# Vision SDK TDA3xx

# (v03.05.00)

# User Guide

**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 info.

## 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 |
| VisionSDK_DataSheet.pdf | Summary of features supported, not supported in a release. Performance and benchmark information. |
| VisionSDK_FAQs.pdf | Document contains FAQ |

## 2 System Requirements

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

### 2.1 Windows Installation

#### 2.1.1 PC Requirements

Installation of this release needs a windows machine with about 8GB of free disk space. Building of the SDK is supported on windows environment.

#### 2.1.2 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

##### 2.1.2.1 A15 Compiler, Linker

The windows installer for the GCC ARM tools should be downloaded from below link

https://launchpad.net/gcc-arm-embedded/+milestone/4.9-2015-q3-update

The tools need to be installed under

 "<install dir>/ti_components/cg_tools/windows/".

**IMPORTANT NOTE: A15 Compiler and linker MUST be installed before proceeding else compile will fail. Also make sure the compiler is installed at the exact path mentioned above**

#### 2.1.3 Code Composer Studio

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

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

## 2.2 Linux Installation

### 2.2.1 PC Requirements

Installation of this release needs a Linux Ubuntu 14.04 machine.

**IMPORTANT NOTE:** If you are installing Ubuntu on a virtual machine, ensure it's a 64 bit Ubuntu.

### 2.2.2 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

#### 2.2.2.1 A15 Compiler, Linker

The Linux installer for the GCC ARM tools should be downloaded from below link

https://launchpad.net/gcc-arm-embedded/+milestone/4.9-2015-q3-update

The tools need to be installed under

"<install dir>/ti_components/cg_tools/linux/".

**IMPORTANT NOTE:** A15 Compiler and linker MUST be installed before initiating the build else compilation will fail. Also make sure the compiler is installed at the exact path mentioned above after installation of vision sdk.

Use following steps to install the toolchain

```
$> cd $INSTALL_DIR/ti_components/cg_tools/linux

$> tar -xvf gcc-arm-none-eabi-4_9-2015q3-20150921-linux.tar.tar
```

**IMPORTANT NOTE:** Ensure the toolchain is for 32 / 64 bit machine as per configuration of installation machine

If your machine is 64 bit and you have downloaded toolchain from link above

Execute following step on installation machine

```
$>sudo apt-get install ia32-libs lib32stdc++6 lib32z1-dev lib32z1 lib32ncurses5 lib32bz2-1.0
```

### 2.2.3 Other software packages for build depending upon OS baseline

Ensure these packages/tools are installed on the installation machine

**uname, sed, mkimage, dos2unix, dtrx, mono-complete, git, lib32z1 lib32ncurses5 lib32bz2-1.0 libc6:i386 libc6-i386 libstdc++6:i386 libncurses5:i386 libz1:i386 libc6-dev-i386 device-tree-compiler mono-complete**

To install

```
$>sudo apt-get install <package_name>
```

## 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 section 2.3.4, 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. LCD should be connected to LCD out connector, as shown in section 2.3.4. 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.

To support multichannel LVDS capture on TDA3xx EVM, there are some board modifications required. Please refer to the "Running VPS Application on TDA3XX EVM" section of the VPS 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 are displayed. This is required as some of the VIP input pins is multiplexed 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 (AR0140 sensor) or IMI modules (OV10640 Rev E sensor) & 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 1080p60 capable Display Monitor

**WARNING: CSI2 Clock:** The maximum CSI2 clock could be 750 MHz, please refer the device data manuals for details. Some of the VisionSDK usecases (UB964 based), overclocks by 50 MHz (i.e. 800 MHz) and it works as expected. This over clocking is due the crystal (25 MHz) used in UB964 EVM, by choosing 24 MHz crystal UB964 CSI2 clock can be operated with-in specified limits.

## 2.3.4 Sensor Interfaces
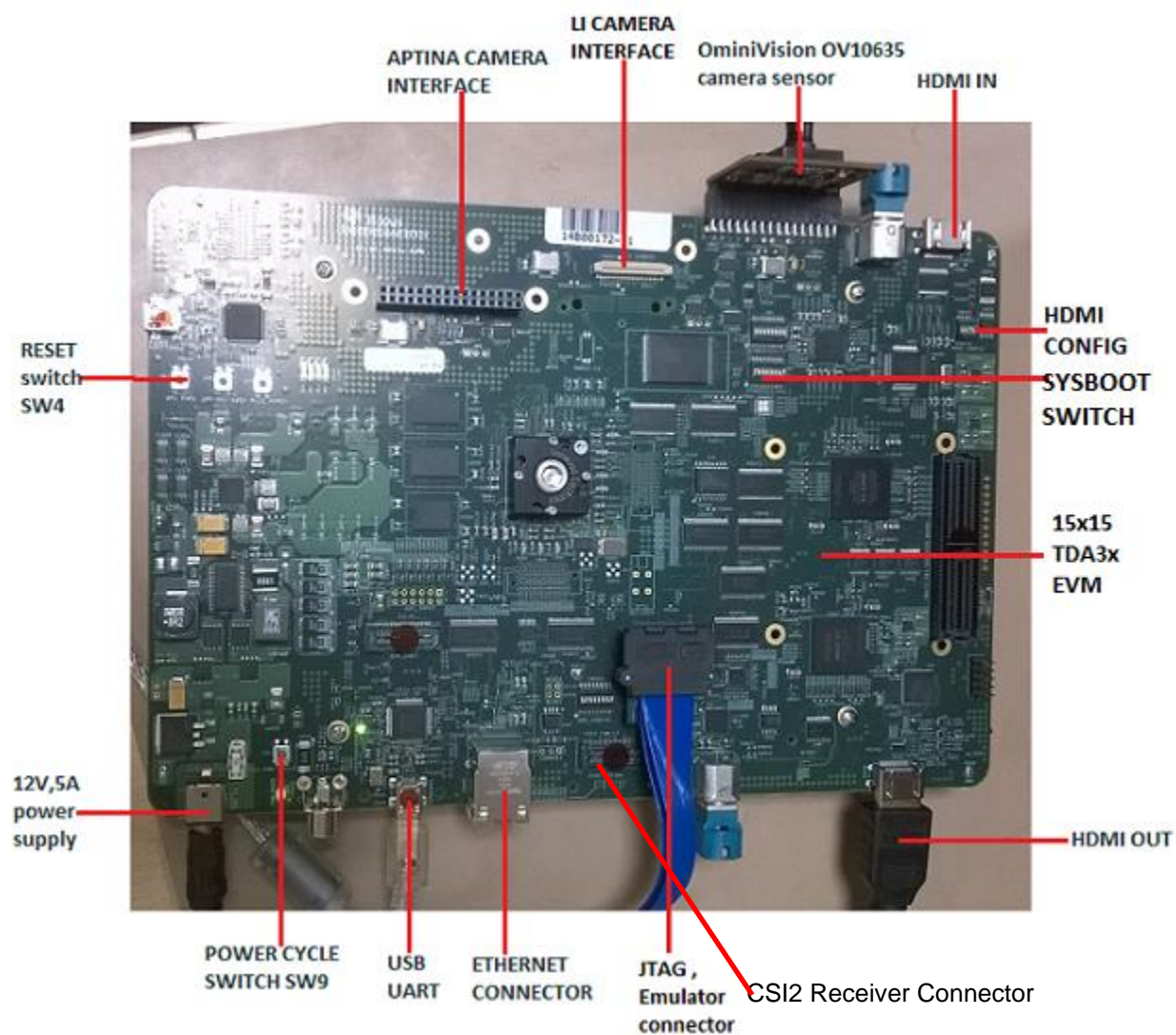


**Figure 2.3 1 15x15 TDA3x EVM (FRONT SIDE)**
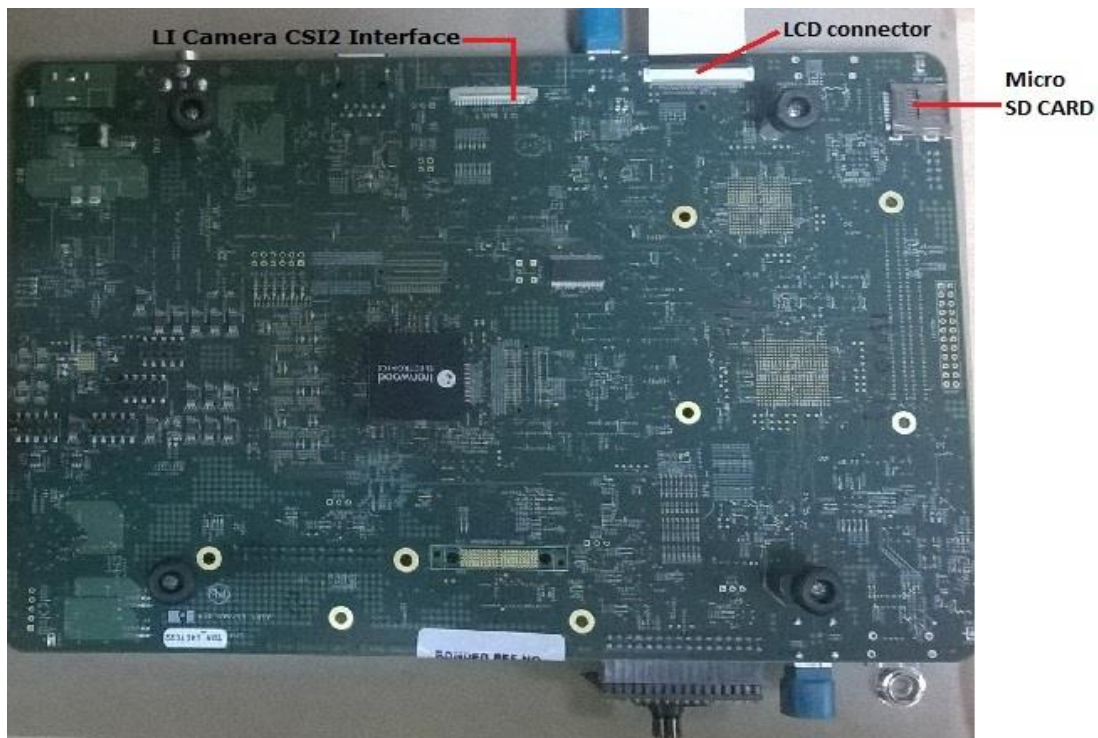
**Figure 2.3 2 15x15 TDA3x EVM (BACK SIDE)**
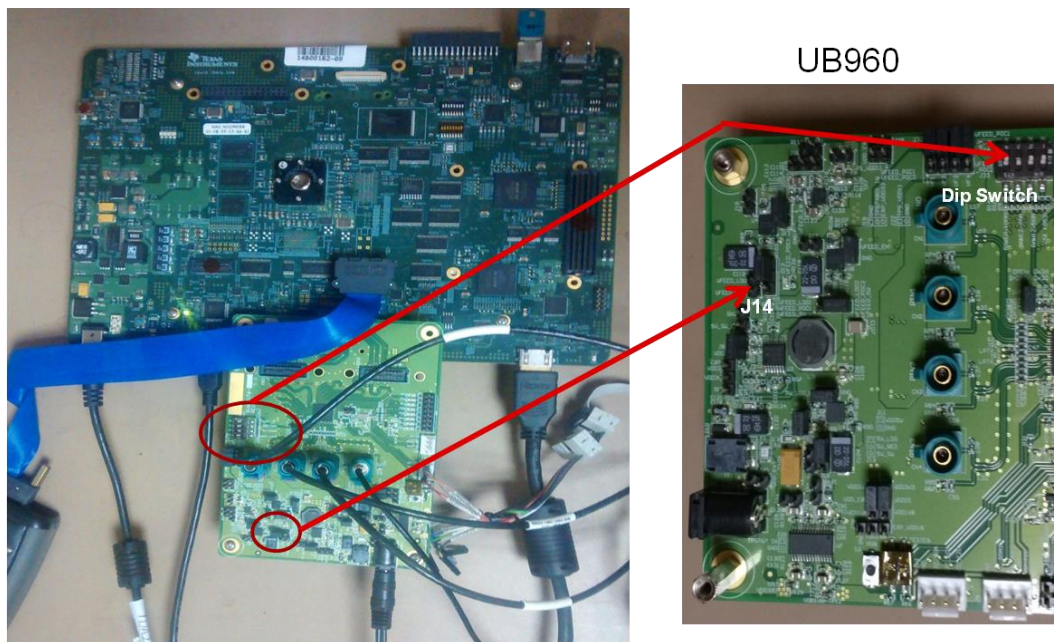


**Figure 2.3 3 TDA3x EVM with UB960 Application Board**

*NOTE:* **For Switch settings refer section 2.4.2**

### 2.3.5    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.

*IMPORTANT NOTE:* **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. Refer the EDID programming points in the section** *Running VPS Application on (TDAXXX EVM)* **documented in** *VPS User Guide* **in** *PDK***.**

## 2.4 Required H/W modification / Configurations

### 2.4.1 TDA3XX EVM Modifications for SCH use case

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

### 2.4.2 Changes required on UB960 Application board

1. Configure to supply 9V on the LVDS lines to TIDA00262 modules
   a. J14 Short pins 1-2 as show in **Error! Reference source found.**
2. Configure UB960 to operate in LVDS in 75 MHz mode
   a. Dip Switch MODE 3 should be ON
3. Refer the picture below

| Mode | | |
|---|---|---|
| | □ | 4 |
| □ | | 3 |
| | □ | 2 |
| | □ | 1 |
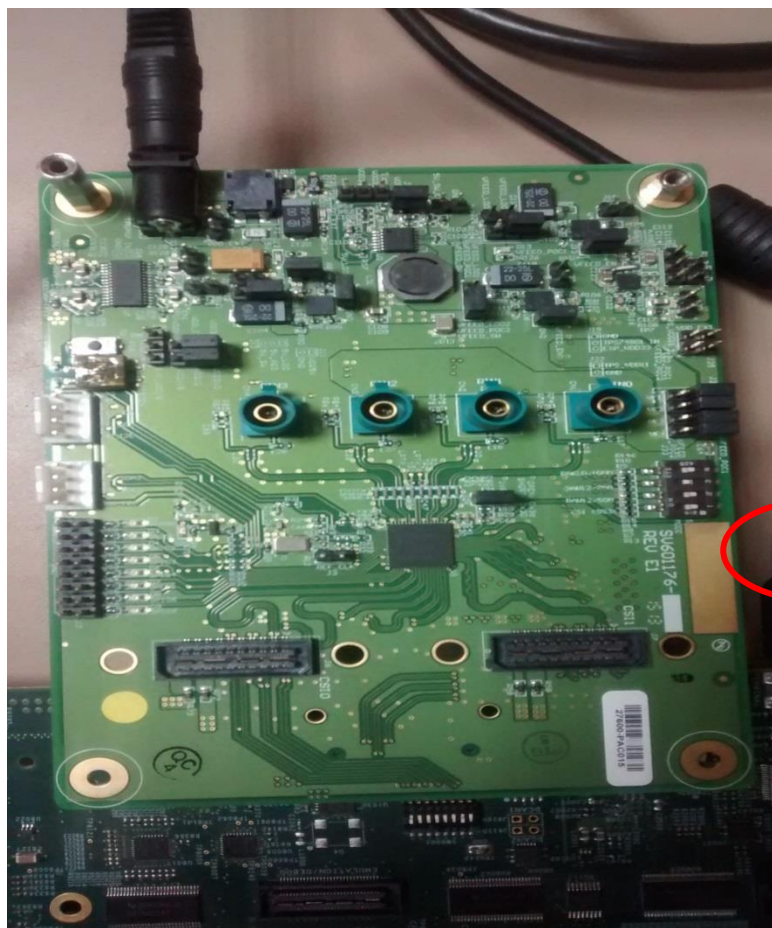
## 2.5 Supported Sensors

Refer to ProcessorSDK_Vision_IssSensor_TestMatrix.xlsx in docs/TestReports for the supported feature on each of the sensors.

## 2.6 Software Installation

PROCESSOR_SDK_VISION_03_xx_xx_xx_setupwin.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 PROCESSOR_SDK_VISION_03_xx_xx_xx would have been created in the installation path.

### 2.6.1 Uninstall Procedure

To uninstall, double click on uninstall.exe created during installation in the folder PROCESSOR_SDK_VISION_03_xx_xx_xx.

At the end of uninstall, PROCESSOR_SDK_VISION_03_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. For more details about optimized build process please refer to VisionSDK_UserGuide_BuildSystem.pdf

## 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)
- 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 2.4.2 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

a) On windows command prompt, go inside the directory PROCESSOR_SDK_VISION_03_xx_xx_xx\vision_sdk\build.

b) Open file Rules.make and

set **MAKEAPPNAME=apps** and **MAKECONFIG=tda3xx_evm_bios_all**

c) 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

**IMPORTANT NOTE:** A15 Compiler and linker MUST be installed before proceeding else compile will fail. Also make sure the compiler is installed at the exact path as mentioned in Directory Structure .

**IMPORTANT NOTE**: **If the installation folder depth is high then windows cmd prompt fails with error that it cannot find a file, even in file is present in mentioned path, this is because Windows has a limitation of 8191 characters for the commands that can execute. In such a situation as a workaround either restrict the folder depth to d:/ or if it cannot be restricted use git bash (version 2.13) to build.  Refer** https://support.microsoft.com/en-in/kb/830473 **for more details**.

(**Always point to xdc path gmake only**)

d) Under vision_sdk\build directory

   i.   When building first time run the below sequence of commands

> **> gmake -s -j depend**
> **> gmake -s –j**

**IMPORTANT NOTE**: For Windows PC use "-j<number of CPUs>" instead of just –j. For example if PC has 2 CPUs then use "-j2". Random build dependency issues has been noticed with –j & windows PC. If not sure about the number of CPUs of PC, then suggests not using –j option with windows build environment.

   ii.   When building after the first time or incremental build, run the below command
**> gmake -s -j**

Executing "**gmake -s -j depend** " will build all the necessary components (PDK drivers, EDMA drivers and sdk dependent files) and "**gmake -s -j"** will build the Vision SDK framework and examples.

**IMPORTANT NOTE**: For incremental build, make sure to do "gmake -s -j depend" before "gmake -s -j" when below variables specified in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\*cfg.mk are changed

- ➢ when PROC_$(CPU)_INCLUDE is changed
- ➢ when DDR_MEM is changed
- ➢ when PROFILE is changed
- ➢ when ALG plugin or usecase is enabled or disabled in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG) \*_cfg.mk

> ➢ 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 "gmake -s -j depend" is not done in these cases then build and/or execution may fail.

**IMPORTANT NOTE**: When options (other than those specified above) are changed in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk, a clean build is recommended for the updated settings to take effect.

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

```
\vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\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
```

f) 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 \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk. A value of "no" means CPU not included in build, value of "yes" means CPU included in build. Make sure to do clean build and then "**gmake -s -j depend**" before "**gmake -s- j**" when number of CPUs included is changed

```
PROC_DSP1_INCLUDE=yes

PROC_DSP2_INCLUDE=yes

PROC_EVE1_INCLUDE=yes

PROC_IPU1_0_INCLUDE=yes

PROC_IPU1_1_INCLUDE=yes
```

g) The build configuration that is selected in config file can be confirmed by doing below

```
> gmake -s showconfig
```

h) Cleaning the build can be done by following command
```
> gmake -s clean
```
Alternatively, below folder can be deleted to delete all generated files
```
> rm -rf
..\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\vision_sdk\bin

> rm -rf ..\links_fw\include\configs
```

## 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
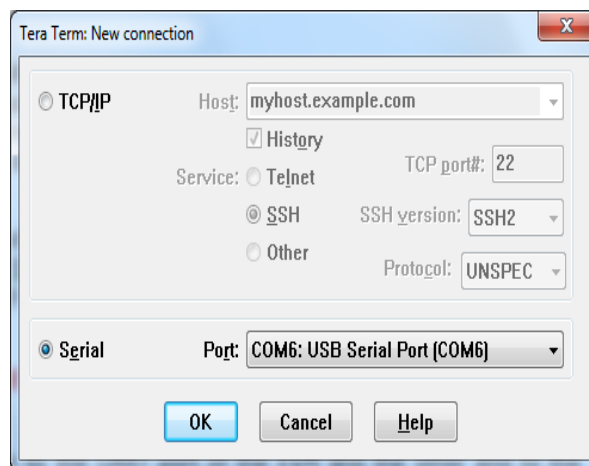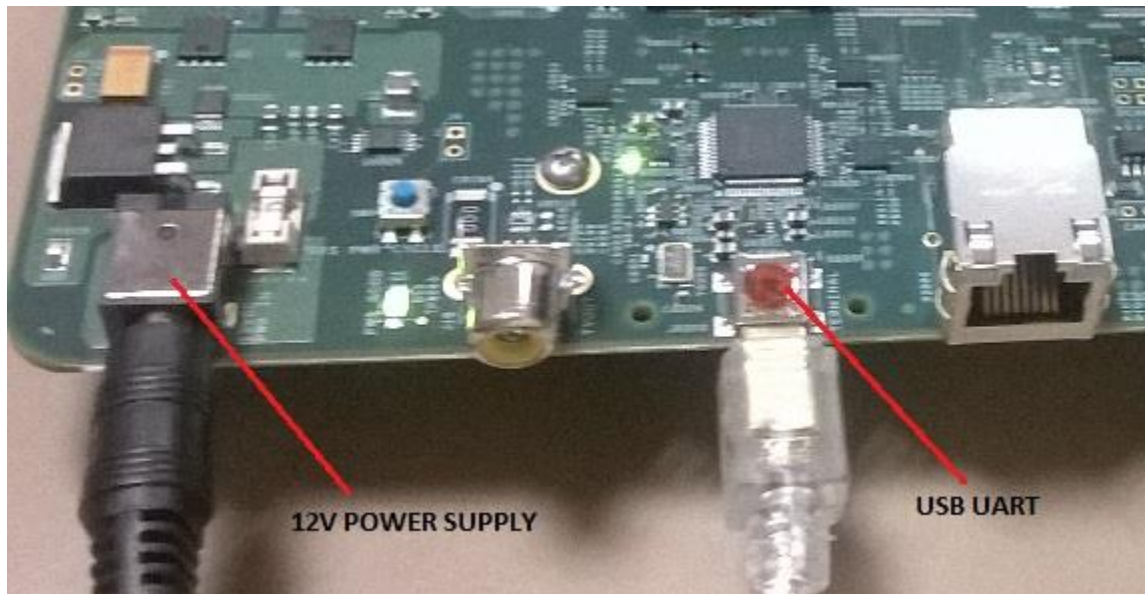
Tera Term: Serial port setup

Port:            COM6
Baud rate:       115200
Data:            8 bit
Parity:          none
Stop:            1 bit
Flow control:    none

OK
Cancel
Help

Transmit delay
0  msec/char     0  msec/line

## 3.4 Boot Modes

Supported boot modes on TDA3xx ES1.0 and ES2.0 device:

| Boot Mode | EVM Switch Setting SYSBOOT(SW2)[1:16] | EVM Switch Setting SW8001[1:8] |
|-----------|---------------------------------------|-------------------------------|
| QSPI_1 | 00011000 10000001 | 0100 0001 |
| QSPI_4 | 10011000 10000001 | 0100 0001 |
| NOR | 01011000 10000101 | 1100 0001 |
| Debug | 00111000 10000001 | XXXXXXXX |

## 3.5 Load using QSPI

### 3.5.1 Steps to generate qspi writer tools

NOTE: SBL qspi image is built from pdk package. To build qspi, run the command gmake -s sbl from vision_sdk\build directory This generates all required tools and all sbl images under vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl directory

1. The flash writer is present in vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi_flash_writer\$(PLATFORM)\qspi_flash_writer_ipu1_0_release.xem4
2. The SBL images are present in vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi\$(OPP)\$(PLATFORM)\sbl_qspi_$(OPP)_ipu1_0_release.tiimage

**IMPORTANT NOTE: "gmake -s sbl"** requires GCC tools need to be installed in "<install dir>/ti_components/cg_tools/<os>/gcc-arm-none-eabi-4_9-2015q3" location. Tool can be downloaded from below link.

https://launchpad.net/gcc-arm-embedded/+milestone/4.9-2015-q3-update

### 3.5.2 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 below command from "vision_sdk\build" folder
   > gmake -s appimage

**IMPORTANT NOTE:**

- The config options, like CPUs to use, debug or release profile etc, used to make the application image will be the values specified in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk

- **The Surround View LUT and Perspective Matrix are flashed at an offset of 20 MB in the QSPI hence make sure the generated appImage doesn't exceed 20 MB in case Surround View use cases are intended to be run.**

### 3.5.3 Flashing steps

Flashing pin settings: Please refer Boot Modes for pin boot mode pin setting.

SYSBOOT
Switch
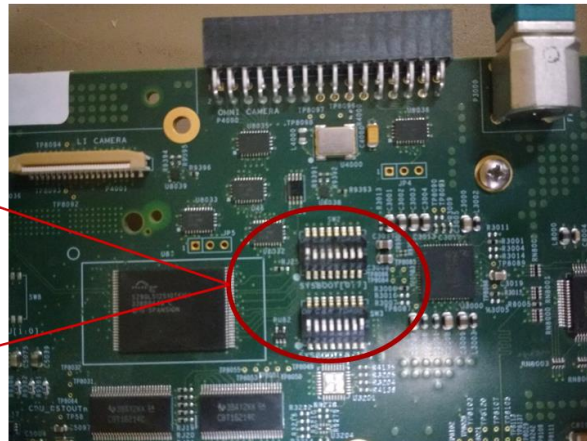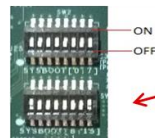
**NOTE: Image indicates the sysboot position on board not the switch settings**

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

1. Connect M4 (IPU). Do CPU reset

2. Load below image on M4

   **C:\PROCESSOR_SDK_VISION_03_XX_XX_XX\vision_sdk\binaries\$(MA KEAPPNAME)\$(MAKECONFIG)\sbl\qspi_flash_writer\$(PLATFORM)\qs pi_flash_writer_ipu1_0_release.xem4**

3. Run the core. You would see below console logs

| | |
|---|---|
| [Cortex_M4_IPU1_C0]<br>QSPI Flash writer application<br>Enter Device type to use<br>1 - 1 bit read from flash<br>2 - 4 bit (Quad) read from flash<br><br>**Select appropriate Device Type, for TDA3x EVM, press '2'.**<br><br>MID - 1<br><br>DID - 18<br><br>Enter 0 for Erase-Only (without flashing any image)<br><br>Note : File size should be less than 33554432 Bytes.<br><br>Enter the file path to flash: C:\PROCESSOR_SDK_VISION_03_XX_XX_XX\vision_sdk\binaries\$(M AKEAPPNAME)\$(MAKECONFIG)\sbl\qspi\$(OPP)\$(PLATFORM)\sbl_qs pi_$(OPP)_ipu1_0_release.tiimage<br><br>**Enter the Offset in bytes (HEX)  0x00** | Erase Options:<br>---------------<br>    0 -> Erase Only Required Region<br>    1 -> Erase Whole Flash<br>    2 -> Skip Erase<br>**Enter Erase Option:  1**<br><br>Load Options:<br>-------------<br>  0 -> fread using code (RTS Library)<br>  1 -> load raw using CCS (Scripting console)<br>**Enter Load Option: 0**<br><br>Read xxxxxx bytes from [100%] file...Done.<br> QSPI whole chip erase in progress<br> QSPI file write started<br>\*\*\*\*\*\*\*\*\*\*\*\*QSPI     flash     completed sucessfully\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

4. Reset the board and Repeat step 1, 2, 3.

<table>
<tr><td>

```
[Cortex_M4_IPU1_C0]
QSPI Flash writer application
Enter Device type to use
1 - 1 bit read from flash
2 - 4 bit (Quad) read from flash
```

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

*MID - 1*

*DID - 18*

*Enter the File Name*

**c:\PROCESSOR_SDK_VISION_03_XX_XX_XX\vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\vision_sdk\bin\$(SOC)\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**

</td><td>

*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.*

**loadRaw(0x80500000, 0, "C:/VISION_SDK_XX_XX_XX_XX/vision_sdk/binaries/$(MAKEAPPNAME)/$(MAKECONFIG)/vision_sdk/bin/$(SOC)/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 below image*

*QSPI file write started*

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

</td></tr>
</table>

NOTE: If flashing binaries for fast boot use case 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 for steps to generate binaries, binary names, offsets

<table>
<tr><td>

```
[Cortex_M4_IPU1_C0]
QSPI Flash writer application
Enter Device type to use
1 - 1 bit read from flash
2 - 4 bit (Quad) read from flash
```

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

*MID - 1*

*DID - 18*

*Enter the File Name*

**c:\PROCESSOR_SDK_VISION_03_XX_XX_XX\vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\vision_sdk\bin\$(SOC)\sbl_boot\AppImage_UcEarly_BE**

</td><td>

*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.*

**loadRaw(0x80500000, 0, "C:/VISION_SDK_XX_XX_XX_XX/vision_sdk/binaries/$(MAKEAPPNAME)/$(MAKECONFIG)/vision_sdk/bin/$(SOC)/sbl_boot/AppImage_UcEarly_BE", 32, false);**

</td></tr>
</table>

| | |
|---|---|
| Enter the Offset in bytes (HEX): **0x80000**<br><br>Erase Options:<br>---------------<br><br>         0 -> Erase Only Required Region<br>         1 -> Erase Whole Flash<br>         2 -> Skip Erase<br>      Enter Erase Option: **0** | **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.*<br><br>*In CCS console Enter any alpha-numeric key once loadraw is complete... as shown in below image*<br><br>*QSPI file write started*<br>*\*\*\*\*\*\*\*\*\*\*\*\*QSPI flash completed successfully\*\*\*\*\*\*\*\*\*\*\*\*\** |

| | |
|---|---|
| ```[Cortex_M4_IPU1_C0]```<br>```QSPI Flash writer application```<br>```Enter Device type to use```<br>```1 - 1 bit read from flash```<br>```2 - 4 bit (Quad) read from flash```<br>***Select appropriate Device Type, for TDA3x EVM, press '2'.***<br><br>*MID - 1*<br><br>*DID - 18*<br><br>Enter the File Name<br>**c:\PROCESSOR_SDK_VISION_03_XX _XX_XX\vision_sdk\binaries\$(MAK EAPPNAME)\$(MAKECONFIG)\vision _sdk\bin\$(SOC)\sbl_boot\AppIma ge_UcLate_BE**<br><br>Enter the Offset in bytes (HEX): **0xA80000**<br><br>Erase Options:<br>---------------<br><br>         0 -> Erase Only Required Region<br>         1 -> Erase Whole Flash<br>         2 -> Skip Erase<br>      Enter Erase Option: **0** | *Load Options:*<br>*------------*<br>  *0 -> fread using code (RTS Library)*<br>  *1 -> load raw using CCS (Scripting console)*<br>*Enter Load Option: **1***<br><br>*Open Scripting console window by clicking "Menu -> View -> Scripting console" and enter below command on scripting console.*<br><br>***loadRaw(0x80500000, 0, "C:/VISION_SDK_XX_XX_XX_XX/vision_s dk/binaries/$(MAKEAPPNAME)/$(MAKECO NFIG)/vision_sdk/bin/$(SOC)/sbl_boot/A ppImage_UcLate_BE", 32, false);***<br><br>**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.*<br><br>*In CCS console Enter any alpha-numeric key once loadraw is complete... as shown in below image*<br><br>*QSPI file write started*<br>*\*\*\*\*\*\*\*\*\*\*\*\*QSPI flash completed successfully\*\*\*\*\*\*\*\*\*\*\*\*\** |

5. On completion change the pin setting as shown in Boot Modes table.

## 3.6 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.

### 3.6.1 Steps to generate qspi writer tools

NOTE: SBL image for the qspi sd boot is built from pdk package.To build this image, Run the command **gmake -s sbl** from vision_sdk\build dir. This generates all required tools under vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\

1.  The flash writer is present in
    vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi_flash_writer\$(PLATFORM)\qspi_flash_writer_ipu1_0_release.xem4
2.  The SBL images are present in
    vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi_sd\$(OPP)\$(PLATFORM)\sbl_qspi_sd_$(OPP)_ipu1_0_release.tiimage

**IMPORTANT NOTE: "gmake -s sbl**" requires GCC tools need to be installed in "<install dir>/ti_components/cg_tools/<os>/gcc-arm-none-eabi-4_9-2015q3" location. Tool can be downloaded from below link.

https://launchpad.net/gcc-arm-embedded/+milestone/4.9-2015-q3-update

### 3.6.2 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 below command from "vision_sdk\build" folder
    > gmake -s appimage

**IMPORTANT NOTE:** The config options, like CPUs to use, debug or release profile etc, used to make the application image will be the values specified in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk

### 3.6.3 Flashing steps

Flashing pin settings: Please refer Boot Modes for pin boot mode pin setting.
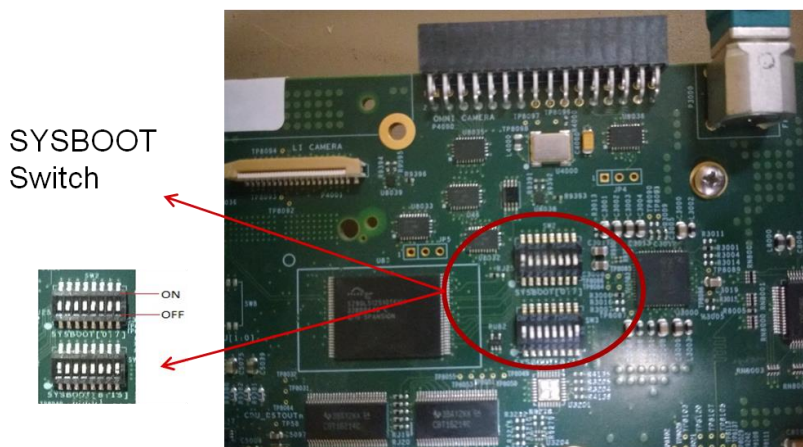


For loading binaries using CCS refer Load using CCS till step 8.

1. Connect M4 (IPU). Do CPU reset

2. Load image on M4
   (**C:\PROCESSOR_SDK_VISION_03_XX_XX_XX\vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi_flash_writer\$(PLATFORM)\qspi_flash_writer_ipu1_0_release.xem4**)

3. Run the core.

   You should get below logs on console outputs

| | |
|---|---|
| `[Cortex_M4_IPU1_C0]`<br>`QSPI Flash writer application`<br>`Enter Device type to use`<br>`1 - 1 bit read from flash`<br>`2 - 4 bit (Quad) read from flash`<br><br>*Select appropriate Device Type, for TDA3x EVM, press '2'.*<br><br>*MID - 1*<br><br>*DID - 18*<br><br>*Enter 0 for Erase-Only (without flashing any image)*<br><br>*Note : File size should be less than 33554432 Bytes.* | *Enter the file path to flash:*<br>C:\PROCESSOR_SDK_VISION_03_XX_XX_XX\vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi_sd\$(OPP)\$(PLATFORM)\sbl_qspi_sd_$(OPP)_ipu1_0_release.tiimage<br><br>*Enter the Offset in bytes (HEX)* **0x00**<br><br>*Erase Options:*<br>*---------------*<br>   *0 -> Erase Only Required Region*<br>   *1 -> Erase Whole Flash*<br>   *2 -> Skip Erase*<br>*Enter Erase Option:* **1**<br><br><br>*Load Options:*<br>*-------------*<br>  *0 -> fread using code (RTS Library)*<br>  *1 -> load raw using CCS (Scripting console)*<br>*Enter Load Option:* **0**<br><br><br>*Read xxxxxx bytes from [100%] file...Done.*<br> *QSPI whole chip erase in progress*<br> *QSPI file write started*<br>************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.

## 3.7 Load using CCS

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

1. GELs are available in

   <Install_dir>\ti_components\ccs_csp \auto_device_support_x.x.x.zip
   NOTE:

   o GELs are also available at
     http://processors.wiki.ti.com/index.php/Device_support_files

     Under Automotive pick
     **Automotive vX.X.X**

   o To install the new GEL versions, you need to extract the zip to
     <CCS_INSTALL_DIR>/ccsv6/ccs_base

     Change the following GEL files for vision SD as below,

   - TDA3xx_multicore_reset.gel
       o Set VISION_SDK_CONFIG to 1
       o 256MB mode not supported

2. CCS Target Configuration creation:

   a. Open "Target Configurations" tab, by navigating through the menu "View -
      >Target Configurations".
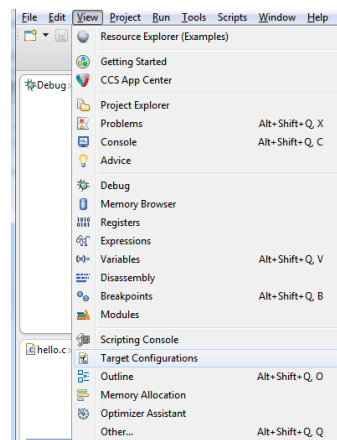


   b. Create a new Target Configuration (TDA3xx Target Configuration) by
      navigating through the menu "File->New->Target Configuration File".
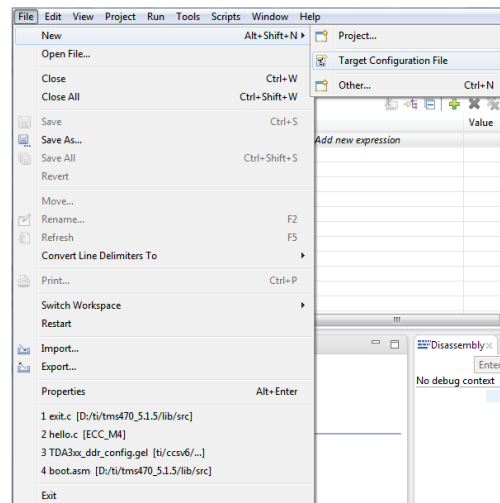
c. Specify Connections as "Spectrum Digital XDS560V2 STM USB Emulator". Specify Board or Device as "TDA3x".
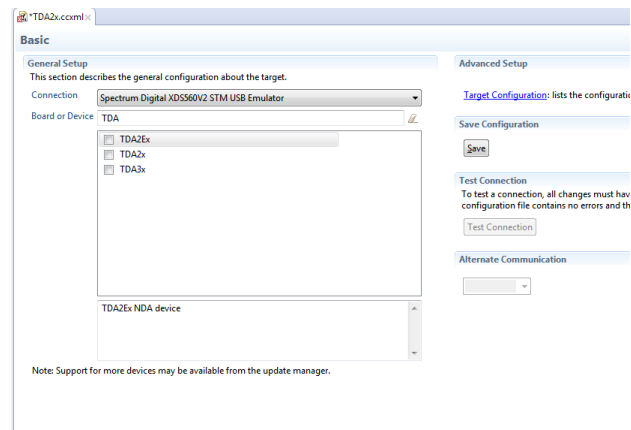


3. Connect JTAG to the board.



JTAG EMULATOR
CONNECTOR
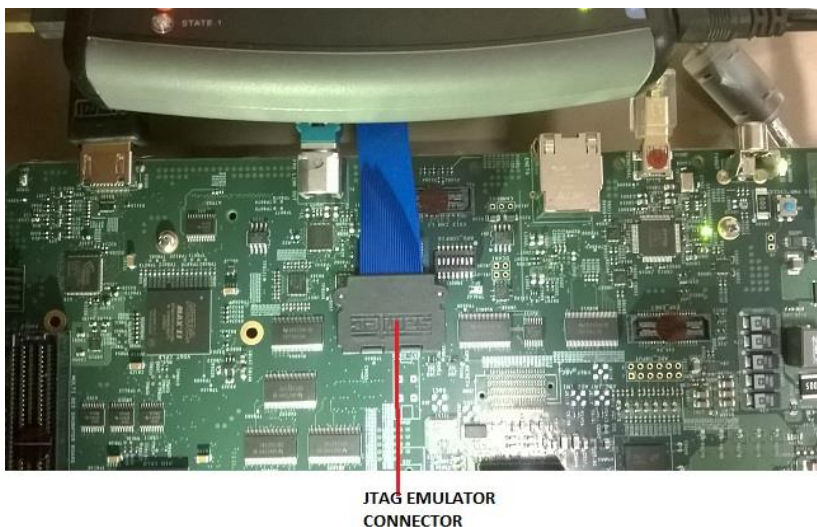
4. Reset EVM through the power recycle button.



12V, 5A Power          Power Cycle
Supply                 Switch
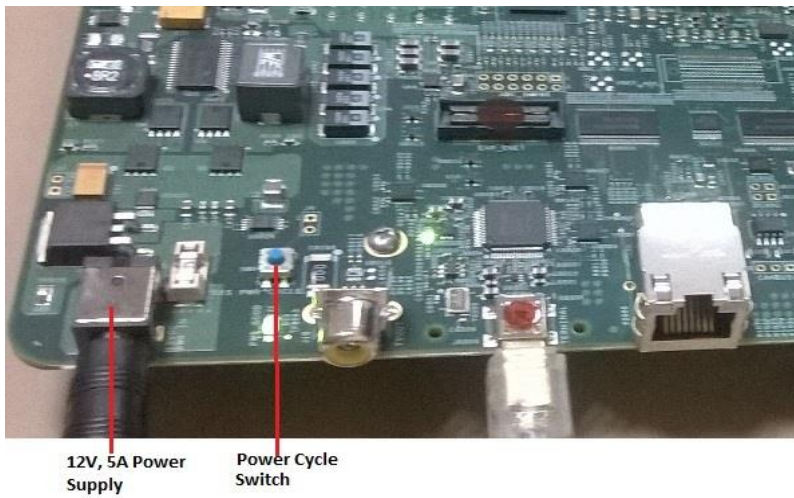
5. Now launch the previously created TDA3xx Target Configuration.
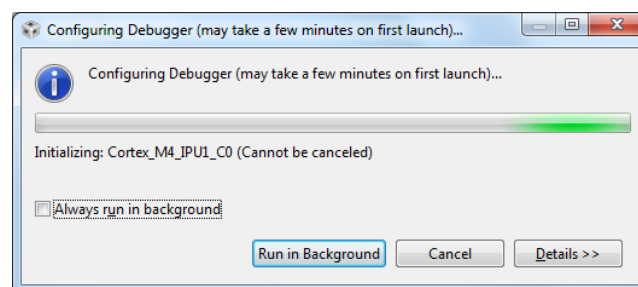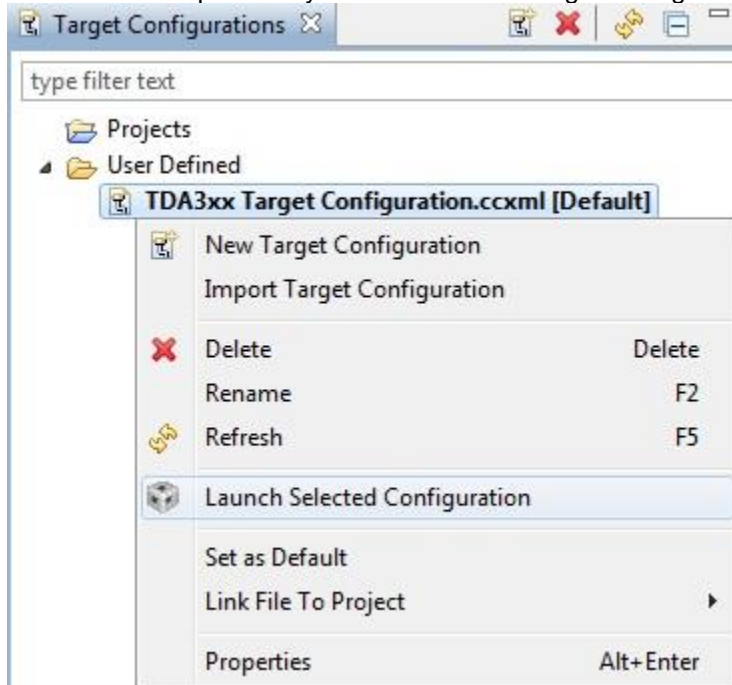
6. Connect to core Cortex_M4_IPU1_C0.
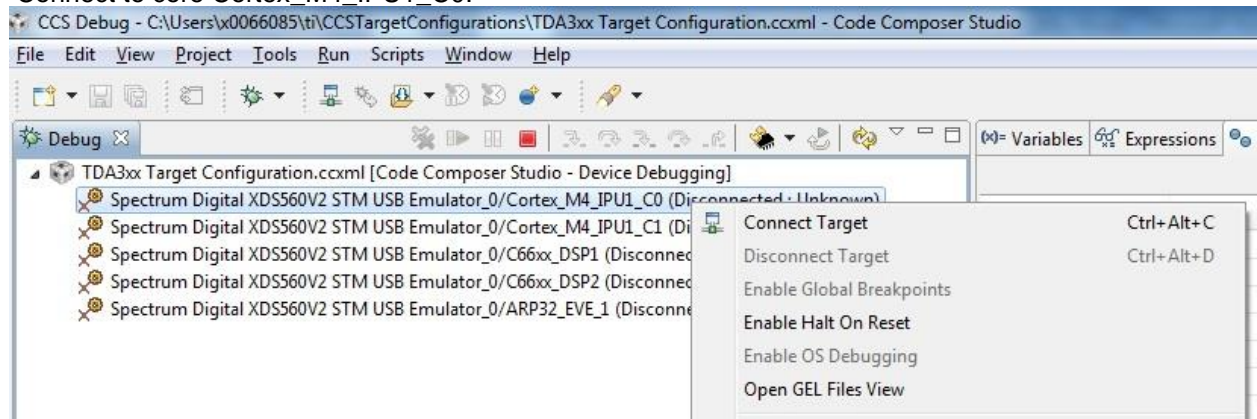


7. On successful connect, the following log appears on CCS console:

*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
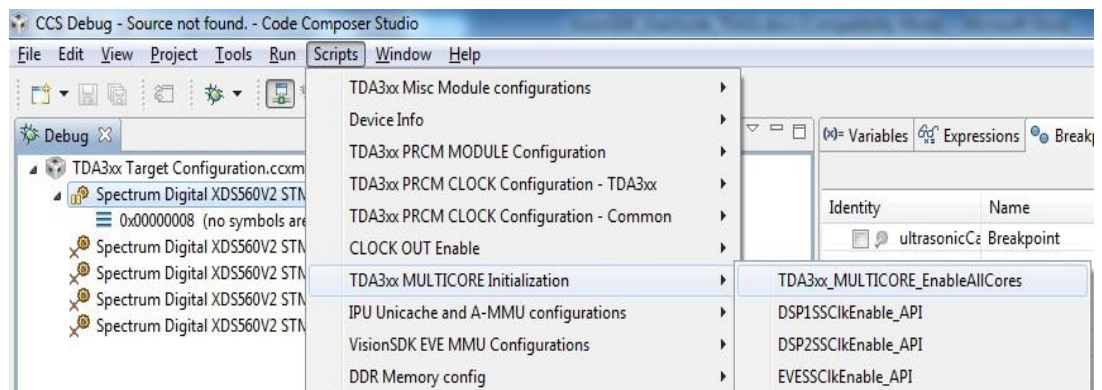


9. On successful script execution, the following log appears on CCS console:

*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. 
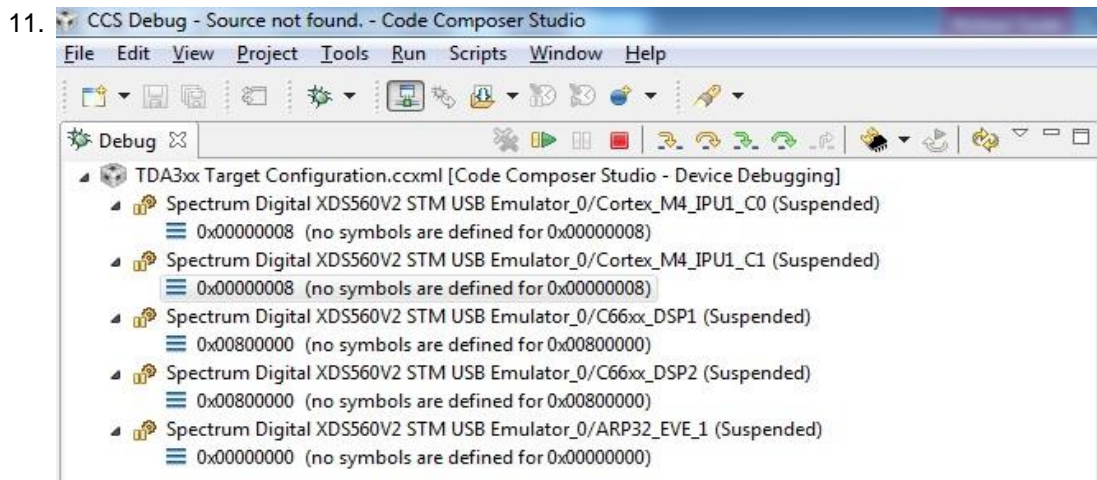

12. On the cores load the binaries as mentioned below
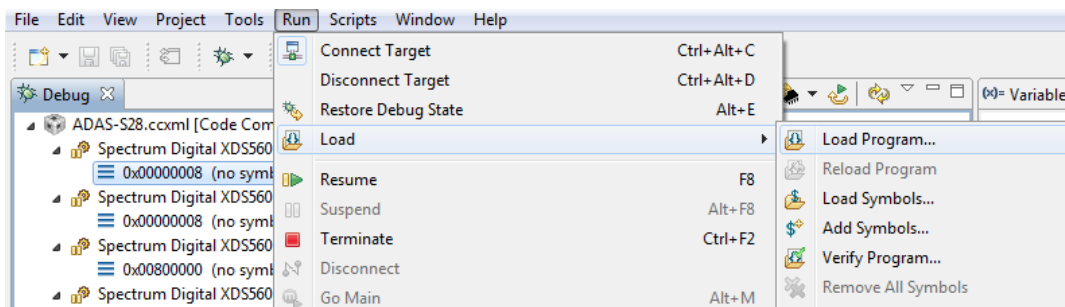


On ARP32_EVE_1, load the binary, "vision_sdk_arp32_1_release.xearp32F".
On C66xx_DSP2, load the binary, "vision_sdk_c66xdsp_2_release.xe66".
On C66xx_DSP1, load the binary, "vision_sdk_c66xdsp_1_release.xe66".
On Cortex_M4_IPU1_C0, load the binary, "vision_sdk_ipu1_0_release.xem4".
On Cortex_M4_IPU1_C1,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

## 3.8 Run the demo

### 3.8.1 Single channel demos with HDMI input

**IMPORTANT NOTE:** To demonstrate better output all single channel use cases that require HDMI input should use video clips mentioned in the table below. These clips can be downloaded from
https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.30602.25095 .

| Use case No. "Runtime Menu" | Use case | Input clip to be played by HDMI player |
|---|---|---|
| 7 | 1CH VIP capture + Sparse Optical Flow (EVE1) + Display | Clip2 |
| b | b: 1CH VIP capture (HDMI) + Lane Detect (DSP1) + Display | Clip1 |
| c | c: 1CH VIP capture (HDMI) + SOF (EVE1) + SFM (DSP1) + Display | Clip2 |
| d | d: 1CH VIP capture (HDMI) + Traffic Light Recognition (TLR) (DSP1) + Display | Clip2 |
| e | e: 1CH VIP capture (HDMI) + Pedestrian, Traffic Sign, Vehicle Detect 2 (EVE1 + DSP1) + Display | Clip2 |
| f | f: 1CH VIP capture (HDMI) + FrontCam Analytics 2 (PD+TSR+VD+LD+TLR+SFM) (DSPx, EVEx) + Display (HDMI) | Clip3 |

**SFM_POSE.bin** - SFM (Usecase 'c') and EUNCAP demo (Usecase 'f') needs SFM_POSE.bin on the **SD card**. It will be available as part of the file downloaded from the above link.

### 3.8.2 Steps to run
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. For HDMI as input select capture source as HDMI "s: System Settings"-> "Capture Settings" -> "2: HDMI Capture 1080P60"
3. 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
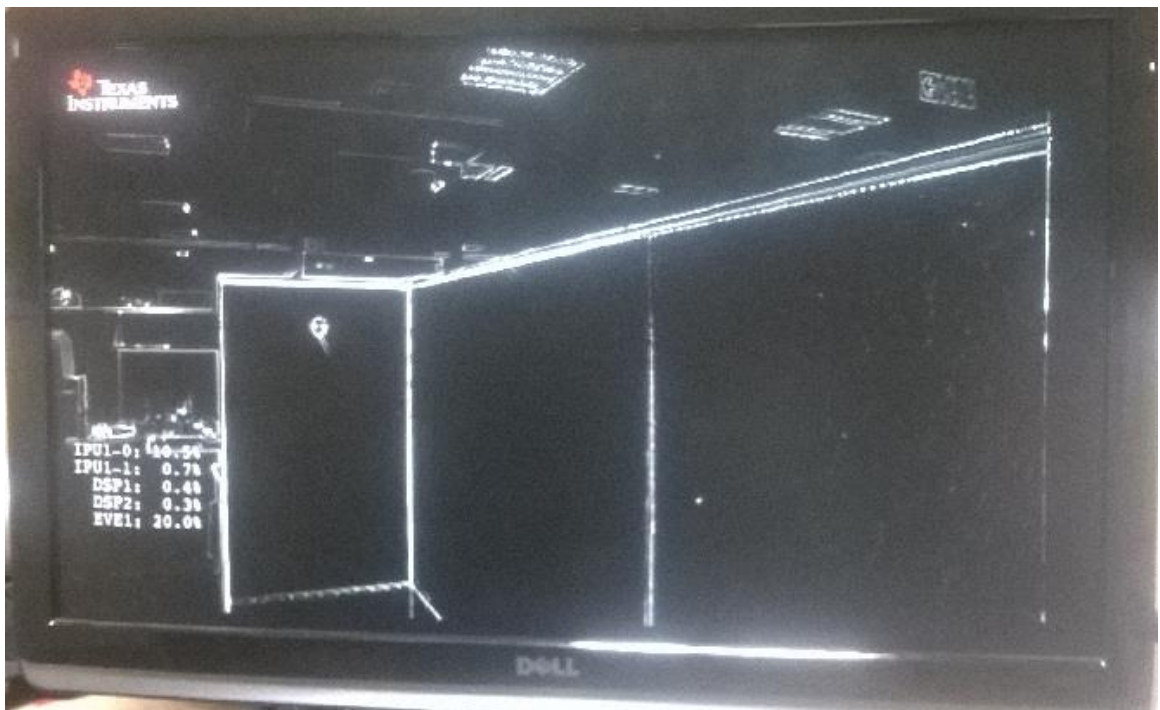
After successful initialization of the use-case, you will see video been display on the HDMI as shown below
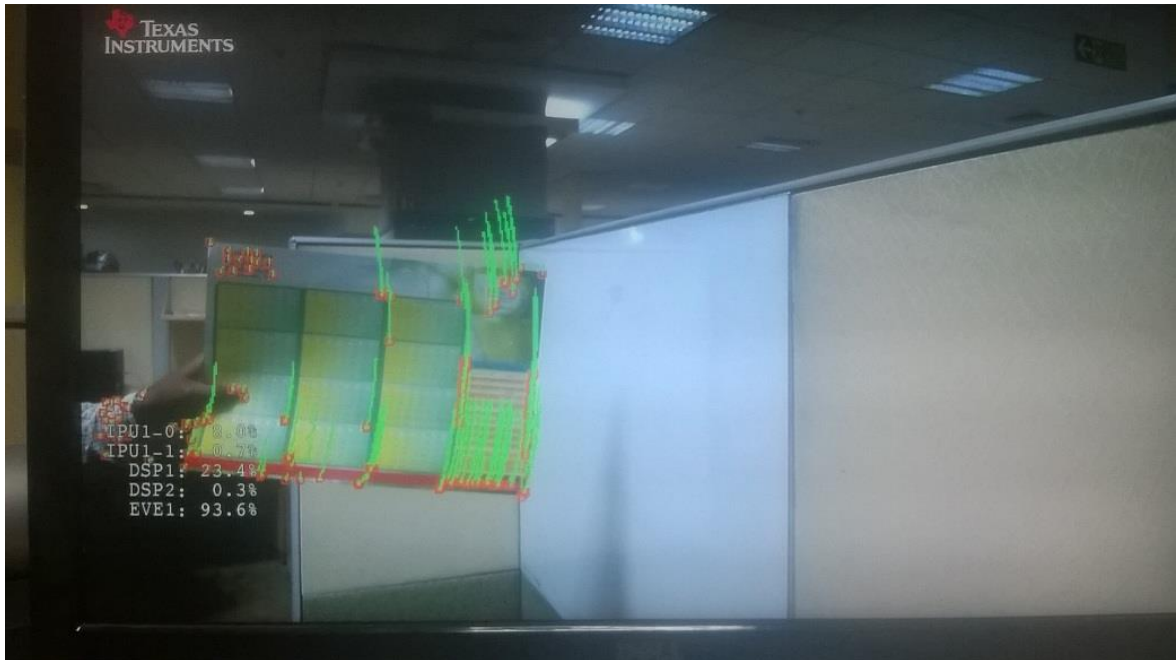
a. SCV use-cases:



b. EDGE Detect use-case:



c. Sparse optical flow usecase

## 3.9 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 3.0 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 user guide 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 in the Vision SDK ISS sensor framework.

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 (VisionSDK_UserGuide_NetworkTools.pdf) for more information on this command.

When ISS use case 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 ISS sensor layer. If the binary file is not available even in ISS sensor layer, it uses ISP default parameters.

**IMPORTANT NOTE:** DCC is currently supported only for AR0140, AR0132, IMX224 and OV10640 Rev E sensors. For AR0140 sensor, the DCC xml files with the tuned ISP parameters can be found in the path vision_sdk\apps\src\rtos\iss\src\sensor\ar0140\dcc_xml, for AR0132 sensor, they can be found in the path vision_sdk\apps\src\rtos\iss\src\sensor\ar0132\dcc_xml, for OV10640 Rev E driver, they can be found in the path vision_sdk\apps\src\rtos\iss\src\sensor\ov10640\dcc_xml and for IMX224 driver, they can be found in the path vision_sdk\apps\src\rtos\iss\src\sensor\imx224\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 vision_sdk\apps\tools\dcc_tools\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 (VisionSDK_UserGuide_NetworkTools.pdf) 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 vision_sdk\apps\src\rtos\iss\src\sensor\ar0140\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\apps\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 Vision SDK utility, it can be found under vision_sdk\apps\tools\dcc_tools\bin2c.exe

e. Replace this header file in vision_sdk\apps\src\rtos\iss\src\sensor\iss_tables\iss_tables_ldc_lut_1920x1080.h

f. Rebuild vision sdk

## 3.10 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 output 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.

### 3.10.1 Usecase configuration

It supports following configuration _Only_

1CH ISS Capture + ISP + LDC + Obj detect + Display

- ➢ Capture - AR0140 Parallel with TDA3x EVM
- ➢ Display - 10 inch LCD
- ➢ Boot mode - QSPI

### 3.10.2 Hardware set up

Refer section 0 "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 cannot be skipped.

- I2C to run at 400KHz
- Support for AR0140 or OV10640 REV E



**Figure: TDA3x EVM Fast boot h/w setup**

### 3.10.3 Build

Fast boot 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 \vision_sdk\apps\configs\tda3xx_evm_bios_all\cfg.mk. By default it is "no".

```
# Fast boot usecase is currently supported only for tda3x
FAST_BOOT_INCLUDE=yes
…
```

Remove DSP2 and IPU1_1 from the \vision_sdk\apps\configs\tda3xx_evm_bios_all\cfg.mk & 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'

- "gmake -s showconfig" command can be used prior to build to confirm the build configuration.
- To build use
  o gmake -s -j depend
  o gmake -s –j
- SBL also needs to be built for fast boot usecase
  - Ensure to remove SBL binaries if SBL was built previously
  - Use "gmake -s sbl" to build SBL

### 3.10.4 Generating and Flashing images
- Refer section 3.5.1 to generate sbl

- To generate the application image run below command from "vision_sdk\build" folder
  > gmake -s appimage

  **IMPORTANT NOTE:** The config options, like CPUs to use, debug or release profile, fast boot enable etc, used to make the application image will be the values specified in \vision_sdk\apps\configs\tda3xx_evm_bios_all\cfg.mk

  - This command should generate AppImages at \vision_sdk\binaries\ apps\tda3xx_evm_bios_all\vision_sdk\bin\tda3xx-evm\sbl_boot

    o AppImage_UcEarly_BE

    o AppImage_UcLate_BE

- Refer section 3.5.3 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

### 3.10.5 Run

Press Power On Reset button on Tda3x EVM. Make sure QSPI boot is selected as mentioned in section 0

Pass criteria

- Display should flash up with preview in 1 sec approx
- Use case 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.

## 3.11 Surround View Fast Boot Use case

The following compile option in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk should be used to enable fast boot for 3D Surround View:

```
SRV_FAST_BOOT_INCLUDE=yes
```

This option builds the Application Image involving only ipu1_0 and DSP_1 cores.

**Important Note:** Ensure the post-calibration tables are present on the SD Card that is being used to run the usecase. The details of such tables can be found in the relevant SRV UserGuides.

## 3.12 Surround View Use case under 128 MB DDR configuration

The following changes are required to be made before building and running the Surround View use cases (calibration, 2D and 3D) with the 128 MB DDR configuration:

1. Set 'DDR_MEM=DDR_MEM_128M' in the platform 'cfg.mk' file
2. Remove EVE1, IPU1_1 and DSP2 from the build in the platform 'cfg.mk' file:
   - PROC_EVE1_INCLUDE=no

- PROC_IPU1_1_INCLUDE=no
- PROC_DSP2_INCLUDE=no
- ECC_FFI_INCLUDE=no

3. Make the following changes to the memory sections in the `\vision_sdk\$(MAKEAPPNAME)\build\tda3xx\mem_segment_definition_128mb.xs` file:

- `EVE1_VECS_SIZE            =   1*KB;`
- `EVE1_CODE_SIZE            = 128*KB;`
- `EVE1_DATA_SIZE            = 128*KB;`
- `IPU1_1_CODE_SIZE          = 128*KB;`
- `IPU1_1_DATA_SIZE          = 128*KB;`
- `IPU1_0_CODE_SIZE          =  10*MB;`
- `IPU1_0_DATA_SIZE          =  10*MB;`
- `DSP1_CODE_SIZE            =   1*MB;`
- `DSP1_DATA_SIZE            =  11*MB;`
- `DSP2_CODE_SIZE            = 128*KB;`
- `DSP2_DATA_SIZE            = 128*KB;`
- `SR1_FRAME_BUFFER_SIZE     =  90*MB;`
- `SR1_BUFF_ECC_ASIL_SIZE    =   4*KB;`
- `SR1_BUFF_ECC_QM_SIZE      =   4*KB;`
- `SR1_BUFF_NON_ECC_ASIL_SIZE =  4*KB;`

4. Increase the size of SR1 heap to 90 MB in the memory map file (.xs).

5. Exclude the use cases which are out of scope for 128 MB build by setting `UC_<use case name>=no` in the platform `uc_cfg.mk` file as shown below, so that the sizes of the IPU1_0 and DSP1 code and data fits within the corresponding memory segments defined in the `\vision_sdk\$(MAKEAPPNAME)\build\tda3xx\mem_segment_definition_128mb.xs` file.

The below use-cases are validated with 128 MB memory configuration.
```
UC_saveDisFrame=yes
UC_srv_calibration=yes
UC_iss_mult_capture_isp_simcop_sv_tda3xx=yes
UC_iss_mult_capture_isp_dewarp_3dsv_tda3xx=yes
UC_iss_capture_isp_simcop_display=yes
UC_vip_single_cam_view=yes
```

```
UC_vip_single_cam_view_dsswb=yes
UC_vip_single_cam_display_metadata=yes
```

## 3.13  Build IPU in SMP mode

By default Vision SDK build enable both core0 and core1 of IPU sub-system in non-SMP mode.  Where both the cores run independently with two separate binaries,

To build IPU sub-system in SMP mode, modify
\vision_sdk\apps\configs\tda3xx_evm_bios_all\cfg.mk as below

> PROC_IPU1_1_INCLUDE=no

> IPU1_SMP_BIOS=yes

Make Clean and Build

## 3.14  DCAN Usecase

The DCAN module is complaint to CAN 2.0B protocol specification and can support bit rates up to 1 Mbit/s. Connections to the CAN network is performed through external (for the device/SoC) transceivers (available on the EVM).

### 3.14.1  Build

The following compile option in "\vision_sdk\apps\configs\tda3xx_evm_bios_all\cfg.mk" should be used to enable DCAN usecase:

```
DCAN_INCLUDE=yes
```

**NOTE:** DCAN demo/Usecase is only supported and validated on TDA3xx EVM.

### 3.14.2  Hardware Set-up

To run this Usecase, an external CAN node- which can be a PC CAN emulation tool or another TDA3xx EVM (running an appropriate application to send/receive CAN messages), needs to be connected to the CAN1 bus of the TDA3xx EVM. Configured bit rate is of 1 Mbit/s and same should be configured for other CAN node. PCAN- a PC CAN emulation tool is used to validate this example as shown in the following
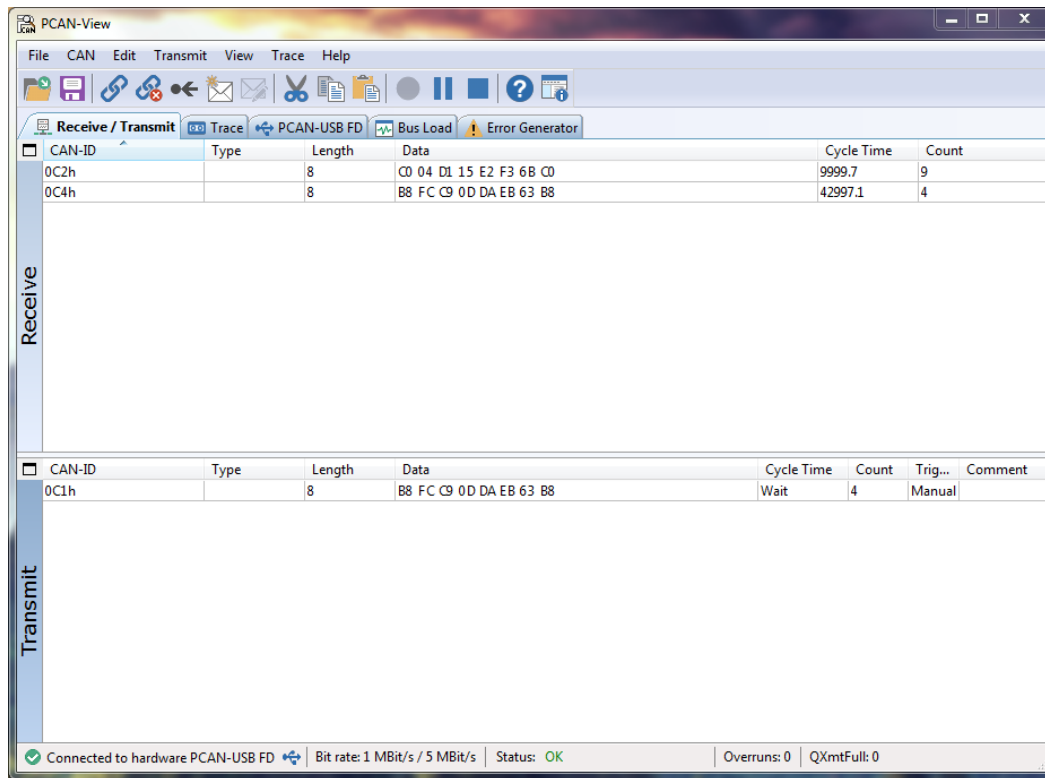
figure:
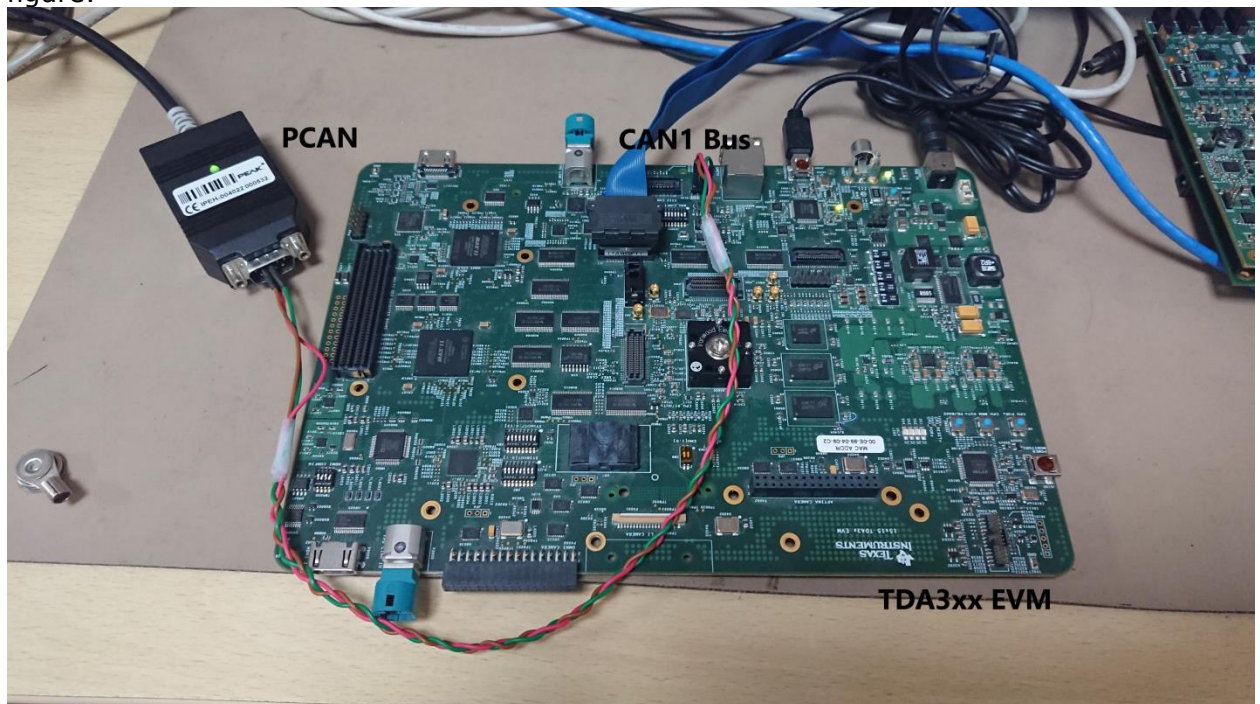


## Figure: DCAN Usecase connections

**NOTE:** For more details about PCAN, please refer to https://www.peak-system.com/PCAN-USB-FD.365.0.html?&L=1.

### 3.14.3 Run

There is no specific option to run this Usecase. This Usecase runs along with all other demo applications/usecases.

This application sends out a periodic control message with ID of '0xC2' at an interval of 10 seconds. On reception of the control message (with ID of '0xC1'), it sends out an ACK message with ID of '0xC4'. Same is shown in below figure:

**Figure: PCAN View showing sent/received messages**

asdasd

# 4 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 |
| 7.0 | 18th March 2016 | Updated for release 2.9 |
| 8.0 | 18th October 2016 | Minor changes and updated for 2.11 |
| 9.0 | 8th February 2017 | Minor changes and updates for vision sdk 2.12 |
| 10.0 | 15th Apr 2017 | Updated for 128M build |
| 11.0 | 19th June 2017 | Updated linux installer |
| 12.0 | 29th June, 2017 | Updated for Processor SDK Vision 3.0 release |
| 13.0 | 13th October 2017 | Updated for Processor SDK Vision 3.01 release |
| 14.0 | 20th December 2017 | Update for Processor SDK vision 3.02 release |
| 15.0 | 29th Mar 2018 | Updated 128MB map file change |
| 16.0 | 26th Jun 2018 | IPU SMP mode support details included |
| 17.0 | 2nd July 2018 | Added details about DCAN demo application. |

**«««« § »»»»**