Vision SDK TDA3xx

(v02.08)

User Guide

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards ought to be provided by the customer so as to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations and notices. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is neither responsible nor liable for any such use.

Resale of TI's products or services with *statements different from or beyond the parameters* stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products.
www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

TABLE OF CONTENTS

1 Introduction

Vision Software Development Kit (Vision SDK) is a multi-processor software development package for TI's family of ADAS SoCs. The software framework allows users to create different ADAS application data flows involving video capture, video pre-processing, video analytics algorithms, and video display. The framework has sample ADAS data flows which exercises different CPUs and HW accelerators in the ADAS SoC and demonstrates how to effectively use different sub-systems within the SoC. Frame work is generic enough to plug in application specific algorithms in the system.

Vision SDK is currently targeted for the TDA2xx and TDA3xx family of SoCs

This document explains the HW/SW setup for TDA3x EVM. Refer to VisionSDK_UserGuide_TDA2xx.pdf for TDA2x EVM related HW/SW setup

1.1 References

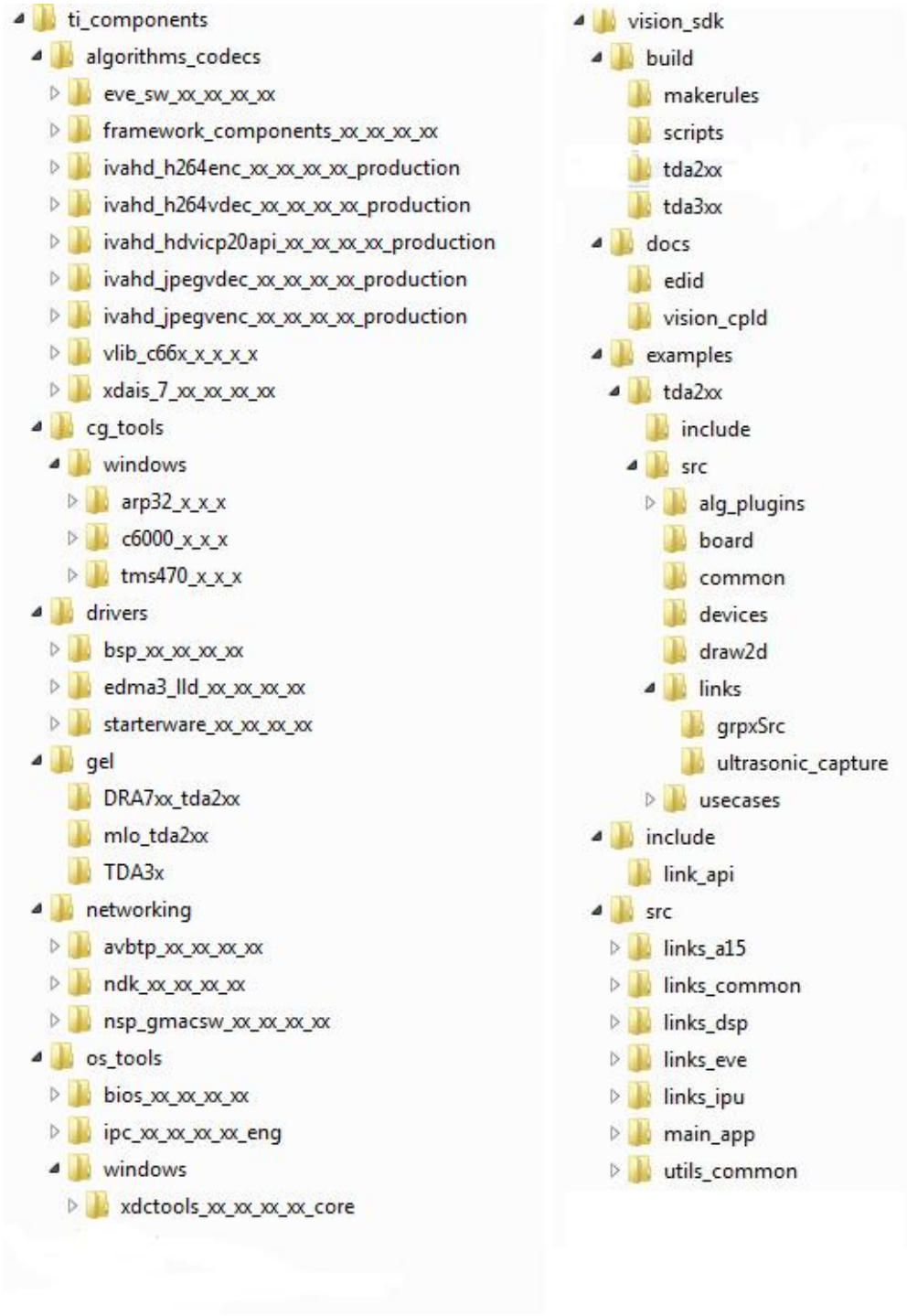
Refer the below additional documents for more information about Vision SDK

Document	Description
VisionSDK_ReleaseNotes.pdf	Release specific information
VisionSDK_UserGuide_TDA3xx.pdf	This document. Contains install, build, execution information
VisionSDK_ApiGuide.CHM	User API interface details
VisionSDK_SW_Architecture.pdf	Overview of software architecture
VisionSDK_DevelopmentGuide.pdf	Details how to create data flow (s) & add new functionality

1.2 Directory Structure

Once Vision SDK is installed, two main directories are created namely vision_sdk and ti_components. Please refer Directory Structure Details for more information

NOTE: vision_sdk\examples\tda2xx contains examples for both TDA3xx and TDA2xx



2 System Requirements

This chapter provides a brief description of the system requirements (hardware and software) and instructions for installing Vision SDK.

2.1 PC Requirements

Installation of this release needs a windows machine with about 6GB of free disk space. Building of the SDK is supported on windows environment.

2.2 Software Requirements

All software packages required to build and run the Vision SDK are included as part of the SDK release package except for the ones mentioned below

2.2.1 Code Composer Studio

CCS is needed to load, run and debug the software. CCS can be downloaded from the below link. CCS version CCS version 6.0.1.00040 should be installed.

http://processors.wiki.ti.com/index.php/Download_CCS

NOTE: Vision SDK can be used with CCS 5.4 and CCS 5.5 as well

2.3 Hardware Requirements

Hardware setup for different use-cases is described in this section

2.3.1 Single Channel (SC) Use-case Hardware Setup

SC use-case needs the below hardware

1. TDA3xx EVM , power supply (12V 5 AMP)
2. Video Sensors, you would require one of the sensors listed in section 2.5. Please refer Figure 2.3.1 / Figure 2.3.2, it visually shows as to where the sensor should be connected.
3. 1Gbps Ethernet Cable (optional)
4. HDMI 1080p60 capable Display Monitor OR LCD Screen, See Figure 2.3.2. 10'' or 7'' LCD is supported

WARNING: LI Camera Interface is different from LI Camera CSI2 Interface. Putting a CSI2 sensor on LI Camera Interface will damage the sensor

2.3.2 VIP multi-channel LVDS capture (SRV) Use-case Hardware Setup

Refer the TDA2x user guide "VisionSDK_UserGuide_TDA2xx.pdf" for the LVDS set-up.

TDA3xx EVM to support multichannel LVDS capture, some board modifications are required. Please refer to the "Running BSP Application on TDA3XX EVM" section of the BSP user guide to get the more details

1. For VIP capture from Multi-deserializer board, the multi-deserializer board should be configured for 4-channel operation. The CN2, CN3 and CN4 jumper settings should be set.
2. In case of Multi-deserializer capture through VIP or ISS capture from sensors, board modification is required in the base board to avoid I2C issues

WARNING: Select the display resolution as HDMI XGA TDM mode <TDA3xx ONLY>. Failing which, only 2 channels of captured video streams is displayed. This is required as some of the VIP input pins is muxed with display output pins.

2.3.3 ISS Multiple Channel (SRV) Use-case Hardware Setup

SRV use-case needs the below hardware

1. TDA3xx EVM, power supply (12V 5 AMP)
2. UB960 Application Board, power supply (12V 5 AMP)
3. 4 TIDA00262 modules & LVDS cables to connect camera modules to UB960 application board
    a. List details of this camera module http://www.ti.com/tool/TIDA-00262?keyMatch=TIDA-00262&tisearch=Search-EN-Everything#tiDevice
4. HDMI 720p60 capable Display Monitor

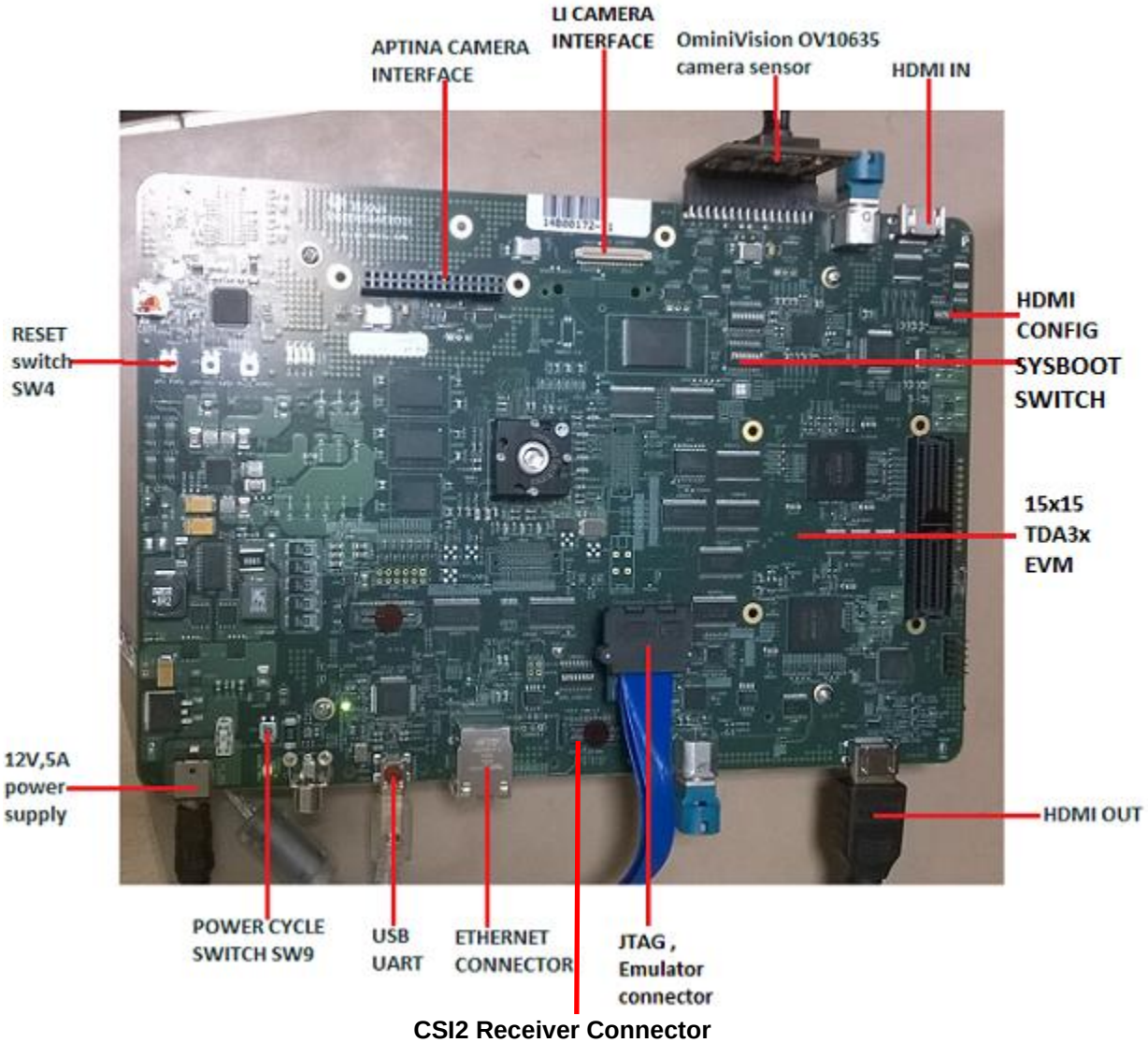


Figure 2.3.1 15x15 TDA3x EVM (FRONT SIDE)

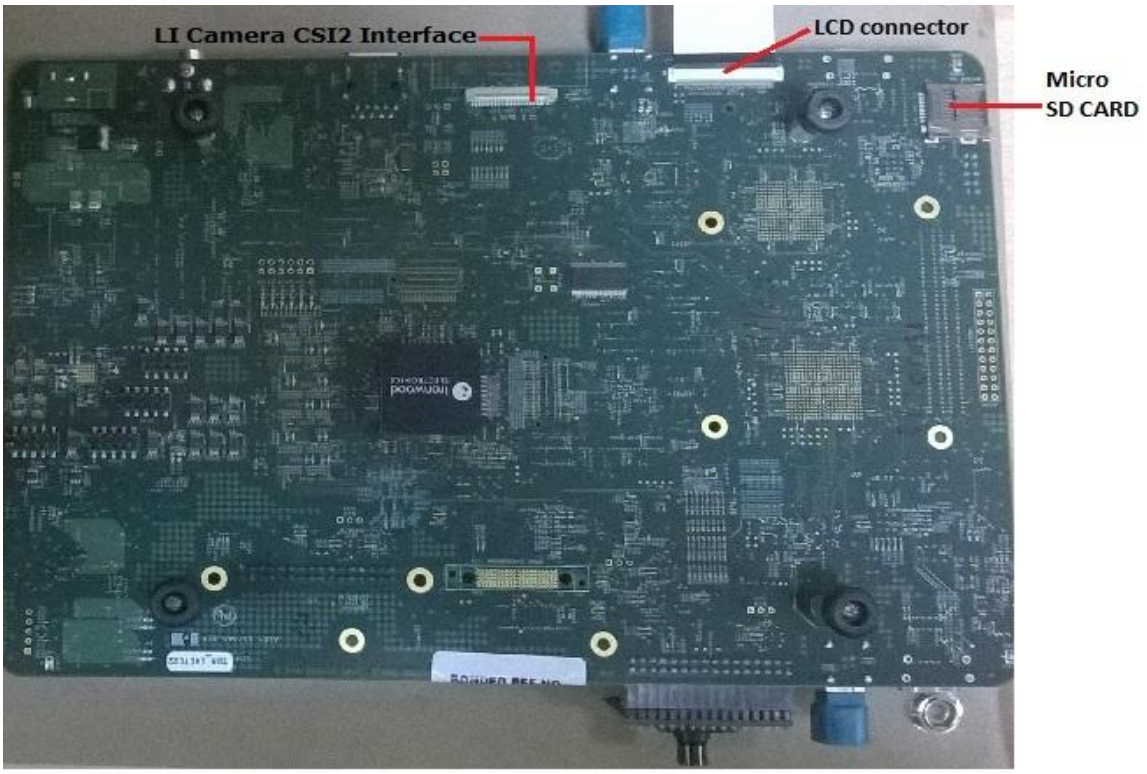


Figure 2.3.2 15x15 TDA3x EVM (BACK SIDE)

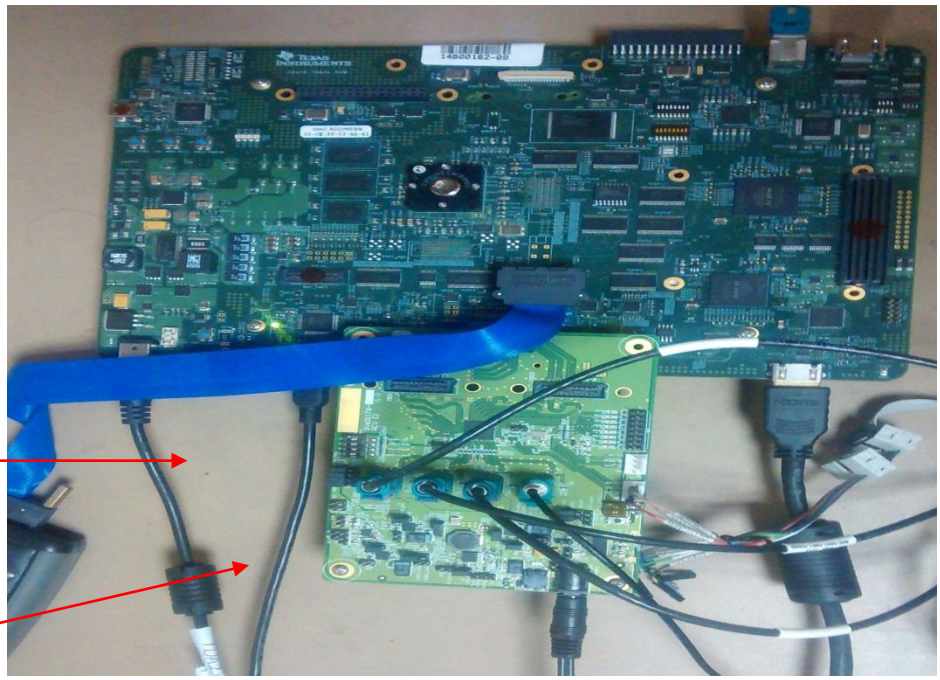


Figure 2.3.3 TDA3x EVM with UB960 Application Board

2.3.4 EDID Programming for HDMI Capture.

EDID information needs to be programmed on the EEPROM present on the EVM. This is required for the HDMI source to recognize the format and resolution supported by the receiver (TDA3xx SoC). If this step is not done or if this step fails, then TDA3xx SoC will not be able to receive data via HDMI.

It's recommended to program the HDMI receivers EDID. The default EDID is programmed to receive 1080P60 video streams only. If stream of different resolution is required (or EDID is corrupted), the EDID would require an update. The following steps details the procedure to re-program the EDID.

1. Change pins 1 and 2 of SW8000 dip switch (refer Figure 2.3.4) to ON.
2. Connect USB cable from board to PC and setup UART for logs (Ref Uart settings)
3. Connect CCS to IPU1-0 core and load below binary (Refer Load using CCS till step 8).

   Binary placed under
   vision_sdk\docs\edid\edid_programmer_tda3x_1080p_60.xem4
4. Run the core and wait till "EDID programming success full" message comes on UART.
5. Terminate the JTAG session and restart the board.
6. Before Running Vision Sdk binaries, change pins 1 and 2 of SW8000 to OFF (towards numbers 1 and 2).

HDMI CONFIG	1	2	3	4
HDMI capture without EDID	ON	OFF	ON	OFF
To do EDID programming	ON	ON	OFF	OFF
HDMI capture with EDID programming	OFF	OFF	ON	OFF

Figure 2.3.4 HDMI Config via SW8000 dip switch

NOTE: Refer Load and Run using CCS for details of running the binaries

2.4 Required H/W modification / Configurations

2.4.1 TDA3XX EVM Modifications for SCH usecase

I2C transactions for few sensors like OV10640 fail at 400KHz I2C frequency. To fix this issue, few resistors need to be updated/changed on the TDA3xx EVM. Please refer to the "Running BSP Application on TDA3XX EVM" section of the BSP user guide to get the more details

For OV10640 Parallel input, the IO Voltage required is 3.3V, but on TDA3xx EVM, 1.8V is provided by default. There could be data loss or I2C transfer failures because of this incorrect IO voltage. Change the IO voltage to 1.8V from 3.3V by removing resistor RJ26 from pins 1-2 and installing it on pins 2-3 on TDA3xx EVM.

2.4.2 Changes required on UB960 Application board

Mode

1. Configure to supply 9V on the LVDS lines to TIDA00262 modules
    a. J14 Short pins 1-2 as show in Figure 2.3.3
2. Configure UB960 to operate in LVDS in 75 MHz mode
    a. Dip Switch MODE 3 should be ON

1	□	
2	□	
3		□
4	□	

2.5 Supported Sensors

The Table below lists the sensors supported on TDA3xx platform. Please refer Figure 2.3.1 / Figure 2.3.2, this details the interfaces that a sensor could be connected to.

Sensor	Details	Comments
OV10635	Sensors OV10635 Sensor (for VIP based use-cases only), should be connected to Omni Vision interface	ISP is not used for this sensor.
OV10640 MIPI (CSI2)	OV10640 CSI2 Leopard imaging (LI) sensor on **LI Camera CSI2 Interface** (for ISS use-cases only)	It outputs RAW 12 in bayer format and ISP is used to convert to required format.
OV10640 (Parallel)	Parallel Leopard imaging (LI) sensor **LI Camera Interface** (for ISS use-cases only)	
AR0132	Aptina parallel imaging sensor on **Aptina Camera Interface** (for ISS use-cases only)	The ISP is used to convert from RAW12 to required format. ISP is not tuned for all lighting conditions for this sensor. ISP runs with the fixed set of configuration for this sensor, which may not be good for all conditions.
AR0140 (Parallel)	Parallel Leopard imaging (LI) sensor **LI Camera Interface** (for ISS use-cases only)	The ISP is used to convert from RAW12 to required format. Algorithms (AWB, AGC) is tuned for this sensor
TIDA00262	Connected to UB960 EVM, this module has AR0140 AT sensor and 913 Serializer. Please refer Figure 2.3.1	The ISP is used to convert from RAW12 to required format. Algorithms (AWB, AGC) is not yet tuned for this sensor.

Table below lists the different features supported by each sensor for the ISS case.

Features\Sensor	OV10640 CSI2	OV10640 Parallel	AR0140 Parallel	TIDA00262	AR0132 Parallel	IMX224 CSI2
Linear Mode			√	√	√	√
Two Pass WDR Mode	√	√	√			√
Single Pass WDR Mode			√			
Auto Exposure			√	√		√
Auto White	√	√	√	√		√

Balance						
DCC Support	√	√	√	√		√

2.6 Software Installation

vision_sdk_xx_xx_xx_xx_setupwin32.exe is the SDK package installer.

Copy the installer to the path of your choice.

Double click the installer to begin the installation.

Follow the self-guided installer for installation.

IMPORTANT NOTE: On some computers running as administrator is needed. Right click on the installer and select option of "Run as administrator". If this is not done then you may see a message like "This program might not have installed correctly

On completion of installation a folder by name VISION_SDK_xx_xx_xx_xx would have been created in the installation path.

2.6.1 Starterware ISS add on package installer

Starterware ISS add-on package is not included in Vision SDK installer, please contact TI local support for this package. Starterware ISS add-on package can be installed on top of Vision SDK installation, this add-on package supports WDR and LDC sub-blocks of ISS. Follow the below steps to install

- Copy the installer starterware_iss_add_on_setupwin32_xx.xx.xx.xx.exe to the path of your choice.
- Double click the installer to begin the installation.
- Enter the path install path as vision_sdk_install_dir\ti_components\drivers\
- Follow the self-guided installer for installation to complete.
- Once installation completes please check "**wdrldc**" folder is present in VISION_SDK_XX_XX_XX_XX\ti_components\drivers\starterware_XX_XX_XX_XX\vpslib\hal\src\

NOTE: Even though source for WDR and LDC not included in installer package, user can use prebuilt binaries in which WDR and LDC is enabled.

2.6.2 Uninstall Procedure

To uninstall, double click on uninstall.exe created during installation in the folder VISION_SDK_xx_xx_xx_xx.

At the end of uninstall, VISION_SDK_xx_xx_xx_xx folder still remains. It is just an empty folder. It can be deleted manually.

3 Build and Run

This chapter provides a brief overview of the sample application or use case present in the SDK and procedure to build and run it.

3.1 Overview of application in release

The Vision SDK supports the following use-cases are grouped under following categories

- Single Camera Use-cases
- Multi-Camera LVDS Use-cases
- AVB RX Use-cases, (TDA2x & TDA2Ex ONLY)
- Dual Display Use-cases, (TDA2x EVM ONLY)
- ISS Use-cases, (TDA3x ONLY)
- Stereo Use-cases, (TDA2x MonsterCam ONLY)
- Network RX/TX Use-cases
- Fast boot ISS capture + display (TDA3x ONLY)*

  ** Not listed in Runtime Menu*

Refer to VisionSDK_DataSheet.pdf for detailed description of each category.

The demos support devices listed in section 2.5 as capture source and HDMI 1080P60 can also be used as a capture source.

The demos support following devices as display devices

- LCD 7-inch 800x480@60fps
- LCD 10-inch 1280x720@60fps
- LCD 10-inch 1920x1200@60fps
- HDMI 1080p60 (default)

Use option "s" on the main menu in UART to select different capture and display devices.

3.2 Building the application

On windows command prompt, go inside the directory VISION_SDK_xx_xx_xx_xx\vision_sdk.

Build is done by executing gmake.

"gmake" is present inside XDC package. For "gmake" to be available in windows command prompt, the XDC path must be set in the windows system path.

IMPORTANT NOTE: xdc path is needed to be set in environment variables. If not, then set it using the set PATH = <Install_dir>/ti_components/os_tools/windows/xdctools_x_xx_xx_xx;%PATH% in command prompt

NOTE: To build for TDA3xx platform, set VSDK_BOARD_TYPE := TDA3XX_EVM in vision_sdk\Rules.make

Under vision_sdk directory run the command

```
> gmake -s all
```

Executing above will automatically clean up previous builds. It will then build all the necessary components (Starterware, BSP drivers, EDMA drivers) and then the Vision SDK framework and examples.

On a successful build completion, following executables will be generated in the below path

```
\vision_sdk\binaries\vision_sdk\bin\tda3xx-evm
 vision_sdk_arp32_1_release.xearp32F
 vision_sdk_c66xdsp_1_release.xe66
 vision_sdk_c66xdsp_2_release.xe66
 vision_sdk_ipu1_0_release.xem4
 vision_sdk_ipu1_1_release.xem4
```

If need be, incremental build can be done by following command

```
> gmake -s
```

To speed up the incremental builds the following can be done as required

The number of processors included in the build can be changed by modifying below values in Rules.make. A value of "no" means CPU not included in build, value of "yes" means CPU included in build. Below is the snippet of Rules.make where need to be edited for TDA3xx

```
# Some CPU's and module's force disable if build is for TDA3xx
#
# Some modules and core's are disabled since they are not
# present in TDA3xx, like VPE, IVAHD, A15_0, EVE2/3/4
#
# Enable some module's if they present only in TDA3xx, like ISS
#
ifeq ($(PLATFORM), tda3xx-evm)

# Disabling or enabling below CPUs/modules for now until they are tested
PROC_DSP1_INCLUDE=yes
PROC_DSP2_INCLUDE=yes
PROC_EVE1_INCLUDE=yes
  .
  .
  .
```

The build time can be made faster by following below steps.
BUILD_DEPENDANCY_ALWAYS = no in Rules.make

NOTE: If Starterware ISS add on package installer is installed as mentioned in Section 2.4.1 than user can build for WDR and LDC feature. Make change in Rules.make "WDR_LDC_INCLUDE=yes". Above is the snippet of Rules.make where need to be edited for TDA3xx.

After this gmake -s will incrementally build and build time will be faster.

However make sure to do "gmake -s depend" before "gmake -s" in the following cases.

- when number of processors enabled is changed in Rules.make
- when DDR_MEM value in Rules.make is changed
- when NDK_PROC_TO_USE to use is changed
- when any .h or .c file in TI component is installed in ti_components is changed
- when any new TI component is installed in ti_components

If this "gmake -s depend" not done then build or execution may fail.

Cleaning the build can be done by following command

```
> gmake –s clean
```

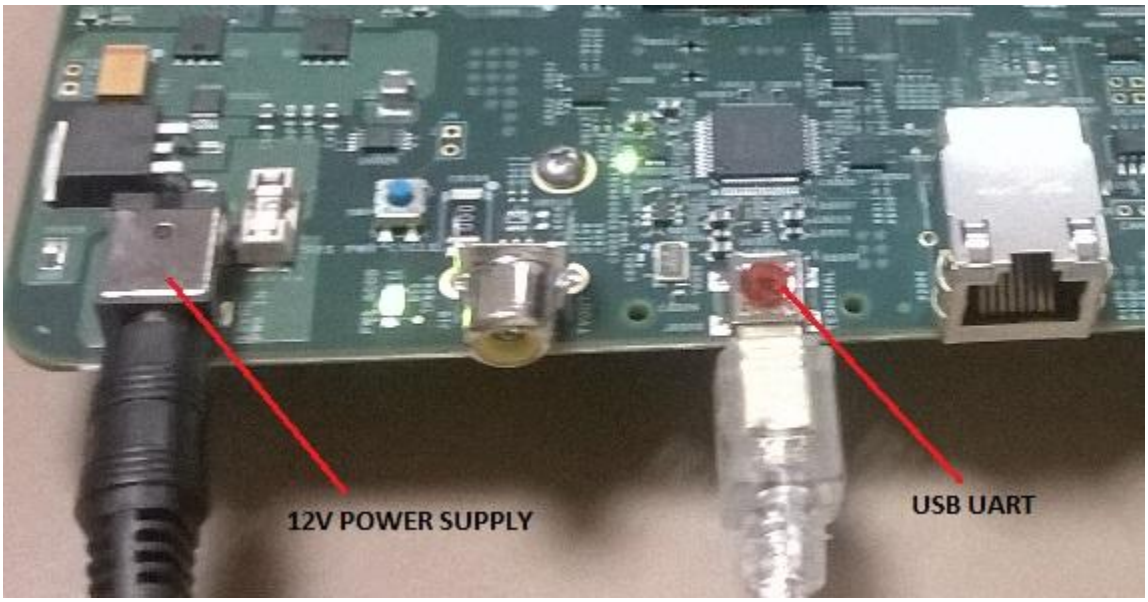
3.3 UART settings

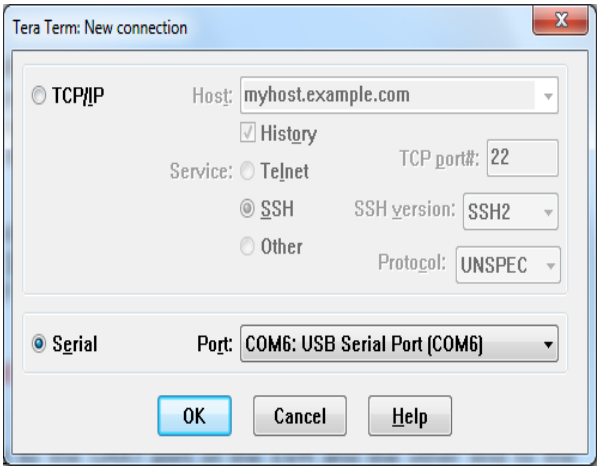
Connect a serial cable to the UART port of the EVM and the other end to the serial port of the PC (*configure the HyperTerminal at 115200 baud rate*) to obtain logs and select demo. EVM it detects 4 UART ports, you need to select the 3rd one.

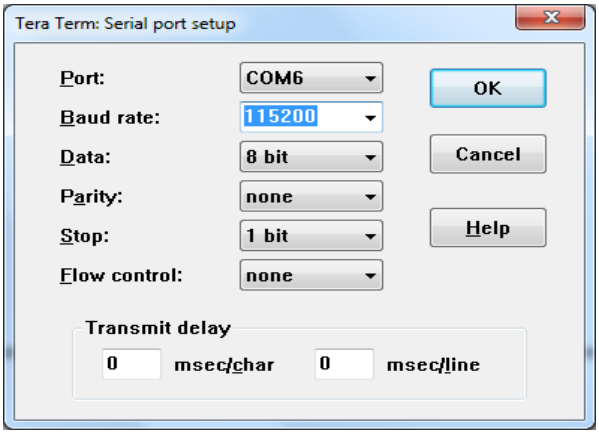
IMPORTANT NOTE: On some EVMs we were observing that UART terminal does not work. Updating the USB to UART driver on PC made UART work on the failings PCs. You can download the drivers from the below link.

http://www.ftdichip.com/Drivers/VCP.htm

http://www.ftdichip.com/Drivers/CDM/CDM%20v2.10.00%20WHQL%20Certified.exe







3.4 Load using QSPI

3.4.1 Steps to generate qspi writer tools

NOTE: SBL qspi image is built from starterware package.
To build qspi Run the command **gmake –s sbl_qspi** from vision_sdk root dir
And run the **sbl_qspi_create_tda3xx.bat** placed at vision_sdk\build\scripts
This generates all required tools under vision_sdk\build\scripts\qspi_tda3xx

1. qspiFlashWriter_m4_release.xem4
2. sbl_qspi

IMPORTANT NOTE: "sbl_qspi_create_tda3xx.bat" requires GCC tools need to be installed in "<install dir>/ti_components/cg_tools/windows/gcc-arm-none-eabi-4_7-2013q3" location. Tool can be downloaded from below link.

https://launchpad.net/gcc-arm-embedded/+milestone/4.7-2013-q3-update

3.4.2 Steps to generate appImage

Following steps need to be followed to generate the application image

1. Make sure the executables are built as shown in Building the application
2. To generate the application image run the batch file shown below
   `vision_sdk\MulticoreImageGen_tda3xx.bat`

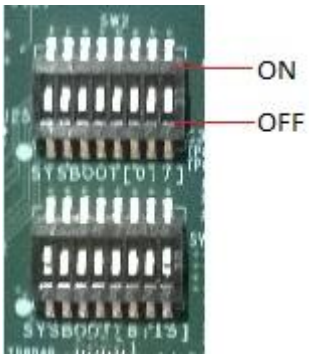
IMPORTANT NOTE: If some cores are disabled from build, comment them from MulticoreImageGen_tda3xx.bat and generate the AppImage.

REM is the comment used to comment in .bat file

REM set App_IPU1_CPU1 is sufficient

3.4.3 Flashing steps

Flashing pin settings:



- Please refer **Boot Modes of SBL** section In **SBL_UserGuide.pdf**

  (VISION_SDK_xx_xx_xx_xx\ti_components\drivers\starterware_xx_xx_xx_xx\bootloader)

For loading binaries using CCS refer Load using CCS till step 8.

1. Connect M4 (IPU)

Load image on M4

C:\VISION_SDK_XX_XX_XX_XX\vision_sdk\build\scripts\qspi_tda3xx\qspiFlashWriter_m4_release.xem4

Run the core.

Console outputs

QSPI Flash writer application

Enter Device type to use

1 - Spansion 1 bit

2 - Spansion 4 bit

3 - Micron 1 bit

4 - Micron 4 bit

5 - Winbond 1 bit

6 - Winbond 4 bit

7 - ISSI 1 bit

8 - ISSI 4 bit

*Select appropriate Device Type, for TDA3x EVM, press '**2**'.*

[Cortex_M4_IPU1_C0]

QSPI Flash writer application

MID - 1

DID – 18

Enter 0 for Erase-Only (without flashing any image)

Note : File size should be less than 33554432 Bytes.

Enter the file path to flash:

C:\VISION_SDK_XX_XX_XX_XX\vision_sdk\build\scripts\qspi_tda3xx\sbl_qspi

Enter the Offset in bytes (HEX) **0x00**

Erase Options:

----

0 -> Erase Only Required Region

1 -> Erase Whole Flash

2 -> Skip Erase

Enter Erase Option: **1**

Load Options:

----

0 -> fread using code (RTS Library)

1 -> load raw using CCS (Scripting console)

Enter Load Option: **0**

Read xxxxxx bytes from [100%] file...Done.

QSPI whole chip erase in progress

QSPI file write started

*************QSPI flash completed sucessfully***************

2. Reset the board and Repeat step 1.

[Cortex_M4_IPU1_C0]

QSPI Flash writer application

Enter Device type to use

1 - Spansion 1 bit

2 - Spansion 4 bit

3 - Micron 1 bit

4 - Micron 4 bit

5 - Winbond 1 bit

6 - Winbond 4 bit

7 - ISSI 1 bit

8 - ISSI 4 bit

*Select appropriate Device Type, for TDA3x EVM, press '**2**'.*

[Cortex_M4_IPU1_C0]

QSPI Flash writer application

MID - 1

DID - 18

Enter the File Name

C:\VISION_SDK_XX_XX_XX_XX\vision_sdk\build\scripts\qspi_tda3x x\sbl_qspi

Enter the Offset in bytes (HEX) **0x00**

Erase Options:

----

0 -> Erase Only Required Region

1 -> Erase Whole Flash

2 -> Skip Erase

Enter Erase Option: **2**

Load Options:

---

0 -> fread using code (RTS Library)

1 -> load raw using CCS (Scripting console)

Enter Load Option: **0**

Read xxxxxx bytes from [100%] file...Done.

QSPI file write started

*************QSPI flash completed sucessfully***************

3. Reset the board and Repeat step 1.

NOTE: If flashing binaries for fast boot usecase then the file name and offsets will be different and this step needs to be done twice, once for UCEarly, once for UCLate. Refer section 3.9 for steps to generate binaries, binary names, offsets

[Cortex_M4_IPU1_C0]

QSPI Flash writer application

Enter Device type to use

1 - Spansion 1 bit

2 - Spansion 4 bit

3 - Micron 1 bit

4 - Micron 4 bit

5 - Winbond 1 bit

6 - Winbond 4 bit

7 - ISSI 1 bit

8 - ISSI 4 bit

*Select appropriate Device Type, for TDA3x EVM, press '**2**'.*

[Cortex_M4_IPU1_C0]

QSPI Flash writer application

MID - 1

DID - 18

Enter the File Name

c:\VISION_SDK_XX_XX_XX_XX\vision_sdk\binaries\vision_sdk\bin\tda3xx-evm\sbl_boot\AppImage_BE

Enter the Offset in bytes (HEX): ***0x80000***

Erase Options:

---

0 -> Erase Only Required Region

1 -> Erase Whole Flash

2 -> Skip Erase

Enter Erase Option: **0**

Load Options:

---

0 -> fread using code (RTS Library)

1 -> load raw using CCS (Scripting console)

Enter Load Option: **1**

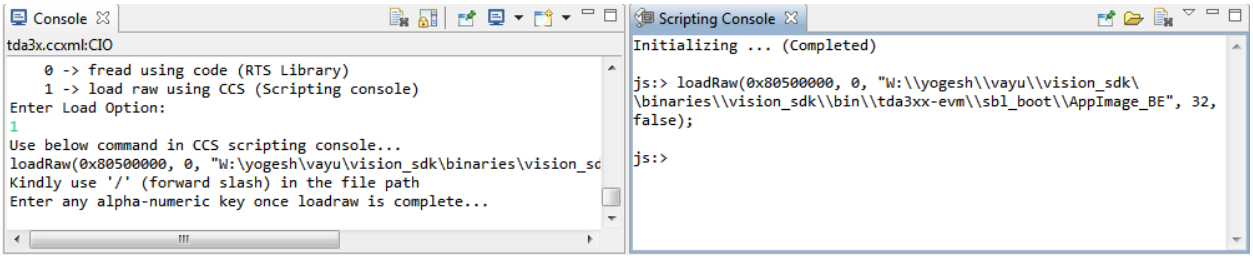
Open Scripting console window by clicking "Menu -> View -> Scripting console" and enter below command on scripting console as shown 3.5.3.1

loadRaw(0x80500000, 0, "C:/VISION_SDK_XX_XX_XX_XX/vision_sdk/binaries/vision_sdk/bin/tda3xx-evm/sbl_boot/AppImage_BE", 32, false);

IMPORTANT NOTE: The load address in loadRaw command could be different based on the board/SBL size etc. SBL figures out the address and prints it on CCS console. Use this address in loadRaw command for copying AppImage_BE.

In CCS console Enter any alpha-numeric key once loadraw is complete... as shown in 3.5.3.1

3.5.3.1 CCS console and scripting console



QSPI file write started

*************QSPI flash completed successfully***************

4. On completion change the pin setting as shown in above table

3.5 Load using QSPI and SD boot

In this mode SBL boots from QSPI but AppImage boots from SD card. This allows us to flash SBL once to QSPI and subsequently we can boot new AppImage just by copying AppImage to SD card.

3.5.1 Steps to generate qspi writer tools

NOTE: SBL qspi_sd image is built from starterware package.

To build qspi_sd Run the command **gmake –s sbl_qspi_sd** from vision_sdk root dir

And run the **sbl_qspi_sd_create_tda3xx.bat** placed at
vision_sdk\build\scripts
This generates all required tools under
vision_sdk\build\scripts\qspi_sd_tda3xx
1. qspiFlashWriter_m4_release.xem4
2. sbl_qspi_sd

IMPORTANT NOTE: "sbl_qspi_sd_create_tda3xx.bat" requires GCC tools need to be installed in "<install dir>/ti_components/cg_tools/windows/gcc-arm-none-eabi-4_7-2013q3" location. Tool can be downloaded from below link.

https://launchpad.net/gcc-arm-embedded/+milestone/4.7-2013-q3-update

3.5.2 Steps to generate appImage

Following steps need to be followed to generate the application image

3. Make sure the executables are built as shown in Building the application
4. To generate the application image run the batch file shown below
   `vision_sdk\MulticoreImageGen_tda3xx.bat`

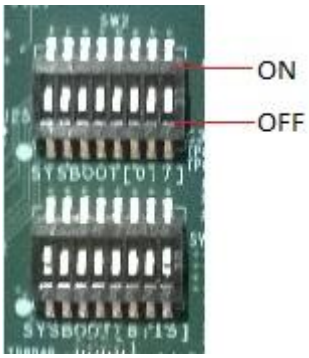
IMPORTANT NOTE: If some cores are disabled from build, comment them from MulticoreImageGen_tda3xx.bat and generate the AppImage.

REM is the comment used to comment in .bat file

REM set App_IPU1_CPU1 is sufficient

3.5.3 Flashing steps

Flashing pin settings:



- Please refer **Boot Modes of SBL** section In **SBL_UserGuide.pdf**

(VISION_SDK_xx_xx_xx_xx\ti_components\drivers\starterware_xx_xx_xx_xx\bootloader)

For loading binaries using CCS refer Load using CCS till step 8.

1. Connect M4 (IPU)

   Load image on M4

   C:\VISION_SDK_XX_XX_XX_XX\vision_sdk\build\scripts\qspi_tda3xx\qspiFlashWriter_m4_release.xem4

   Run the core.

   Console outputs

   [Cortex_M4_IPU1_C0]
   QSPI Flash writer application
   Enter Device type to use
   1 - Spansion 1 bit
   2 - Spansion 4 bit
   3 - Micron 1 bit
   4 - Micron 4 bit
   5 - Winbond 1 bit
   6 - Winbond 4 bit
   7 - ISSI 1 bit
   8 - ISSI 4 bit
   *Select appropriate Device Type, for TDA3x EVM, press '**2**'.*

   [Cortex_M4_IPU1_C0]
   QSPI Flash writer application
   MID - 1
   DID – 18
   Enter 0 for Erase-Only (without flashing any image)
   Note : File size should be less than 33554432 Bytes.
   Enter the file path to flash:
   C:\VISION_SDK_XX_XX_XX_XX\vision_sdk\build\scripts\qspi_tda3xx\sbl_qspi_sd
   Enter the Offset in bytes (HEX) **0x00**
   Erase Options:
   ----
   0 -> Erase Only Required Region
   1 -> Erase Whole Flash
   2 -> Skip Erase
   Enter Erase Option: **1**

   Load Options:
   ----
   0 -> fread using code (RTS Library)
   1 -> load raw using CCS (Scripting console)
   Enter Load Option: **0**

   Read xxxxxx bytes from [100%] file...Done.
   QSPI whole chip erase in progress

QSPI file write started

*************QSPI flash completed sucessfully***************

NOTE: User needs to copy the AppImage to root folder in SD card and insert SD card and power on EVM to boot it. SD card should be formatted as FAT32 with 512 bytes per sector.

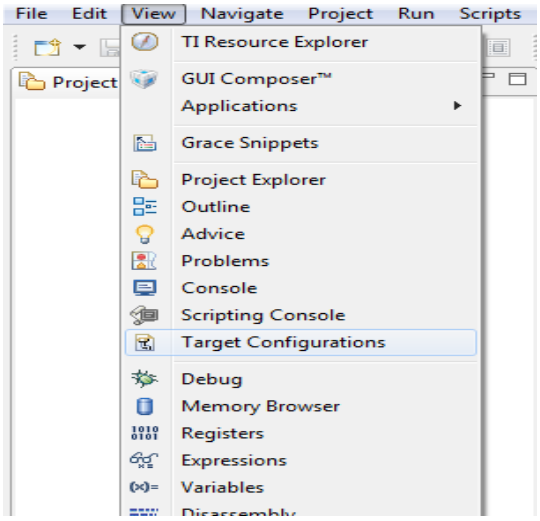
3.6 Load using CCS

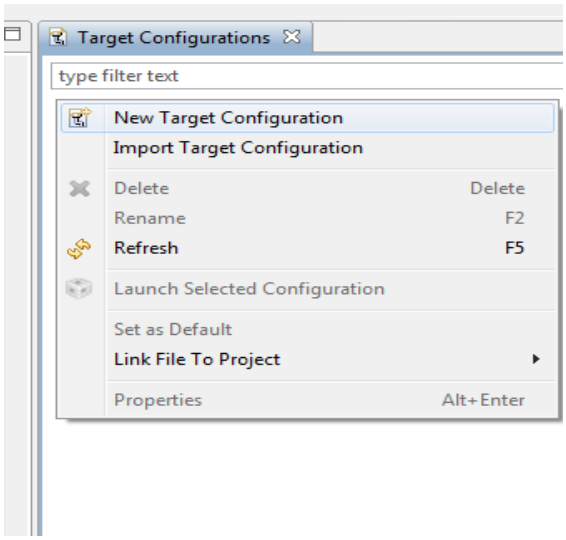
After installing CCS, follow below steps to complete the platform setup,

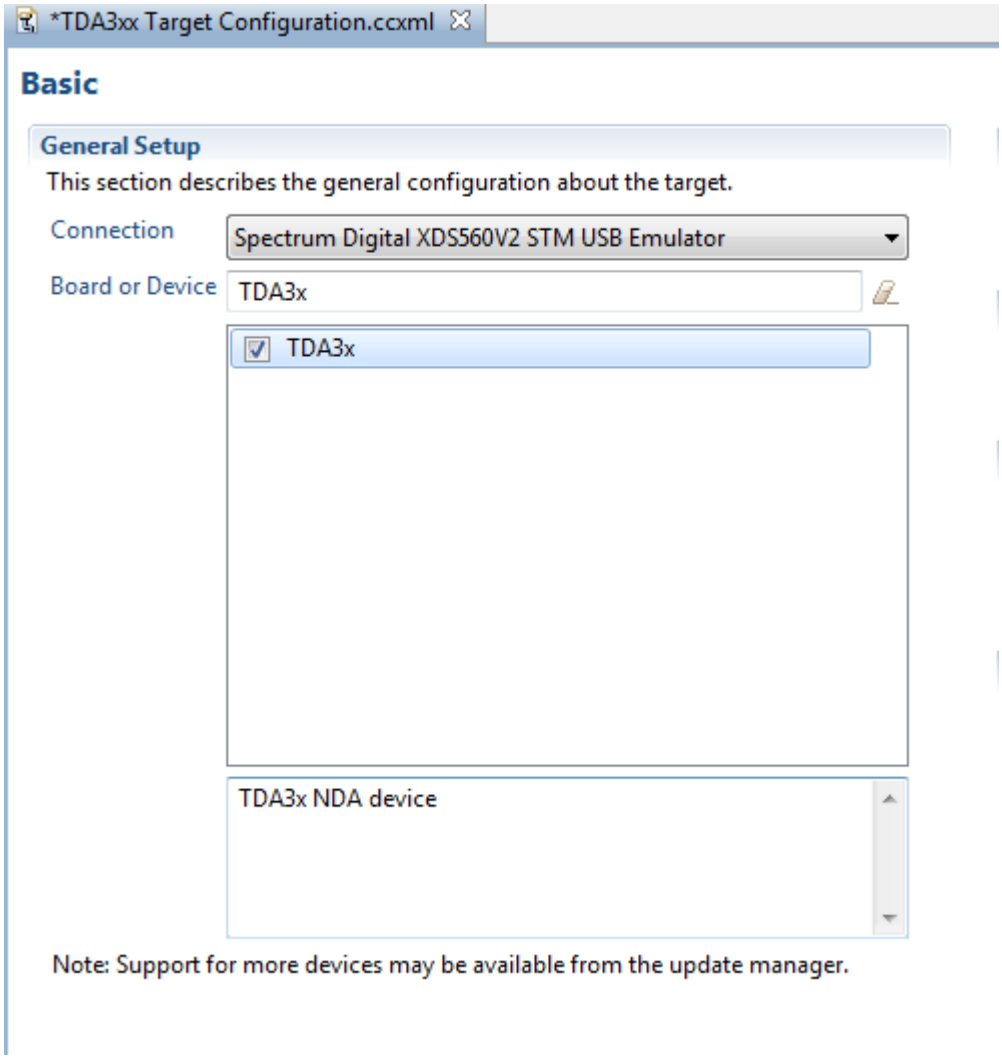
1. Install Chip Support Package from below location
   <Install_dir>\ti_components\ccs_csp\tda3xx\CCS_CSP_TDA3x_SR1.0_NDA_TRM_vB_gels4.zip
   NOTE: Follow the install guide in CSP to install it.
   In CCS check for Software update and update the TI Emulators and spectrum digital emulators.

   Change the following GEL files for vision SD as below,
   - TDA3xx_multicore_reset.gel
     - Set VISION_SDK_CONFIG to 1
     - 256MB mode not supported

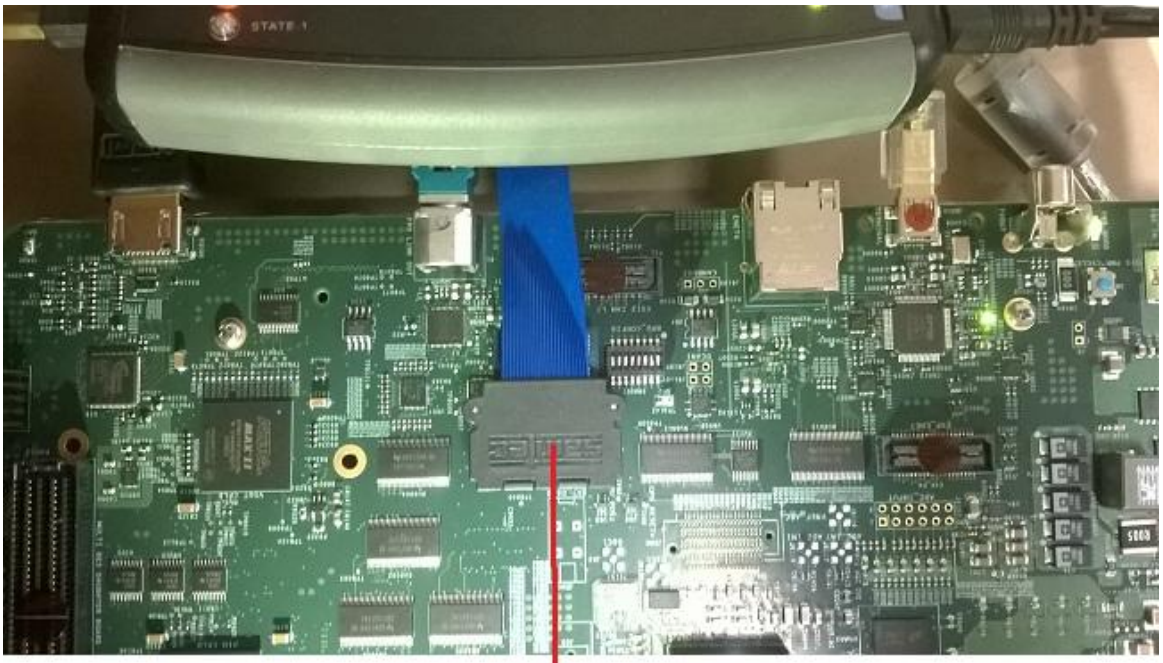
2. CCS Target Configuration creation:
   a. Open "Target Configurations" tab, by navigating through the menu "View ->Target Configurations".
   b. Create a new Target Configuration (TDA3xx Target Configuration) by navigating through the menu "File->New->Target Configuration File".
   c. Specify Connections as "Spectrum Digital XDS560V2 STM USB Emulator".
   d. Specify Board or Device as "TDA3x".



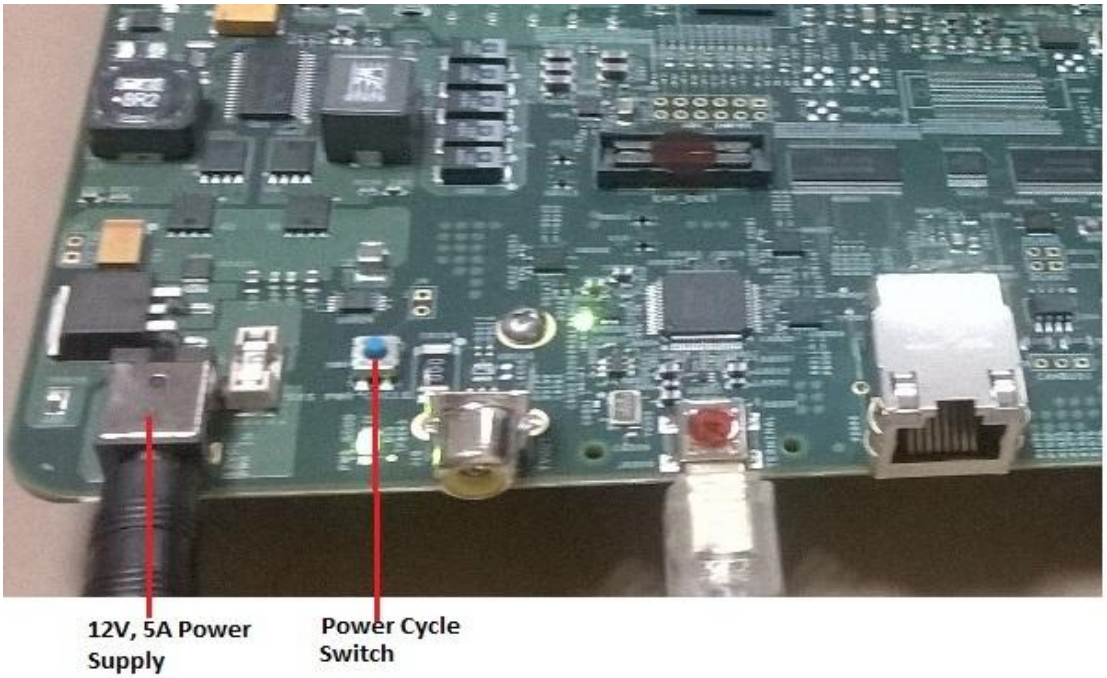

Target Configurations
type filter text
New Target Configuration
Import Target Configuration
Delete Delete
Rename F2
Refresh F5
Launch Selected Configuration
Set as Default
Link File To Project
Properties Alt+Enter


*TDA3xx Target Configuration.ccxml
Basic
General Setup
This section describes the general configuration about the target.
Connection
Spectrum Digital XDS560V2 STM USB Emulator
Board or Device
TDA3x
TDA3x
TDA3x NDA device
Note: Support for more devices may be available from the update manager.

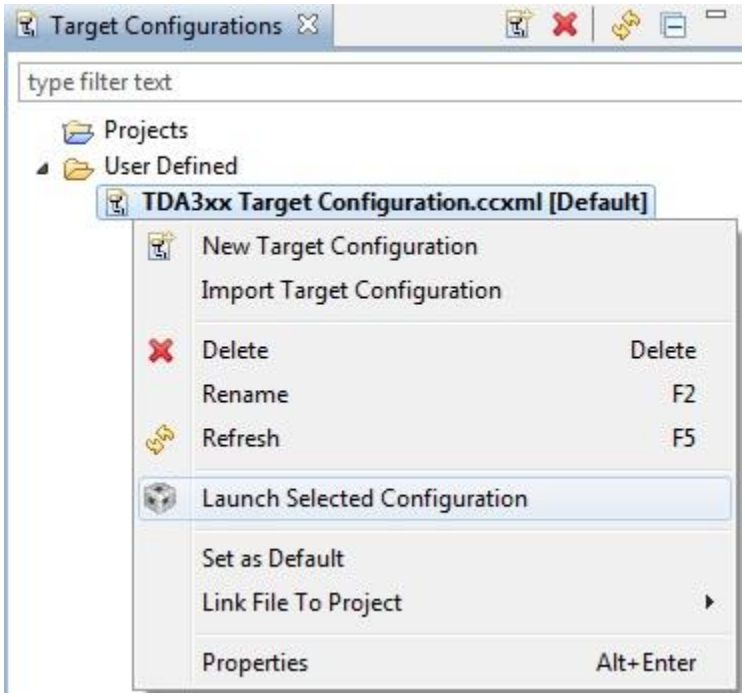
3. Connect JTAG to the board.

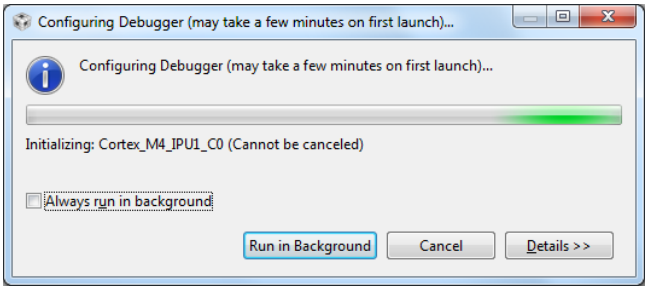


4. Reset EVM through the power recycle button.

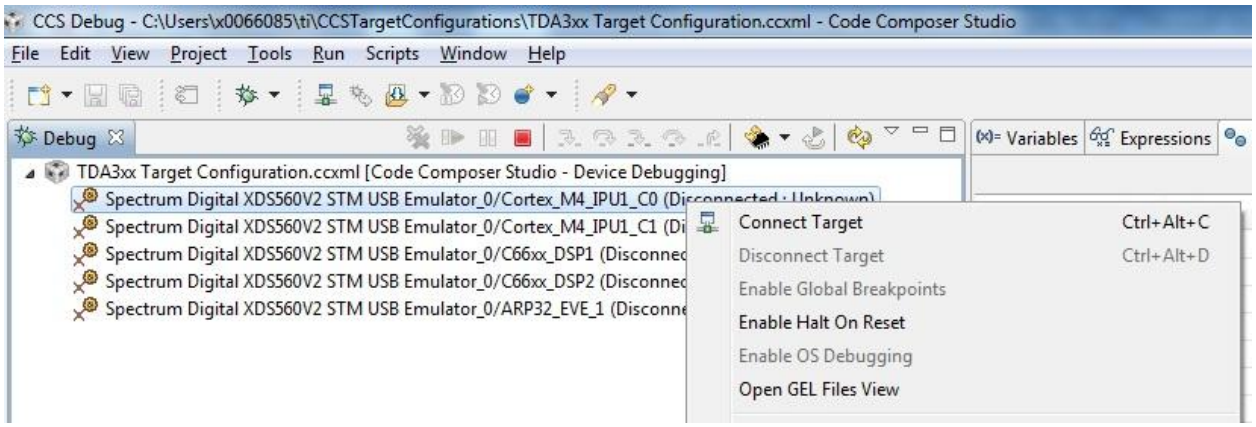


5. Now launch the previously created TDA3xx Target Configuration.





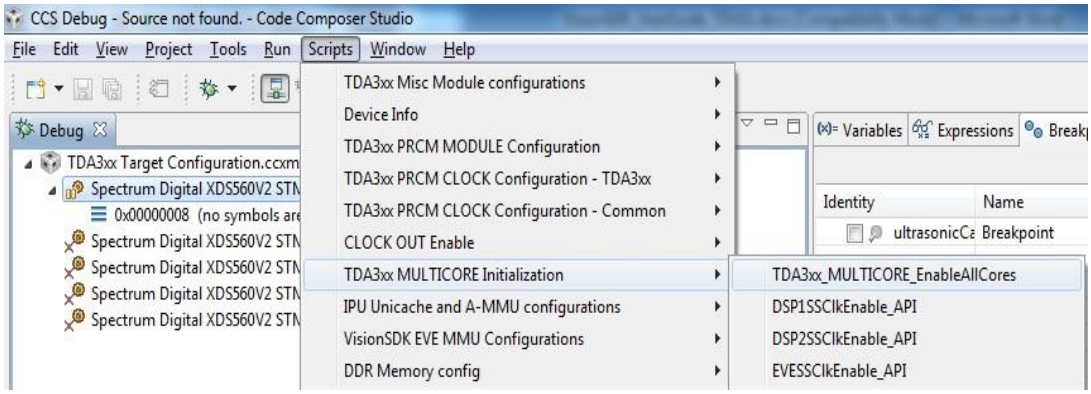
6. Connect to core Cortex_M4_IPU1_C0.



7. On successful connect, the following log appears on CCS console:

```
Cortex_M4_IPU1_C0: GEL Output: --->>> TDA3xx Target Connect Sequence DONE
!!!!!  <<<---
```

8. Select Cortex_M4_IPU1_C0, navigate to Scripts->TDA3xx MULTICORE Initialization TDA3xx_MULTICORE_EnableALLCores

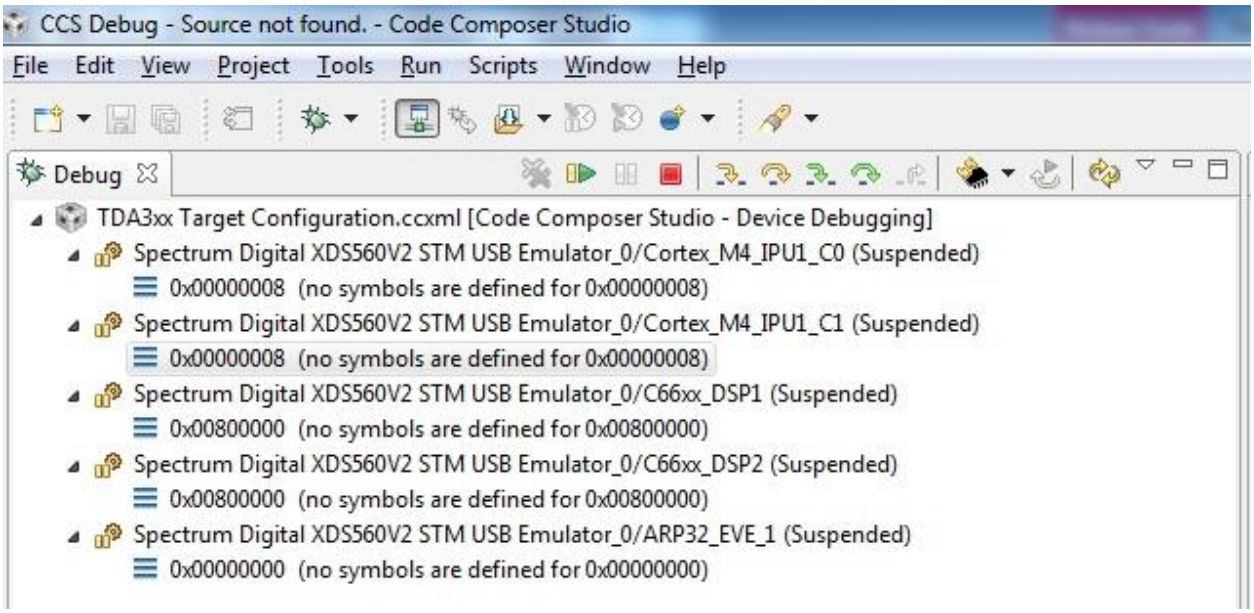


9. On successful script execution, the following log appears on CCS console:

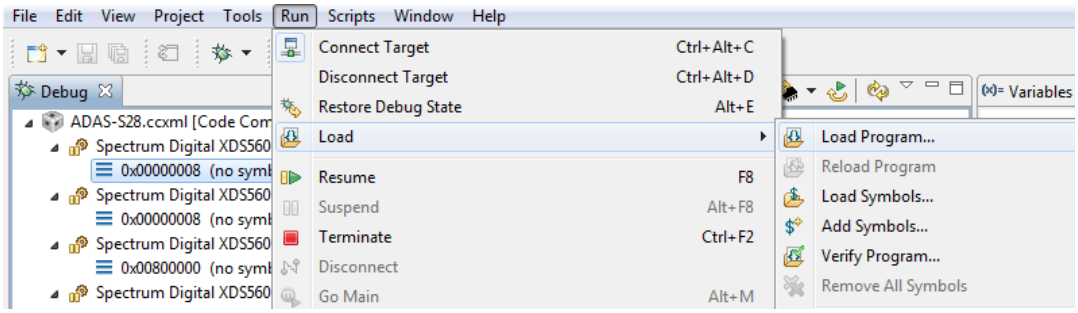
```
Cortex_M4_IPU1_C0: GEL Output: --->>> EVESS Initialization is DONE! <<<---
```

10. Now connect the core shown below,
    ARP32_EVE_1, C66xx_DSP1, C66xx_DSP2 and Cortex_M4_IPU1_C1.

11. Once the cores are connected, do CPU Reset for all the cores.



12. On the cores load the binaries as mentioned below



On ARP32_EVE_1, load the binary, “vision_sdk_arp32_1_release.xearp32F”.
On C66xx_DSP2, load the binary, “vision_sdk_c66xdsp_2_release.xe66”.
On C66xx_DSP1, load the binary, “vision_sdk_c66xdsp_1_release.xe66”.
On Cortex_M4_IPU1_C0, load the binary, “vision_sdk_ipu1_0_release.xem4”.
On Cortex_M4_IPU1_C1, load the binary, “vision_sdk_ipu1_1_release.xem4”.

IMPORTANT NOTE: Binary for Cortex_M4_IPU1_C0 MUST be loaded before Cortex_M4_IPU1_C1 since IPU1-0 does MMU config for the complete IPU1 system. Other binaries can be loaded in any order.

3.7 Run the demo

1. Power-on the Board after loading binaries by (SD, QSPI, NOR or CCS) and follow Uart settings to setup the console for logs and selecting demo.

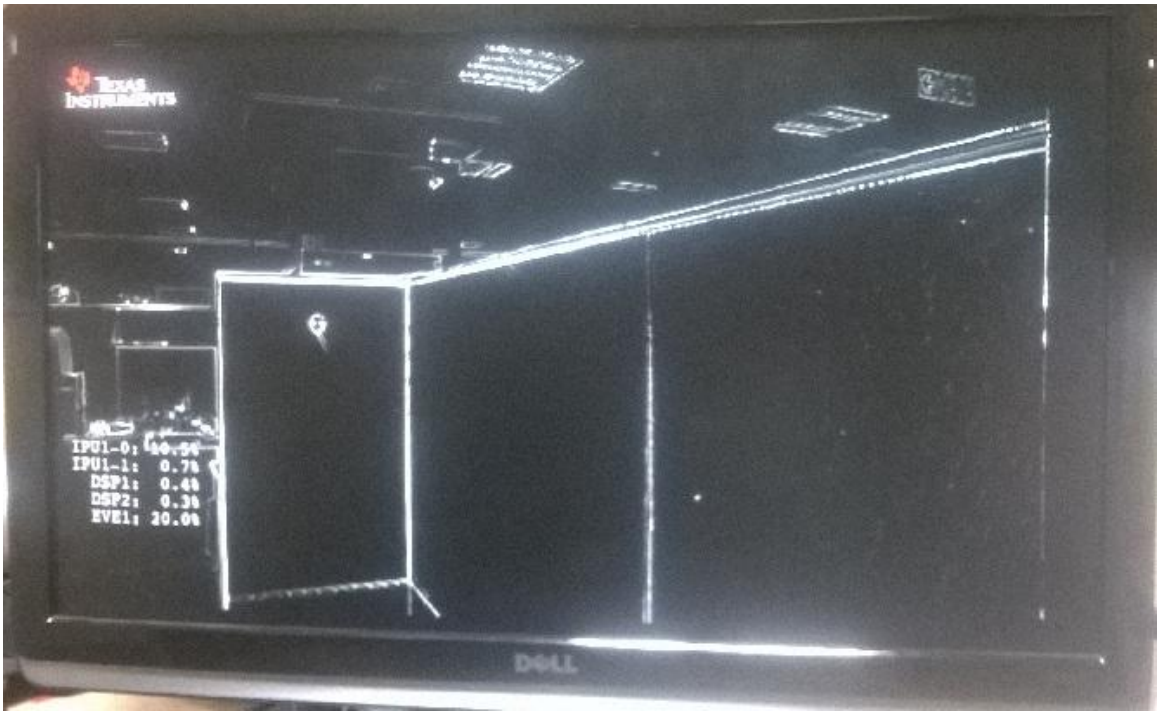
2. Select demo required from the menu by keying in corresponding option from the uart menu.

IMPORTANT NOTE: Make sure you select SCV (1Ch VIP capture) use-case or ISS use-case depending on the camera that is connected and supported

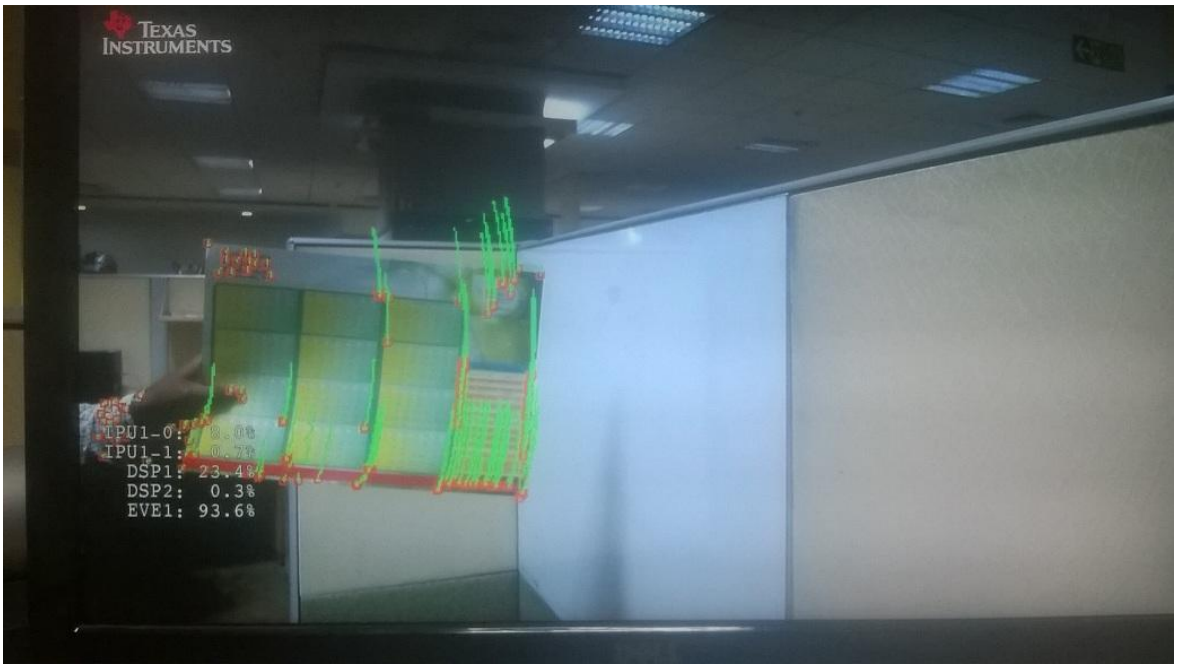
After successful initialization of the use-case, you will see video been display on the HDMI as shown below,

a. SCV use-cases:

b. EDGE Detect use-case:



c. Sparse optical flow usecase



3.8 DCC

Dynamic Camera Configuration (DCC) tool is a PC based tool suit that is primarily used for offline tuning of raw images obtained from raw camera sensors connected to ISS hardware. Apart of tuning tool, DCC also contains ISP simulator.

NOTE: DCC tool can be downloaded from the below CDDS link. DCC version 2.1, compatible with the Vision SDK 2.8 release, should be installed. Please contact local TI FAE to get access to this CDDS link.

https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.33350.26722

DCC tool is dependent on matlab runtime libraries, refer to the DCC userguide and install required MatLab runtime.

DCC tuning tool with help of plug-ins generate a set of DCC XML and BIN files. BIN files contain tuned values in binary file format for various ISP modules. These binary files for different ISP modules are merged into single binary file and used by the sensor driver in the BSP.

The binary file can also be flashed in the QSPI. Network tool command iss_save_dcc_file can be used to save DCC binary file in the QSPI. Refer to the network tool documentation for more information on this command.

When ISS usecase is run in the vision sdk, it reads these binary files, parses them and applies to the ISP modules. Vision SDK first tries to use DCC binary file from the QSPI, if it is not available in the QSPI, it will use binary file from the BSP sensor driver. If the binary file is not available even in BSP sensor driver, it uses ISP default parameters from the video sensor layer in the Vision SDK.

IMPORTANT NOTE: DCC is currently supported only for AR0140, IMX224 and OV10640 sensors. For AR0140 sensor, the DCC xml files with the tuned ISP parameters can be found in the path ti_components\drivers\vayu_drivers\bspdrivers_\src\devices\ar0140\src\dcc_xml, for OV10640 driver, they can be found in the path ti_components\drivers\vayu_drivers\bspdrivers_\src\devices\ov10640\src\dcc_xml and for IMX224 driver, they can be found in the path ti_components\drivers\vayu_drivers\bspdrivers_\src\devices\imx224\src\dcc_xml .

DCC tuning tool exposes key parameters for each plugin which controls important tuning parameters. Users who are imaging and TI ISP experts can control/modify each parameter of plugin through XML files. Use below steps for updating and applying new tuned parameters.

- Update the DCC xml file for the ISP modules for the given sensor
- Convert xml file to binary file using dcc generator tool. This is a windows based tool to convert xml file to binary file. This tool can be found from the BSP_PATH\docs\tda3xx\dcc_gen_win.exe. This tool takes name of the xml file as an argument and generates the binary file from xml file at the same folder where xml file is stored.
    - Usage: dcc_gen_win.exe <Plugin's XML FILE >

- Send the binary file to the target using iss_send_dcc_file command of the network tool. Please refer to the network tool documentation for getting information on network tool

IMPORTANT NOTE: DCC xml to bin file convertor, dcc_gen_win.exe, is supported only on Windows platform. Also this tool is dependent on DCC GUI tool, so make sure that the gui tool is installed on the computer before using dcc_gen_win.exe executable. Please contact local TI support to get DCC GUI tool.

Once the tuned parameters are tested and finalized, they can be permanently stored in the QSPI or in driver.

For storing tuned parameters in the QSPI, run iss_save_dcc_file command of the network tool. Please refer to the network tool documentation for more information about this command. This command saves the binary file at the fixed offset in the QSPI. After saving the binary file in QSPI, restart the ISS usecase to check the output of the tuned parameters.

For storing tuned parameters in the sensor driver, go to the dcc_xml folder under sensor driver. For example, for AR0140 sensor, go to the BSP_PATH\src\devices\ar0140\src\dcc_xml\. This folder contains all the xml files that vision sdk is using for this sensor. Copy the updated the xml file under this folder and run generate_dcc.bat file from the windows. This batch file converts all xml files into binary files, merges all binary files to single binary file and converts binary file to header file, which will be used by the driver. After running this batch, restart the ISS usecase to check the output of the tuned parameters.

Below is list of the DCC plugins supported in vision sdk.

ISIF_CLAMP	DC Offset/Black Level offset in the ISIF This plugin is currently used only for setting blank level offset. No other parameters from this plugin are used.
IPIPE_GIC	Green Imbalance Correction Module of IPIPE
IPIPE_NF1	Noise Filter 1 module of the IPIPE
IPIPE_NF2	Noise Filter 2 module of the IPIPE
IPIPE_DPC_OTF	Defect correction OTF module
IPIPE_CFA	Color Filter Array module
IPIPE_Gamma	Gamma Correction module
IPIPE_RGB2RGB1	RGB to RGB color correction module-1 Supports multi photospace*, which means multiple set of parameters can be defined based on the photospace for this module.
IPIPE_3D_LUT	3D Lut module
IPIPE_RGB2RGB2	RGB to RGB color correction module-2 Supports multi photospace*

IPIPE_RGB2YUV	RGB to YUV Color Conversion module
IPIPE_EE	Edge Enhancer module
IPIPEIF_SPLIT	VP Decompanding module of the IPIPEIF
IPIPEIF_WDRMERGE	WDR Merge and WDR Companding module of the IPIPEIF Most of the WDR merge parameters are calculated on the fly based on the exposure ratio, so the only WDR merge parameter used from this plugin are enable and black level for long and short exposure.
GLBCE	GLBCE module
NSF3V	NSF3v module Supports multi photospace*
CNF	Chroma noise filter module Supports multi photospace*
AWB_ALG	AWB Calibration Parameters TI AWB algorithm requires these calibration parameters. If not provided, it uses default calibration parameters
LDC	Lens Distortion Correction Module

***** Photospace is defined by three parameters, exposure time, analog gain and color temperature. A range of these parameters creates one photospace. Refer the DCC Gui documentation to get more details on how to create photospace

* Although multiple photospace is supported only for few modules, xml files for almost all modules could have multiple set of parameters based on the multiple photospace. If the module does not support multiple photospace and xml file contains multiple set of parameters, only the first parameter set is used by the parser.

Updating Mesh LDC table in Vision SDK:

For the new fisheye lens, follow below steps to update the mesh LDC table

a. Get the new LDC table for the new Fisheye lens
b. Go to the Vision_sdk\tools\LDC_mesh_table_convert directory in the vision SDK
c. Run the perl script convert.pl as shown below

   perl convert.pl input_table.txt imagewidth imageheight downscalefactor

   Here inputtable.txt file contains mesh LDC table for the new lens

   imagewidth and imageheight are size of the input image

   downscalefactor is the down scale factor by which input table is down scaled.

d. It will generate the file input_table.bin file, convert this bin file to header file using bin2c BSP utility, it can be found under ti_components/drivers/vayu_drivers/bspdrivers_/docs/tda3xx

e. Replace this header file in vision_sdk\examples\tda2xx\src\usercases\common\chains_common_iss_ldc_table_02.h

f. Change the LDC table index to 1 while calling API ChainsCommon_SetIssSimcopConfig from chains_issIspSimcop_Display_SetSimcopConfig API in the file vision_sdk\examples\tda2xx\src\usecases\iss_capture_isp_simcop_display\chains_issIspSimcop_Display.c

g. Rebuild vision sdk

3.9 Fast boot usecase

This usecase is mainly targeted for rear view camera systems and mainly demonstrates how boot time can be optimized to show sensor capture ouput on display (preview) first on power on reset and then switch to analytics output shown on display.

As the execution sequence for this usecase is different than all other usecases, it is not enlisted in console RunTime Menu.

It is a fixed configuration demo usecase which works when you press reset button on the TDA3X EVM.

3.9.1 Usecase configuration

It supports following configuration _Only_

- 1CH ISS Capture + ISP + LDC + Obj detect + Display
  - Capture – OV10640P (Parallel interface) with TDA3x EVM
  - Display – 10 inch LCD
  - Boot mode - QSPI

3.9.2 Hardware set up

Refer section 2.4 "Required H/W modification / Configurations" to understand board modification needed for TDA3X with and above mentioned usecase configuration. It is important to have this done before fast boot usecase is tried. H/w mods for following can not be skipped.

- I2C to run at 400KHz
- Support for OV10640P

Figure: TDA3x EVM Fast boot h/w setup

3.9.3 Build

Important Note: Fast boot usecase needs Iss add on package to be installed first. Refer section 2.6.1 for more details.
pre-built binaries are kept under VISION_SDK_XX_XX_XX_XX\vision_sdk\prebuilt\tda3xx-evm\fast_boot, please note the usecase configuration mentioned in section 3.9.1

- This is special usecase demonstrating how boot time can be optimized for any vision_sdk usecase; idea here is to have preview display up in minimum possible time and then switch to actual usecase.
- The usecase is not enlisted in runtime menu it can be enabled using following variable in Rules.make under tda3xx-evm. By default it is "no".

```
ifeq ($(PLATFORM), tda3xx-evm)
...
FAST_BOOT_INCLUDE=yes
...
```

Remove DSP2 and IPU1_1 from the Rules.make & define WDR_LDC_INCLUDE to "yes"

```
PROC_DSP2_INCLUDE=no
PROC_IPU1_1_INCLUDE=no
NDK_PROC_TO_USE=none
WDR_LDC_INCLUDE=yes
```

Important Note: These can be defined 'yes' even in fast boot usecase but they are not needed for this usecase and can contribute to boot time hence removed from build config. User may enable these as per their usecases.

Important Note: In order to test the DSP and EVE analytics off and on options one must make sure to not include IPU1_1 in the build and the PROC_IPU1_1_INCLUDE should be 'no'

- Use gmake –s all to build, "gmake config" command can be used prior to build to confirm the build configuration.
- SBL also needs to be built for fast boot usecase
  - Ensure "gmake –s sbl_clean" is done if SBL was built previously
  - Use "gmake –s sbl_qspi" to build SBL

3.9.4 Generating and Flashing images

- Refer section 3.4.1 to generate sbl_qspi
- User following commands to build two AppImages

When building using Linux

```
$>cd <ROOTDIR>
$>chmod +x ./MulticoreImageGen_tda3xx_fast_boot.sh
$>./MulticoreImageGen_tda3xx_fast_boot.sh
```

When building using Windows

```
> MulticoreImageGen_tda3xx_fast_boot.bat
```

- This script should generate AppImages at \vision_sdk\binaries\vision_sdk\bin\tda3xx-evm\sbl_boot
  - AppImage_UcEarly_BE
  - AppImage_UcLate_BE
- Refer section 3.4.3 to flash following images at given offsets.

Image	QSPI offset to be flashed in
sbl_qspi	0x0
AppImage_UcEarly_BE	0x80000
AppImage_UcLate_BE	0xA80000

Important Note: Ensure images are flashed at given offsets only, order is not mandatory

3.9.5 Run

Press Power On Reset button on Tda3x EVM. Make sure QSPI boot is selected as mentioned in section 3.4

Pass criteria

- Display should flash up with preview in < 1 sec
- Usecase should switch to Object detect algorithm and Pedestrian / Traffic signs detection should start as soon as they are in field of view after boot up.

- You should see boot time printed on the LCD below the CPU performance bar.
- In order to run the analytics ON (option 3) and OFF (option 4) scenario one can choose to select any of the highlighted menu options. The display shows the status of the PD+ TSR Object detection above the CPU Performance bar.

```
1: Save Captured Frame
2: Save SIMCOP Output Frame
3: PD and TSR ON
4: PD and TSR OFF
```

Important Note: Ensure IPU1_1 image is not a part of the application image when trying these two options.

4 Frequently Asked Questions

4.1 Hardware Board Related FAQs

Q. I selected a use-case and it hangs during initialization
Make sure you are running the SCV use-case on SCV hardware setup and LVDS use-case on LVDS hardware setup. For LVDS hardware use 12V and 7A supply only.
Q. Sometimes I see a message "LCD not connected" on the UART console after running the use-case but I see normal display on the LCD
Ignore this message, the software / hardware is falsely reporting LCD is not connected.
Q. LVDS use-case init hangs on first try after power-cycle
Make sure all Sensor are connected as mentioned in earlier section. For LVDS hardware use 12V and 7A supply only.
Q. Sometimes LVDS setup hangs during use-case initialization second time after power-cycle
Suspecting this to be a board connectivity issue. Tighten the application / deserializer boards, Power cycle and retry. For LVDS hardware use 12V and 7A supply only.
Q. After CCS reload without power-cycle LVDS setup hangs during use-case initialization
Suspecting this to be a board connectivity issue. Power cycle, reset the board and retry.
Q. Sometimes application does not come to main menu after reloading from CCS
Power cycle before reloading application

4.2 Build, install, load, run related FAQs

Q. How do I speed up build and load time ?
IMPORTANT: Do "(g)make –s config" to check current build options. When Rules.make is changed, do "(g)make –s config" to see if the change will really take effect. One or more of the following steps can be used to speed up build time, 1. Edit Rules.make and only select the CPU that you are interested in Ex: If DSP1 is used, PROC_DSP1_INCLUDE is set equal to yes. If EVE1 is not used, PROC_EVE1_INCLUDE can be set equal to no. 2. Edit Rules.make and set BUILD_DEPENDANCY_ALWAYS=no around line ~490. When BUILD_DEPENDANCY_ALWAYS=no, an additional step of (g)make –s depend needs to called before (g)make –s when number of CPUs, or BSP/Starterware is changed. See comment in Rules.make above this line for

exact details. This will avoid checking and building dependancies every time (g)make –s is called and thus helps reduce build time

3. Invoke make with "-j" option. This will trigger parallel compilations of C files. A side effect of this is that is case any file has compilation error the error may get mixed with other valid C file compilations. Hence in case of any error make sure to scroll up in the console to find the error

One or more of the following steps can be used to speed up load time,

1. Edit algorithmLink_cfg.c (\vision_sdk\src\links_common\algorithm) can comment Alg Plugin init of algorithms not needed for current use-case. This will avoid linking of these algorithms and therefore reduce final binary size and hence load time. Specifically
   a. when surround view not being used, comment the following lines (this saves about 5MB per DSP)
      // AlgorithmLink_Synthesis_initPlugin();
      // AlgorithmLink_pAlign_initPlugin();
      // AlgorithmLink_gAlign_initPlugin();
      // AlgorithmLink_UltrasonicFusion_initPlugin();
   b. when stereo disparity is not required, comment the following lines (this saves 5MB on EVE2)
      // AlgorithmLink_RemapMerge_initPlugin();
2. Edit Rules.make to disable CPUs not needed as mentioned above, this will help reduce load time as well
3. Edit Rules.make to disable IVAHD "IVAHD_INCLUDE=no", this will help reduce IPU1-0 load time on TDA2x. This option should be used only if IVAHD MJPEG/H264 encode/decode is not required.

Q. If I am using only few cores in the device, can I build code only for them and how?

Yes, building for only the cores of interest can be done. Controlling this is present in file \vision_sdk\Rules.make,

Ex: If DSP1 is used, PROC_DSP1_INCLUDE is set equal to yes.
If EVE1 is not used, PROC_EVE1_INCLUDE can be set equal to no.

Do "gmake –s depend" followed by "gmake –s" after making the changes, to regenerate the binaries. Load and run only the required binaries via CCS/SD card

Q. How can I change build option from release to debug for a processor core?

In file \vision_sdk\Rules.make, there is an option to set build type for each processor core. For ex: For DSP1, PROFILE_c66xdsp_1 can be set as release as debug.

Q. I am using Windows 7 and when I build I get errors no permission to generate batch file?

Please check all the steps to build are followed , Try running as administrator .

Even if the issue exists there may be specific settings set for the PC.

There would be some makefile changes to be done.

CACLS $(DEST_ROOT)/maketemp_configuro_cmd_$(CORE).bat /E /P Everyone:F

To be replaced by

ICACLS $(DEST_ROOT)/maketemp_configuro_cmd_$(CORE).bat /grant Administrator:(OI)(CI)F

This is to be changed in rules_m4.mk rules_a15.mk rules_66.mk rules_arp32.mk
Q. Can the version of tools or other components be changed? If yes, how?
Please check with the support team if this release can work with versions other than the ones present in the release package. If yes, then component install path can be controlled via the file \vision_sdk\Rules.make. Paths for several tools and components used are provided in this file.
Q. Is Linux based build supported?
Linux based build can be supported, but it needs linux based compiler tools (GCC, M4, DSP, EVE cgtools) and XDC to be installed. Two installer packages are for released - one for windows with windows compiler tools and other for linux with linux compiler tools. Make sure to install the appropriate one depending on the machine you wish to compile on.
Q. In Surround View Demo I don't see proper display, whereas video from sensor shows proper data.
Yes , This issue is observed if previous Surround view demo calibration tables are not written completely to qspi memory. You have to erase table from the qspi memory and recalibrate . Option is available runtime in the same demo.
Q. Is there any script to automate loading and running of cores via CCS
Yes, this is possible via CCS based debug server scripting (DSS). Refer to a sample script file located at \vision_sdk\build\scripts\launch_visionsdk_tdaNxx.js. Refer to comments in this file for customizing as required.
Q. How to ensure alignment when sharing data structures across cores
When data structures are accessed across cores, it's a good practice to ensure all elements are 32-bit – this ensures structures are 32-bit aligned. Alternately when defining data-structure programmer can take care of alignment by declaring 32-bit first, then 16-bit and later 8-bit data. This will ensure no packing.
Q. EVE self-branch instruction block - EVE1_VECS (Important Note – refer mem_segment_definition_512mb.xs)
In SBL, EVE self-branch instruction is inserted @ 0x80000000 if no AppImage for EVE (if EVE is not enabled in the build). This could overwrite the code/data loaded at 0x80000000. So Reserve a small memory block in the beginning of the DDR @0x8000 0000 for EVE self-branch instruction if no AppImage for EVE. If EVE is enabled, then the EVE VECS/DATA/CODE is placed @0x8000 0000, and hence we did not observe any issue. If EVE is disabled, then also DO NOT removes this EVE1_VECS section @0x80000000, which is used by SBL to place EVE self-branch instruction.

4.3 PM and resource related FAQs

Q. How to Enable/Disable CPUIDLE functionality. How is it advantages using CPUIDLE.
CPUIDLE functionality helps in power saving, When the core is executing idle instruction the ip are put in sleep/Standby mode. **Please note that task sleep or**

any timer related calls will not work for EVE if PM is enabled
Q. I see the core is halted at IDLE/WFI instruction every time when I load the symbols in release mode
This is because of Power Management and CPUIDLE enabled in Vision SDK. The core executes idle/wfi instruction when idle. It wakes up on interrupt and hence debug the code in release mode becomes difficult. When binaries are built in debug mode CPUIDLE is disabled.
Q. How can I debug when binaries are in release mode. The application doesn't step when in WFI/IDLE looks like hung state.
If in case debugging release mode binary is essential then, CPUIDLE needs to be disables for eliminating the WFI/IDLE issue, CPU_IDLE_ENABLED=no in Rules.make in vision_sdk
Q. Does Vision Sdk use GP timer in their configurations which are those?
Yes, Vision Sdk does use the timers. Timer 1,2,3,4 are been used by SDK.
Q. Interrupts used by temperature utils
Vision Sdk uses interrupt no **61** for temperature Temperature Sensor Interrupt. A hot and cold events are triggered to the assigned interrupt number. Temperature hot and cold threshold can be set using api in utils_temperature.h
Q. Timers used by CPU cores to allow CPU load and BIOS Tick when CPUs are powered down
Vision SDK uses GP Timers for the cores (2 per ISA) to use a proxies for the BIOS Tick and TimeStampProvider. To know which timers are used for which CPU core on which device refer the VisionSDK_DevelopmentGuide Chapter 7.1.2

5 Directory Structure Details

Details of the directory structure are as follows

Directory	Description
vision_sdk	Root directory of vision SDK
vision_sdk/build	Support files for building entire release package
vision_sdk/build/makerules	Make rules for components and cores
vision_sdk/build/tda3xx	Rules specifically for TDA3xx device
vision_sdk/docs	Documentation for vision SDK
vision_sdk/examples	Sample Use case / application folder
vision_sdk/examples/tda2xx	**Use case for TDA2xxx and TDA3xx device**
vision_sdk/examples/tda2xx/include	All include files needed for use case
vision_sdk/examples/tda2xx/src	All source files needed for use case
vision_sdk/include	All the include files for the framework
vision_sdk/include/link_api	Interface files for all the links
vision_sdk/src	All the source files for the framework
vision_sdk/src/links_common	Files which are for common across all links
vision_sdk/src/links_dsp	Source files for individual links present on DSP
vision_sdk/src/links_eve	Source files for individual links present on EVE
vision_sdk/src/links_ipu	Source files for individual links present on IPU
vision_sdk/src/main_app	Folder for main() functionality
vision_sdk/src/utils_common	Common utilities used in framework
ti_components	Root directory of tools accompanying vision SDK
ti_components/algorithms_codecs/eve_sw_xx_xx_xx_xx	EVE Kernels Library
ti_components/algorithms_codecs/framework_components_x_xx_xx_xx	Framework Components
ti_components/algorithms_codecs/ivahd_hdvicp20api_xx_xx_xx_xx_xxxx	The HDVICP library is an interface between HDVICP2 hardware and the codec
ti_components/algorithms_codecs/ivahd_jpegvdec_xx_xx_xx_xx_xxxx	MJpeg decoder files
ti_components/algorithms_codecs/vlib_c66x_x_x_x_x	Video Processing Functions Library
ti_components/cg_tools	All the code gen tools
ti_components/cg_tools/windows/arm_x_x_x	Tools needed for ARM CPU cores
ti_components/cg_tools/windows/arp32_x_x_x	Tools needed for ARP32 core
ti_components/cg_tools/windows/c6000_x.x.x	Tools needed for C66x DSP
ti_components/drivers	All the drivers used in Vision SDK
ti_components/drivers/bsp_xx_xx_xx_xx	Driver components for all the peripherals
ti_components/drivers/edma3_lld_xx_xx_xx_xx	Driver for system DMA usage
ti_components/gel/DRA7xx	Gel files modified for Vision-Sdk
ti_components/networking	Networking related tools
ti_components/networking/avbtp_xx_xx_xx_xx	AVB Stack files
ti_components/networking/ndk_x_xx_xx_xx	Network Development Kit
ti_components/networking/nsp_vayu_x_xx_xx_xx	Network Development Kit Support Package
ti_components/drivers/starterware_xx_xx_xx_xx	Lowest level SW interface for programming HW registers
ti_components/os_tools	Operating System related tools

Directory	Description
ti_components/os_tools/bios_x_xx_xx_xx	BIOS operating sytem used in Vision SDK
ti_components/os_tools/ipc_x_xx_xx_xx	Interprocessor communication files
ti_components/os_tools/windows/xdctools_x_xx_xx_xx	XDC tools related files

6 Revision History

Version	Date	Revision History
1.0	22th August 2014	Initial Version
2.0	14th November 2014	Added QSPI+SD boot, GCC and CCSversion
3.0	31st December 2014	Some minor changes
4.0	3rd March, 2015	Added new section DCC
5.0	28th June 2015	Added fast boot usecase
6.0	16th Oct 2015	Updated for release 2.8

«««« § »»»»