tda2ex EVM and M4 core of tda3xx EVM. This example runs two test cases to test the QSPI Flash. This example first reads the QSPI flash device ID and Manufacturer ID and prints on UART console and then asks for the user input for quick flash test or extended flash test.

1. *Quick Flash test:* Erases one block of flash memory and writes a pattern in the QSPI flash memory, and reads back the data in memory mapped mode and checks if the patterns match.
2. *Extended Flash Test*: This test erases the entire flash memory and writes a pattern in the QSPI flash memory, and reads back the data in memory mapped mode and checks if the patterns match.

For testing the QSPI Flash, please set the EVM switch to the settings as given below for TDA2xx and TDA2Ex platform.

BootMode	EVM Switch Setting SYSBOOT(SW2)[1:16]	EVM Switch Setting SW5[1:10]
Debug	00000000 10000001	0001100000

To run the QSPI flash test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and TDA2Ex and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx .
3) Connect the UART terminal & launch Uart console.
4) Load the "qspi_test_app_a15host_release.xa15fg" present in "<starterware_rootdir>\binary\qspi_test_app\bin\tda2xx for TDA2xx / "qspi_test_app_a15host_release.xa15fg" present in "<starterware_rootdir>\binary\qspi_test_app\bin\tda2ex for TDA2Ex / "qspi_test_ app_m4_release.xem4" present in "<starterware_rootdir>\binary\qspi_test_app\bin\tda3xx for TDA3xx and execute it from CCS.
5) Enter the test case you want to run. Enter 1 for quick flash test or "2" extended flash test.
6) The example will finish testing the QSPI flash and prints the appropriate message on the UART console if the test passes or fails.

Please see the example log from the UART console for the QSPI flash test.

```
QSPI App

MID - 0x1

DID - 0x18

Enter 1 for quick flash test or 2 for extended flash test: 1

Erasing QSPI Flash Block

Erase Complete

Writing test pattern to flash..

Write flash Completed

Reading the flash data…

Data Read Write Test Passed.

Quick Flash Test Complete.

------------------------------------------------------------------------------

QSPI App

MID - 0x1

DID - 0x18

Enter 1 for quick flash test or 2 for extended flash test:  2

Erasing entire QSPI Flash. This takes 50-60 seconds
```

```
Erase Completed
Testing block no.0
Completed
Testing block no.1
Completed
Testing block no.2
Completed
Testing block no.3
Completed
Testing block no.4
Completed
……..
……..
……..
Testing block no.253
Completed
Testing block no.254
Completed
Testing block no.255
Completed
Entire Flash Test Complete
```

8.2.5. EEPROM Test

This example is in <install_path>/examples/i2c_diag_test/eeprom_i2c

The EEPROM test is an example running from CORTEXA15 core (on TDA2xx and TDA2Ex) and M4 core (on TDA3xx) to verify the EEPROM interface on base board and JAMR3 board for TDA2xx and on base board for TDA3xx and TDA2Ex. This test provides UART interface to select the board on which the EEPROM interface is to be tested. This test allows user to select the EEPROM block to be verified and also allows user to print the block data on the UART console. On base board EEPROM slave device (U105 of TDA2xx) has slave address of 0x50 (for TDA2xx) / 0x51 (for TDA3xx and TDA2Ex) and can be accessed through I2C instance zero. On JAMR3 board EEPROM slave device (U21 of TDA2xx) has slave address of 0x51 and can be accessed through I2C instance zero.

To run eeprom test example, please follow the steps below.

1) Open CCS & launch the target configuration.

2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and TDA2Rx and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx.
3) Power ON bit 10 of userconfig switch (SW5) to provide write access to EEPROM for TDA2xx and TDA2Ex and Power ON bit 2 of switch (SW8001) to provide write access to EEPROM for TDA3xx.
4) Connect the UART terminal & launch Uart console.
5) Load the "eeprom_app_a15host_release.xa15fg" present in "<starterware_rootdir>\binary\eeprom_app\bin\tda2xx for TDA2xx / "eeprom_app_a15host_release.xa15fg" present in "<starterware_rootdir>\binary\eeprom_app\bin\tda2ex for TDA2Ex / "eeprom_app_ app_m4_release.xem4" present in "<starterware_rootdir>\binary\eeprom_app\bin\tda3xx for TDA3xx and execute it from CCS.
6) On UART console select the board on which the EEPROM has to be tested and follow the UART instructions on the console. Success or failure message is printed on the console for the selected test case.

Please see the example log from the console for the eeprom test for TDA2xx.

```
**** EEPROM TEST ****
Menu:
1. Base  board test
2. JAMR3 board test
a. Test  All
x. Exit
Select test board : 1
Probing EEPROM with slave address 0x50 successfull

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : 1
Enter block to be tested : 1
Block Data: 0x2
0x3 0x4 0x5 0x6 0xFE 0x8 0x9 0xA
0xB 0xC 0xD 0xE 0xF 0x10 0x11 0x12
0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A
0x1B 0x1C 0x1D 0x1E 0x1F 0x20 0x21 0x22
0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A
0x2B 0x2C 0x2D 0x2E 0x2F 0x30 0x31 0x32
0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A
0x3B 0x3C 0x3D 0x3E 0x3F 0x40 0x41
I2c Block read and save test successfull

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : 2
```

```
Enter block to be tested : 1
EEPROM block verify Successful

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : x
EEPROM block test exiting...



**** EEPROM TEST ****
Menu:
1. Base  board test
2. JAMR3 board test
a. Test  All
x. Exit
Select test board : 2
Probing EEPROM with slave address 0x51 successfull

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : 1
Enter block to be tested : 2
Block Data: 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
I2c Block read and save test successfull

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : 2
Enter block to be tested : 2
EEPROM block verify Successful

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
```

```
x. Exit
Select test type : 1
Enter block to be tested : 2
Block Data: 0x2
0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xA
0xB 0xC 0xD 0xE 0xF 0x10 0x11 0x12
0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A
0x1B 0x1C 0x1D 0x1E 0x1F 0x20 0x21 0x22
0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A
0x2B 0x2C 0x2D 0x2E 0x2F 0x30 0x31 0x32
0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A
0x3B 0x3C 0x3D 0x3E 0x3F 0x40 0x41
I2c Block read and save test successfull

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : x
EEPROM block test exiting...


**** EEPROM TEST ****
Menu:
1. Base  board test
2. JAMR3 board test
a. Test  All
x. Exit
Select test board : x
EEPROM board diagnostic test exiting...
```

Please see the example log from the console for the eeprom test for TDA3xx / TDA2EX

```
Probing EEPROM with slave address 0x50 successfull
**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : 1
Enter block to be tested : 2
Block Data: 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
```

```
I2c Block read and save test successfull

**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : 2
Enter block to be tested : 2
EEPROM block verify Successful
**** BLOCK TEST ****
Menu:
1. Block read and Save
2. Block verify
x. Exit
Select test type : x
EEPROM block test exiting...


**** EEPROM TEST ****
Menu:
1. Base  board test
2. JAMR3 board test
a. Test  All
x. Exit
Select test board : x
EEPROM board diagnostic test exiting...
```

8.2.6. UART3 Test

This example is in <install_path>/examples/uart/uart3

UART3 test example runs from CORTEXA15 core for TDA2xx and M4 for TDA3xx to verify that UART3 can transmit and receive data through terminal on PC (115200 8 N 1).

TDA2xx: To select UART3 set select signal SEL_UART3_SPI2 to low on RU111. For this on IO expander U57, configure pin P16 to low. To select UART3 for terminal set UART_SEL1_3 i.e. pin P14 to high.

TDA3xx: To select UART3, on IO expander TCA6424A -2, configure MUX_CNTL[8] to low.

To run UART3 test example, please follow the steps below.

1) Set the SW5 on EVM to 0001000000 for TDA2xx
2) Open CCS & launch the target configuration.
3) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx ..
4) Connect the UART terminal & launch Uart console with setup 115200 8 N 1.
5) Load the “uart3_test_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\uart3_test_app\bin\tda2xx for TDA2xx / “uart3_test_ app_m4_release.xem4” present in “<starterware_rootdir>\binary\uart3_test_app\bin\tda3xx for TDA3xx and execute it from CCS.
6) Enter some data on UART console followed by a newline.

7) Entered data is echoed back on UART Console.
8) Check if echoed data is same as entered data and verify UART3 functionality.

8.2.7. UART1 Test

This example is in <install_path>/examples/uart/uart1

UART1 test example runs from CORTEXA15 core for TDA2xx and TDA2Ex and M4 Core for TDA3xx to verify that UART1 can transmit and receive data through terminal on PC (115200 8 N 1).

TDA3xx: To select UART1, on IO expander TCA6424A -2, configure MUX_CNTL[0] , MUX_CNTL[9] to high and MUX_CNTL[10] to low .

To run UART1 test example, please follow the steps below.

1) Set the SW5 on EVM to 0000000000 for TDA2xx.
2) Open CCS & launch the target configuration.
3) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx .
4) Connect the UART terminal & launch Uart console with setup 115200 8 N 1.
5) Load the “uart1_test_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\uart1_test_app\bin\tda2xx for TDA2xx / “uart1_test_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\uart1_test_app\bin\tda2ex for TDA2Ex / “uart1_test_ app_m4_release.xem4” present in “<starterware_rootdir>\binary\uart1_test_app\bin\tda3xx for TDA3xx and execute it from CCS.
6) Enter some data on UART console followed by a newline.
7) Entered data is echoed back on UART Console.
8) Check if echoed data is same as entered data and verify UART1 functionality.

8.2.8. Temperature Sensor Test

This example is in <install_path>/examples/i2c_diag_test/temp_sensor_app

The temperature sensor test is an example running from CORTEXA15 core for TDA2xx and TDA2Ex and M4 core for TDA3xx to read the temperature from the I2C slave device (TMP102AIDRLT) on the EVM and convert the hexadecimal value to degrees and then print on the console. On the EVM TMP102AIDRLT temperature sensor (U117 of TDA2xx, TDA2Ex and TDA3xx) is connected to i2c instance zero and has slave address 0x48.

To run temperature sensor test example, please follow the steps below.

1) Open CCS & launch the target configuration.
9) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx .
2) Connect the UART terminal & launch Uart console.
3) Load the “temp_sensor_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\temp_sensor_app\bin\tda2xx for TDA2xx / “temp_sensor_app_a15host_release.xa15fg” present in “<starterware_rootdir>\binary\temp_sensor_app\bin\tda2ex for TDA2Ex / “temp_sensor_

app_m4_release.xem4" present in "<starterware_rootdir>\binary\temp_sensor_app\bin\tda3xx for TDA3xx and execute it from CCS.

4) On successful execution temperature (in degree Celsius) is displayed on the UART console or test failure status is displayed on the UART console.

Please see the example log from the console for the temperature sensor test for TDA3xx.

```
I2C0 Passed for address 0x1B!!!

-----------------------------------------
-----------------------------------------

I2C1 Passed for address 0x2C!!!


Platform              : EVM
SOC                   : TDA3XX
SOC Revision          : ES1.0
FT Revision           : 0
Core                  : M4
Board Detected        : TDA3XX BASE
EEPROM Base Board Name    : ADAS-LOW
Base Board Revision       : REV A
Daughter Card Revision    : REV A

Probing Temperature sensor with slave address 0x48 successful
 Temperature is: 16 degree C
Temperature sensor test successful
```

8.2.9. Boot-up Test

This example is in <install_path>/examples/ boot

The boot test is an example running from CORTEXA15 core for TDA2xx and M4 core for TDA3xx to read the boot configuration switch in different boot modes (TDA2xx : sd boot, qspi boot, nor boot / TDA3xx : qspi boot, nor boot).

TDA2xx:

To do boot test the "boot_app_a15host_release.xa15fg" image to be convert into multicore image as required by the boot memory (little/big endian) format.

To run boot test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode.
3) Connect the UART terminal & launch Uart console.

   SD boot test:

   Copy the MLO file and multicore image into sd card boot partition and set the boot switches to sd boot mode and then reset the EVM.

   QSPI boot test:

Load qspi flash writer and flash the tiimage to 0th location of qspi and copy multicore image at 0x8000 And set the boot switches in qspi boot mode then reset the EVM.

4) After the completion see the logs on the console which reads the boot switch configuration of corresponding boot memory.
5) The example will finish testing the boot test and prints the appropriate message on the UART console if the test passes or fails.

TDA3xx:

To run boot test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Set the SYSBOOT[0:7] switch to QSPI/ NOR bootMode for TDA3xx .
3) Connect the UART terminal & launch Uart console.
4) Load the "boot_ app_m4_release.xem4" present in "<starterware_rootdir>\binary\ boot_app\bin\tda3xx and execute it from CCS.
5) On successful execution see the logs on the console which reads the boot switch configuration of corresponding boot memory.

8.2.10. *I2c all test*

This example is in <install_path>/examples/ i2c_diag_test/i2c_all

The i2c all test is an example running from CORTEXA15 core to run all the i2c diagnostic test cases.

To run i2c all test examples, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card.
3) Power ON bit 10 of userconfig switch (SW5) to provide write access to EEPROM.
4) Connect the UART terminal & launch Uart console.
5) Load the "i2c_test_app_a15host_release.xa15fg" present in "<starterware_rootdir>\binary\i2c_test_app\bin\tda2xx and execute it from CCS.
6) On UART console appropriate messages will be displayed for all the tests. The application also toggles the user LED's on the EVM and also prints the test success or failure message.

Please see the example log from the console for the i2c all test.

```
eeprom_test:

write Test successfull

I2c read/write successfull 0xFE

EEPROM read Test successful
```

```
temperature_test:

TEMPERATURE read Test success

temperature is: 39 degree C



pmic reading id:

VENDOR_ID_LSB: 0x51

VENDOR_ID_MSB: 0x4

PRODUCT_ID_LSB: 0x35

PRODUCT_ID_MSB: 0xC0

PMIC ID read Test success

PMIC WRITE/READ Test success 0x44



TOGGLE LED'S ON GPIO_EXPANDER ..

TOGGLE LED'S DONE ..




testing GPIO_EXPANDER_2:

I2c read successfull

WRITE/READ Test successfull 0x00x0

I2c read successfull

WRITE/READ Test2 successfull 0xFF0xFF
```

8.2.11. LCD Test

This example is in <install_path>/examples/DssApp

LCD test example runs from CORTEX-M4 core and generates various display patterns on the panel. The display patterns are of size 720x480 and begin from coordinates (40, 0). The background color is set to black. There are total four display patterns and 900 frames are displayed per pattern in the sequence: Align, Bit, Color & Image.

To run LCD test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode.
3) Load the binary "DssApp_m4_release.xem4" present in "<starterware_rootdir>\binary\DssApp\bin\tda2xx" for TDA2xx / "<starterware_rootdir>\binary\DssApp\bin\tda2ex" for TDA2Ex / "<starterware_rootdir>\binary\DssApp\bin\tda3xx" for TDA3xx and execute it from CCS.
4) Check if all the patterns are displayed correctly to verify LCD functionality.

8.2.12. SD Card Test

This example is in <install_path>/examples/sd_fileIO

The SD card test is a test running from a15 core of tda2xx EVM and tda2ex EVM and m4 core of tda3xx EVM. This example runs test case to test the SD Card form either base board or JAMR3 board (for TDa2xx alone). This application provides UART interface to select SD card to be tested from base board or JAMR3 board (for TDa2xx alone).

To run the SD card test example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card for TDA2xx and TDA2Ex and SYSBOOT[0:7] switch to QSPI 4 bit Mode for TDA3xx ..
3) Connect the UART terminal & launch Uart console.
4) Load the "mmcsd_fileIO_app_a15host_release.xa15fg" present in <starterware_rootdir>/ /binary/mmcsd_fileIO_app/bin/tda2xx for TDA2xx / "mmcsd_fileIO_app_a15host_release.xa15fg" present in <starterware_rootdir>/ /binary/mmcsd_fileIO_app/bin/tda2ex for TDA2Ex / "mmcsd_fileIO_ app_m4_release.xem4" present in <starterware_rootdir>/ /binary/mmcsd_fileIO_app/bin/tda3xx for TDA3xx and execute it from CCS.
5) The example will finish testing the SD card and prints the appropriate message on the UART console if the test passes or fails.

NOTE: SD card file IO test fails to open or create file if executed for second time without board reset for TDA2xx.

Please see the example log from the console for the test for TDA3xx.

```
**** SD CARD FILE IO TEST ****
I2C0 Passed for address 0x1B!!!
I2C0 Passed for address 0x20!!!
-----------------------------------------------
I2C1 Passed for address 0x2C!!!


Platform              : EVM
SOC                   : TDA3XX
SOC Revision          : ES1.0
FT Revision           : 0
```

```
Core                    : M4
Board Detected          : TDA3XX BASE
EEPROM Base Board Name  : ADAS-LOW
Base Board Revision     : REV A
Daughter Card Revision  : REV A

 SD Card Init Done.
Testing File read/write functions on SD CARD...
 FILEIO testapp completed successfully

SD CARD FILE IO board diagnostic test exiting...
```

8.2.13. NOR Flash Read Write Test

This example is in <install_path>/examples/nor/nor_read_write

The NOR flash read write test is a test running from a15 core of tda2xx and tda2ex EVM and m4 core of tda3xx-EVM. This example runs three test cases to test the entire NOR flash.

1) The first test case reads device ID and manufacturer ID and prints it on the console.
2) The second test case does an erase, memory read/write of one block of NOR flash. It erases, writes a pattern to the NOR flash, reads back the data in memory and checks if the patterns match.
3) The third test case erases the entire NOR flash, writes a pattern to the NOR flash, reads back the data and checks if the patterns match.

To run the NOR flash test example, please follow the steps below.

6) Open CCS & launch the target configuration.
7) Change the SYSBOOT Switch to debug mode for TDA2xx and TDA2Ex.
8) Connect the UART terminal & launch Uart console.
9) Load the “nor_read_write_a15host_release.xa15fg” present in <starterware_rootdir>/binary/nor_read_write/bin/tda2xx for TDA2xx / “nor_read_write_a15host_release.xa15fg” present in <starterware_rootdir>/binary/nor_read_write/bin/tda2ex for TDA2Ex / “nor_read_write_m4_release.xem4” present in <starterware_rootdir>/binary/nor_read_write/bin/tda3xx for TDA3xx and execute it from CCS.
10) Enter “1”, “2” or “3” depending on the test you want to run.
11) The example will finish testing the NOR flash and print the appropriate message on the UART console if the test passes or fails.
12) If the test passes the UART console will prompt again to run another testcase.

Please see the example log from the console for the test for TDA2xx.

```
NOR Flash Test App
Enter the testcase number
1: Read Manufacturer ID and Device ID of NOR
2: Erase, Write/Read and Verify 1 block of NOR flash
3: Erase, Write/Read and Verify entire NOR flash
```

```
Any other number to quit
1
Running Test Case 1: Read Manufacturer ID and Device ID of NOR
CFI Query...passed.
NOR Initialization:
    Command Set: AMD
    Manufacturer: AMD
    Size: 0x40 MB

Test Passed.

Enter the testcase number
1: Read Manufacturer ID and Device ID of NOR
2: Erase, Write/Read and Verify 1 block of NOR flash
3: Erase, Write/Read and Verify entire NOR flash
Any other number to quit
2

Running Test Case 2:Erase, Write/Read and Verify 1 block of NOR flash
CFI Query...passed.
NOR Initialization:
    Command Set: AMD
    Manufacturer: AMD
    Size: 0x40 MB

Erasing the NOR Flash upto range: 0x8020000
Erased through 0x8020000
Erase Completed

Writing 0x20000bytes to NOR...
NOR Write OK through 0x8008000.
NOR Write OK through 0x8010000.
NOR Write OK through 0x8018000.
NOR Write OK through 0x8020000.

Test Passed.

Enter the testcase number
1: Read Manufacturer ID and Device ID of NOR
2: Erase, Write/Read and Verify 1 block of NOR flash
3: Erase, Write/Read and Verify entire NOR flash
Any other number to quit
3

Running Test Case 3: Erase, Write/Read and Verify entire NOR flashCFI Query...passed.
NOR Initialization:
```

```
        Command Set: AMD
        Manufacturer: AMD
        Size: 0x40 MB
NOR Init Done

Erasing the NOR Flash upto range: 0xC000000
Erased through 0x8020000
Erased through 0x8040000
Erased through 0x8060000
Erased through 0x8080000
Erased through 0x80A0000
.
.
Erased through 0xBF80000
Erased through 0xBFA0000
Erased through 0xBFC0000
Erased through 0xBFE0000
Erased through 0xC000000
Erase Completed

Writing 0x20000bytes to NOR...
NOR Write OK through 0x8008000.
NOR Write OK through 0x8010000.
NOR Write OK through 0x8018000.
NOR Write OK through 0x8020000.
Testing block no.0Completed

Writing 0x20000bytes to NOR...
NOR Write OK through 0x8028000.
NOR Write OK through 0x8030000.
NOR Write OK through 0x8038000.
NOR Write OK through 0x8040000.
Testing block no.1Completed

Writing 0x20000bytes to NOR...
NOR Write OK through 0x8048000.
NOR Write OK through 0x8050000.
NOR Write OK through 0x8058000.
NOR Write OK through 0x8060000.
Testing block no.2Completed
.
.
.
.
Writing 0x20000bytes to NOR...
NOR Write OK through 0xBFA8000.
```

```
NOR Write OK through 0xBFB0000.
NOR Write OK through 0xBFB8000.
NOR Write OK through 0xBFC0000.
Testing block no.509Completed

Writing 0x20000bytes to NOR...
NOR Write OK through 0xBFC8000.
NOR Write OK through 0xBFD0000.
NOR Write OK through 0xBFD8000.
NOR Write OK through 0xBFE0000.
Testing block no.510Completed

Writing 0x20000bytes to NOR...
NOR Write OK through 0xBFE8000.
NOR Write OK through 0xBFF0000.
NOR Write OK through 0xBFF8000.
NOR Write OK through 0xC000000.
Testing block no.511Completed

Enter the testcase number
1: Read Manufacturer ID and Device ID of NOR
2: Erase, Write/Read and Verify 1 block of NOR flash
3: Erase, Write/Read and Verify entire NOR flash
Any other number to quit
 4
NOR Flash Test App

Exiting the nor test
```

8.2.14. Video Loopback Test

This example is in <install_path>/examples/videoLoopback/

The video loopback test runs from M4 of tda2xx and tda2ex EVM and tda3xx EVM.

This example configures the OV sensor, does VPS init and then captures the images from the sensor and displays it on LCD. It uses single common buffer for both capture to update the captured image and display to display the captured image. This example is not supported for multiple different buffers for capture and display. So videoLoopback output image quality will not be good for non-static images.

Running Capture example on TDA2xx and TDA2Ex:

Make sure OV sensor is connected to the vision app board before running the example.

For REV B Vision application board, the video config SW3 present in the Vision application board should be set to

SW3-1 OFF

SW3-2 ON

SW3-3 OFF

SW3-4 ON

SW3-5 OFF

SW3-6 ON

SW3-7 OFF

SW3-8 ON

Running Capture example on TDA3xx:

Make sure OV sensor is connected to the baseboard before running the example.

To run the video loopback example, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Connect CortexA15_0 core and run the script in DRA7xx MULTICORE Initialization-> IPU1SSClkEnable_API for TDA2xx
3) Connect CortexM4_IPU1_C0 core
4) Load the "videoLoopback_m4_release.xem4" present in <starterware_rootdir>/binary/ videoLoopback /bin/tda2xx for TDA2xx/ in <starterware_rootdir>/binary/ videoLoopback /bin/tda2ex for TDA2Ex in <starterware_rootdir>/binary/ videoLoopback /bin/tda3xx for TDA3xx on M4 and execute it from CCS.
5) The example will run the loop back application and the video can be seen on the LCD display.

8.2.15. McASP Sinetone Test

This example is in <install_path>/examples/mcasp/mcasp_sinetone

The mcasp Sinetone test is an example running on tda2xx/tda2ex A15 and DSP core. This example demonstrates the sinetone generation using mcasp as master. Data to generate Sinetone is transmitted in DMA mode. This could be analyzed using any PC Audio Analyzer tool by connecting Line Out (P13 port) of EVM to Line in of PC.

To run McASP Sinetone test, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card.
3) Load the "mcasp_sinetone_app_a15host_release.xa15fg" on A15 core or "mcasp_sinetone_app_c66x_release.xe66" on DSP core present in "<starterware_rootdir>/binary/mcasp_sinetone_app/bin/tda2xx or "<starterware_rootdir>/binary/mcasp_sinetone_app/bin/tda2ex and execute it from CCS.
4) For running from DSP core A15 core needs to be in running state before running binary on DSP.
5) On successful execution Sinetone can be heard from the audio lineout by connecting head phone or can be analyzed using audio analyzer tools by connecting to Line In port of PC.

Please see the example log from the console for the McASP Sinetone app.

```
McASP Sinetone application
GPIO Expander configured
 Codec Init successful
DAC Initialization successful
```

```
I2S DMA PAram init done
McASP Configured
```

8.2.16. MPU Retention, DSP, IPU, EVE CPU Idle and Wakeup Test

This example is in <install_path>/ examples/pm/cpuidle for MPU, IPU, DSP and in <install_path>/ examples\pm\arp32_cpuidle for EVE. (Note: ARP32 Compiler installation is required for EVE)

The MPU Retention and wakeup test is an example running on tda2xx A15 core. The DSP, IPU and EVE CPU Idle tests run on their respective cores. This example demonstrates the ability to take the MPU/IPU/DSP/EVE to low power retention mode and then wakeup using a timer interrupt. The example loops through the low power state and wakeup cycle 10 times before declaring success. The example illustrates the use of Power Management HAL and Library APIs which allow the MPU to go to retention and IPU/DSP/EVE to clock gated state.

To run the MPU Retention and Wakeup test, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card.
3) Load the “pm_cpuidle_test_app_a15host_release.xa15fg” on A15 or “pm_cpuidle_test_app_c66x_release.xe66” on DSP, “pm_cpuidle_test_app_m4_release.xem4” on M4 core present in “<starterware_rootdir> \binary\pm_cpuidle_test_app\bin\<PLATFORM>” and execute it from CCS.
4) On successful execution the following output can be seen on the UART Console.

Please see the example log from the console for the PM CPUIdle Test App.
MPU:

```
PM CPUIdle Test App
XBar is sucessfully connected to inst:35
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
```

```
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Enter Idle
Enter Targeted Power State successfully
Exit Idle
Test Completed!!
```

DSP:

```
PM CPUIdle Test App
XBar is successfully connected to inst:35
Enter Idle
Time Taken to Wakeup:996
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:977
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:979
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:981
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:980
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:982
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:984
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:983
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:979
Enter Targeted Power State successfully
Enter Idle
Time Taken to Wakeup:981
Enter Targeted Power State successfully
Test Completed!!
```

IPU:

```
PM CPUIdle Test App
XBar is sucessfully connected to inst: 35
Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
```

```
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Enter Idle
Exit Idle
Enter Targeted Power State successfully

Test Completed!!
```

8.2.17. Clock Rate Configuration Test

This example is in <install_path>/ examples/pm/ clkrate_manager

The Clock Rate Configuration test is an example running on tda2xx A15 core, tda2xx M4 Core and tda3xx IPU (M4) core. This example demonstrates the ability to read the clock rate for different clocks for a given CPU (MPU/IPU/DSP/GPU/IVA/EVE). The example first reads the current clock configuration and then checks for OPP_NOM, OPP_OD and OPP_HIGH frequencies along with voltage changes by using the PMLIB clock rate APIs before declaring pass or fail. The example illustrates the use of Power Management LIB which allows changing the CPU OPP.

To run the MPU Retention and Wakeup test, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card.

3) Load the “pm_clkrate_test_app_a15host_release.xa15fg” on A15 core for TDA2xx or “pm_clkrate_test_app_m4_release.xem4” on M4 core for TDA3xx/TDA2xx present in “<starterware_rootdir> \binary\ pm_clkrate_test_app\bin\tda2xx” or “<starterware_rootdir> \binary\ pm_clkrate_test_app\bin\tda3xx” and execute it from CCS.

4) On successful execution the following output can be seen on the UART Console for TDA2xx or CCS Console for TDA3xx.

Please see the example log from the console for the Clock Rate Configuration test on TDA2xx.

```
PM ClockRate Test App
Clkrate Manager Cpu Set and Get Frequency test:

the Clock Rate passed for ModID PMHAL_PRCM_MOD_GPU ClkId PMHAL_PRCM_CLK_GPU_HYD_GCLK is
425600000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_GPU ClkId PMHAL_PRCM_CLK_GPU_HYD_GCLK Passed
After Clock Rate Set, the Clock Rate for ModID PMHAL_PRCM_MOD_GPU ClkId
PMHAL_PRCM_CLK_GPU_HYD_GCLK is 500000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_GPU ClkId PMHAL_PRCM_CLK_GENERIC is 425600000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_GPU ClkId PMHAL_PRCM_CLK_GENERIC Passed
After Clock Rate Set, the Clock Rate for ModID PMHAL_PRCM_MOD_GPU ClkId PMHAL_PRCM_CLK_GENERIC is
532000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_DSP1 ClkId PMHAL_PRCM_CLK_GENERIC is 600000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_DSP1 ClkId PMHAL_PRCM_CLK_GENERIC Passed
After Clock Rate Set, the Clock Rate for ModID PMHAL_PRCM_MOD_DSP1 ClkId PMHAL_PRCM_CLK_GENERIC is
600000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_EVE1 ClkId PMHAL_PRCM_CLK_GENERIC is 535000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_EVE1 ClkId PMHAL_PRCM_CLK_GENERIC Passed
After Clock Rate Set, the Clock Rate for ModID PMHAL_PRCM_MOD_EVE1 ClkId PMHAL_PRCM_CLK_GENERIC is
650000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_MPU ClkId PMHAL_PRCM_CLK_GENERIC is 750000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_MPU ClkId PMHAL_PRCM_CLK_GENERIC Passed
After Clock Rate Set, the Clock Rate for ModID PMHAL_PRCM_MOD_MPU ClkId PMHAL_PRCM_CLK_GENERIC is
1500000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_MPU ClkId PMHAL_PRCM_CLK_GENERIC is 1500000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_MPU ClkId PMHAL_PRCM_CLK_GENERIC Passed
After Clock Rate Set, the Clock Rate for ModID PMHAL_PRCM_MOD_MPU ClkId PMHAL_PRCM_CLK_GENERIC is
1176000000 Hz

Clkrate Manager Cpu Set and Get Frequency test Completed!!

Clkrate Manager Bypass Freq Test:

the Clock Rate passed for ModID PMHAL_PRCM_MOD_DSP1 ClkId PMHAL_PRCM_CLK_GENERIC is 532000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_EVE1 ClkId PMHAL_PRCM_CLK_GENERIC is 650000000 Hz
```

Clkrate Manager Bypass Freq Test Completed!!

Clkrate Manager Module Set Frequency test:

the Clock Rate passed for ModID PMHAL_PRCM_MOD_DSS ClkId PMHAL_PRCM_CLK_DSS_GFCLK with clkrate 192000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_DSS ClkId PMHAL_PRCM_CLK_DSS_GFCLK Passed for clkRate 192000000 Hz
After Clock Rate Set,Clock Rate for ModID PMHAL_PRCM_MOD_DSS ClkId PMHAL_PRCM_CLK_DSS_GFCLK is 192000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_CPGMAC ClkId PMHAL_PRCM_CLK_GMAC_RFT_CLK with clkrate 266000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_CPGMAC ClkId PMHAL_PRCM_CLK_GMAC_RFT_CLK Passed for clkRate 451584000 Hz
After Clock Rate Set,Clock Rate for ModID PMHAL_PRCM_MOD_CPGMAC ClkId PMHAL_PRCM_CLK_GMAC_RFT_CLK is 451584000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_I2C5 ClkId PMHAL_PRCM_CLK_IPU_96M_GFCLK with clkrate 96000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_I2C5 ClkId PMHAL_PRCM_CLK_IPU_96M_GFCLK Passed for clkRate 96000000 Hz
After Clock Rate Set,Clock Rate for ModID PMHAL_PRCM_MOD_I2C5 ClkId PMHAL_PRCM_CLK_IPU_96M_GFCLK is 96000000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_MCASP1 ClkId PMHAL_PRCM_CLK_MCASP1_AHCLKR with clkrate 112896000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_MCASP1 ClkId PMHAL_PRCM_CLK_MCASP1_AHCLKR Passed for clkRate 56448000 Hz
After Clock Rate Set,Clock Rate for ModID PMHAL_PRCM_MOD_MCASP1 ClkId PMHAL_PRCM_CLK_MCASP1_AHCLKR is 56448000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_TIMER5 ClkId PMHAL_PRCM_CLK_TIMER5_GFCLK with clkrate 20000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_TIMER5 ClkId PMHAL_PRCM_CLK_TIMER5_GFCLK Passed for clkRate 451584000 Hz
After Clock Rate Set,Clock Rate for ModID PMHAL_PRCM_MOD_TIMER5 ClkId PMHAL_PRCM_CLK_TIMER5_GFCLK is 451584000 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_DCAN1 ClkId PMHAL_PRCM_CLK_DCAN1_SYS_CLK with clkrate 20000000 Hz
ClockSET for ModID PMHAL_PRCM_MOD_DCAN1 ClkId PMHAL_PRCM_CLK_DCAN1_SYS_CLK Passed for clkRate 22579200 Hz
After Clock Rate Set,Clock Rate for ModID PMHAL_PRCM_MOD_DCAN1 ClkId PMHAL_PRCM_CLK_DCAN1_SYS_CLK is 22579200 Hz

the Clock Rate passed for ModID PMHAL_PRCM_MOD_GPIO1 ClkId PMHAL_PRCM_CLK_WKUPAON_SYS_GFCLK with clkrate 32786 Hz
ClockSET for ModID PMHAL_PRCM_MOD_GPIO1 ClkId PMHAL_PRCM_CLK_WKUPAON_SYS_GFCLK Passed for clkRate 32786 Hz
After Clock Rate Set,Clock Rate for ModID PMHAL_PRCM_MOD_GPIO1 ClkId PMHAL_PRCM_CLK_WKUPAON_SYS_GFCLK is 32786 Hz

```
Clkrate Manager module Set Frequency test Completed!!
```

8.2.18. PM System Configuration Test

This example is in <install_path>/ examples/pm/systemconfig

The PM System Configuration test is an example running on tda2xx A15 core, tda2xx M4 core and tda3xx IPU (M4) core. This example demonstrates the ability to configure the desired power state for a given module based on the entries in the power spread sheet. The example loops through the different modules and power states and tries to program the same for each module using PM LIB sysconfig APIs before declaring pass or fail. The example illustrates the use of Power Management LIB which allows system configuration.

To run the PM System Configuration test, please follow the steps below.

1) Open CCS & launch the target configuration.
2) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card.
3) Load the "pm_systemconfig_test_app_a15host_release.xa15fg" on A15 core for TDA2xx or "pm_systemconfig_test_app_m4_release.xem4" on M4 core for TDA3xx present in "<starterware_rootdir> \binary\ pm_systemconfig_test_app\bin\tda2xx" or "<starterware_rootdir> \binary\ pm_systemconfig_test_app\bin\tda3xx" and execute it from CCS.
4) On successful execution the following output can be seen on the UART Console for TDA2xx or CCS Console for TDA3xx.

Please see the example log from the console for the PM System Configuration test on TDA2xx.

```
PM System Config Test App
PMHAL_PRCM_MOD_OCP2SCP1: PASS
PMHAL_PRCM_MOD_OCP2SCP3: PASS
PMHAL_PRCM_MOD_ATL: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_DUMMY_MODULE4: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_DUMMY_MODULE1: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_DUMMY_MODULE2: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_DUMMY_MODULE3: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_IO_SRCOMP_CORE: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SMARTREFLEX_CORE: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SMARTREFLEX_DSPEVE: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SMARTREFLEX_GPU: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SMARTREFLEX_IVAHD: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SMARTREFLEX_MPU: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_USB_PHY1_ALWAYS_ON: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_USB_PHY2_ALWAYS_ON: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_USB_PHY3_ALWAYS_ON: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_EFUSE_CTRL_CUST: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
```

PMHAL_PRCM_MOD_DMA_SYSTEM: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_DSP1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_DSP2: PMLIB_SYS_CONFIG_AUTO_CG PASS
PMHAL_PRCM_MOD_BB2D: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_DSS: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SDVENC: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_DLL: No Valid Module Mode, Cannot be enabled from software
Optional Clocks if any have been enabled
PMHAL_PRCM_MOD_DMM: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_EMIF1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_EMIF2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_EMIF_OCP_FW: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_EVE1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_EVE2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_EVE3: PMLIB_SYS_CONFIG_AUTO_CG PASS
PMHAL_PRCM_MOD_EVE4: PMLIB_SYS_CONFIG_AUTO_CG PASS
PMHAL_PRCM_MOD_CPGMAC: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_GPU: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_I2C5: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MCASP1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_TIMER5: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER6: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER7: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER8: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_UART6: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_IPU1: PMLIB_SYS_CONFIG_AUTO_CG PASS
PMHAL_PRCM_MOD_IPU2: PMLIB_SYS_CONFIG_AUTO_CG PASS
PMHAL_PRCM_MOD_IVA: PMLIB_SYS_CONFIG_AUTO_CG PASS
PMHAL_PRCM_MOD_SL2: PMLIB_SYS_CONFIG_AUTO_CG PASS
PMHAL_PRCM_MOD_IEEE1500_2_OCP: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MMC1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MMC2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MLB_SS: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_SATA: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_OCP2SCP1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_OCP2SCP3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_USB_OTG_SS1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_USB_OTG_SS2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_USB_OTG_SS3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_USB_OTG_SS4: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_CTRL_MODULE_BANDGAP: Module has dependencies
PMHAL_PRCM_MOD_DLL_AGING: Module has dependencies
PMHAL_PRCM_MOD_L3_INSTR: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_L3_MAIN_2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_OCP_WP_NOC: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_GPMC: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_L3_MAIN_1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MMU_EDMA: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MMU_PCIESS: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_OCMC_RAM1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_OCMC_RAM2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_OCMC_RAM3: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS

PMHAL_PRCM_MOD_OCMC_ROM: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SPARE_IVA2: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_VCP1: Module has dependencies
PMHAL_PRCM_MOD_VCP2: Module has dependencies
PMHAL_PRCM_MOD_SPARE_CME: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_HDMI: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_ICM: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_SATA2: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_UNKNOWN4: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_UNKNOWN5: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_UNKNOWN6: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_VIDEOPLL1: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_VIDEOPLL2: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_VIDEOPLL3: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_TPCC: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TPTC1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TPTC2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_L4_CFG: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_OCP2SCP2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SAR_ROM: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_SPARE_SMARTREFLEX_RTC: No Valid Module Mode, Cannot be enabled from software
Optional Clocks if any have been enabled
PMHAL_PRCM_MOD_SPARE_SMARTREFLEX_SDRAM: No Valid Module Mode, Cannot be enabled from software
Optional Clocks if any have been enabled
PMHAL_PRCM_MOD_SPARE_SMARTREFLEX_WKUP: No Valid Module Mode, Cannot be enabled from software
Optional Clocks if any have been enabled
PMHAL_PRCM_MOD_SPINLOCK: Module has dependencies
PMHAL_PRCM_MOD_IO_DELAY_BLOCK: No Valid Module Mode, Cannot be enabled from software
Optional Clocks if any have been enabled
PMHAL_PRCM_MOD_MAILBOX1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MAILBOX10: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MAILBOX11: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MAILBOX12: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MAILBOX13: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MAILBOX2: Module has dependencies
PMHAL_PRCM_MOD_MAILBOX3: Module has dependencies
PMHAL_PRCM_MOD_MAILBOX4: Module has dependencies
PMHAL_PRCM_MOD_MAILBOX5: Module has dependencies
PMHAL_PRCM_MOD_MAILBOX6: Module has dependencies
PMHAL_PRCM_MOD_MAILBOX7: Module has dependencies
PMHAL_PRCM_MOD_MAILBOX8: Module has dependencies
PMHAL_PRCM_MOD_MAILBOX9: Module has dependencies

PMHAL_PRCM_MOD_I2C1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_I2C2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_I2C3: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_I2C4: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_L4_PER1: Module has dependencies
PMHAL_PRCM_MOD_TIMER10: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_TIMER11: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_TIMER2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER3: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER4: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER9: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_ELM: Module has dependencies
PMHAL_PRCM_MOD_HDQ1W: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCSPI1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCSPI2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCSPI3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCSPI4: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_UART1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_UART2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_UART3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_UART4: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_UART5: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_GPIO2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_GPIO3: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_GPIO4: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_GPIO5: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_GPIO6: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_GPIO7: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_GPIO8: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MMC3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MMC4: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_DCAN2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_L4_PER2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_UART7: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_UART8: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_UART9: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_PRUSS1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_PRUSS2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCASP2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCASP3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCASP4: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCASP5: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCASP6: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCASP7: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MCASP8: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_PWMSS1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_PWMSS2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_PWMSS3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_QSPI: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_L4_PER3: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER13: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_TIMER14: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_TIMER15: PMLIB_SYS_CONFIG_DISABLED PASS

```
PMHAL_PRCM_MOD_TIMER16: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_AES1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_AES2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_DES3DES: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_DMA_CRYPTO: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_FPKA: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_RNG: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_SHA2MD51: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_SHA2MD52: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_MPU: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_MPU_MPU_DBG: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_PCIESS1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_CSI1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_CSI2: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_LVDSRX: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_VIP1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_VIP2: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_VIP3: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_VPE: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_DEBUG_LOGIC: No Valid Module Mode, Cannot be enabled from software
Optional Clocks if any have been enabled
PMHAL_PRCM_MOD_MPU_EMU_DBG: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_ADC: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_COUNTER_32K: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_CTRL_MODULE_WKUP: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_DCAN1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_GPIO1: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_IO_SRCOMP_WKUP: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_KBD: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_L4_WKUP: Module has dependencies
PMHAL_PRCM_MOD_SAR_RAM: Module has dependencies
PMHAL_PRCM_MOD_SPARE_SAFETY1: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_SAFETY2: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_SAFETY3: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_SAFETY4: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_UNKNOWN2: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_SPARE_UNKNOWN3: No Valid Module Mode, Cannot be disabled from software
Optional Clocks if any have been disabled
PMHAL_PRCM_MOD_TIMER1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_TIMER12: Module has dependencies
PMHAL_PRCM_MOD_UART10: PMLIB_SYS_CONFIG_DISABLED PASS
PMHAL_PRCM_MOD_WD_TIMER1: PMLIB_SYS_CONFIG_ALWAYS_ENABLED PASS
PMHAL_PRCM_MOD_WD_TIMER2: PMLIB_SYS_CONFIG_DISABLED PASS
--------------------------------------------------
Sysconfig Test case has passed
```

8.2.19. Junction Temperature Sensor Test

This example is in <install_path>/ examples/pm/ junction_temperature_sensor

The Junction Temperature Sensor test is an example running on tda2xx A15 core and tda3xx IPU (M4) core. This example demonstrates the ability to read the temperature sensor registers and determine the junction temperature for different voltage rails. The example first reads the current temperature, the average temperature, enables HOT events for hot thermal alerts and COLD events for cold thermal alerts before declaring pass or fail. The example illustrates the use of Power Management HAL which allows monitoring thermal events.

To run the Junction Temperature test, please follow the steps below.

5) Open CCS & launch the target configuration.
6) Change the SYSBOOT Switch to debug mode or SD mode if booting from SD card.
7) Load the "pm_junctiontemp_test_app_a15host_release.xa15fg" on A15 core for TDA2xx or "pm_junctiontemp_test_app_m4_release.xa15fg" on M4 core for TDA3xx present in "<starterware_rootdir> \binary\pm_cpuidle_test_app\bin\tda2xx" or "<starterware_rootdir> \binary\pm_cpuidle_test_app\bin\tda3xx" and execute it from CCS.
8) On successful execution the following output can be seen on the UART Console for TDA2xx or CCS Console for TDA3xx.

Please see the example log from the console for the Junction Temperature test on TDA2xx. The junction temperature as measured on your sample can show some varying results depending on using a socketed EVM or a non-socketed EVM, nominal/split-lot sample.

```
PM Junction Temperature Measure Test App
----------------------------------------
 Measure the Current Temperature of different Voltage Rails
----------------------------------------
Voltage Domain: PMHAL_PRCM_VD_MPU
Current Temperature Range in Degrees C = [34.800 , 35.200]
Voltage Domain: PMHAL_PRCM_VD_CORE
Current Temperature Range in Degrees C = [34.400 , 34.800]
Voltage Domain: PMHAL_PRCM_VD_IVAHD
Current Temperature Range in Degrees C = [34.800 , 35.200]
Voltage Domain: PMHAL_PRCM_VD_DSPEVE
Current Temperature Range in Degrees C = [34.800 , 35.200]
Voltage Domain: PMHAL_PRCM_VD_GPU
Current Temperature Range in Degrees C = [34.400 , 34.800]
----------------------------------------
 Measure the Average Temperature of different Voltage Rails
----------------------------------------
Voltage Domain: PMHAL_PRCM_VD_MPU
Average Temperature Range in Degrees C = [88.800 , 89.200]
Voltage Domain: PMHAL_PRCM_VD_CORE
Average Temperature Range in Degrees C = [80.0 , 80.400]
Voltage Domain: PMHAL_PRCM_VD_IVAHD
Average Temperature Range in Degrees C = [81.600 , 82.0]
Voltage Domain: PMHAL_PRCM_VD_DSPEVE
Average Temperature Range in Degrees C = [75.200 , 75.600]
Voltage Domain: PMHAL_PRCM_VD_GPU
Average Temperature Range in Degrees C = [82.800 , 83.200]
----------------------------------------
```

```
 Check for Hot events of different Voltage Rails
---------------------------------------
XBar is sucessfully connected to inst:121
Voltage Domain: PMHAL_PRCM_VD_MPU
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_CORE
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_IVAHD
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_DSPEVE
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_GPU
Temperature Event Occurred
---------------------------------------
 Check for Cold events of different Voltage Rails
---------------------------------------
Voltage Domain: PMHAL_PRCM_VD_MPU
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_CORE
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_IVAHD
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_DSPEVE
Temperature Event Occurred
Voltage Domain: PMHAL_PRCM_VD_GPU
Temperature Event Occurred
---------------------------------------
Junction Temperature App Passed!!
---------------------------------------
```

Please see the example log from the console for the Junction Temperature test on TDA3xx. The junction temperature as measured on your sample can show some varying results depending on using a socketed EVM or a non-socketed EVM, nominal/split-lot sample.

```
[Cortex_M4_IPU1_C0]
PM Junction Temperature Measure Test App
---------------------------------------
 Measure the Current Temperature of different Voltage Rails
---------------------------------------
Voltage Domain: PMHAL_PRCM_VD_CORE

Current Temperature Range in Degrees C = [43.400 , 44.0]
---------------------------------------
 Measure the Average Temperature of different Voltage Rails
---------------------------------------
Voltage Domain: PMHAL_PRCM_VD_CORE

Average Temperature Range in Degrees C = [42.400 , 42.800]
---------------------------------------
 Check for Hot events of different Voltage Rails
---------------------------------------
Voltage Domain: PMHAL_PRCM_VD_CORE

Temperature Event Occurred
---------------------------------------
```

```
 Check for Cold events of different Voltage Rails
---------------------------------------
Voltage Domain: PMHAL_PRCM_VD_CORE

Temperature Event Occurred
---------------------------------------
Junction Temperature App Passed!!
---------------------------------------
```

9. Un-installing

The StarterWare installer installs the un-installer in <install_path> directory. The name of the un-installer is *uninstall.exe*. To uninstall these packages, please run the uninstaller.

10. References

1. DM814x TRM : http://www.ti.com/lit/ug/sprugz8c/sprugz8c.pdf