

# Vision SDK TDA2Ex

(v02.08.00)

## User Guide

**Copyright © 2014 Texas Instruments Incorporated. All rights reserved.**

Information in this document is subject to change without notice. Texas Instruments may have pending patent applications, trademarks, copyrights, or other intellectual property rights covering matter in this document. The furnishing of this documents is given for usage with Texas Instruments products only and does not give you any license to the intellectual property that might be contained within this document. Texas Instruments makes no implied or expressed warranties in this document and is not responsible for the products based from this document.

## 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](http://www.ti.com/sc/docs/stdterms.htm)

Mailing Address:

Texas Instruments  
Post Office Box 655303  
Dallas, Texas 75265

Copyright © 2014, Texas Instruments Incorporated

---

**TABLE OF CONTENTS**

---

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
1.1	References .....	4
1.2	Directory Structure.....	5
<b>2</b>	<b>Installation Overview .....</b>	<b>6</b>
2.1	PC Requirements.....	6
2.2	Software Requirements.....	6
2.3	Hardware Requirements.....	7
2.4	Software Installation .....	11
<b>3</b>	<b>Build and Run .....</b>	<b>12</b>
3.1	Overview of application in release .....	12
3.2	Building the application.....	13
3.3	UART settings .....	15
3.4	Load using SD card .....	17
3.5	Load using QSPI.....	19
3.6	Load using NOR .....	23
3.7	Load using CCS.....	27
3.8	Run the demo .....	34
<b>4</b>	<b>Frequently Asked Questions.....</b>	<b>37</b>
4.1	Hardware Board Related FAQs .....	37
4.2	Build, install, load, run related FAQs .....	37
4.3	PM and resource related FAQs.....	40
<b>5</b>	<b>Directory Structure Details.....</b>	<b>41</b>
<b>6</b>	<b>Revision History .....</b>	<b>42</b>

## 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 shows customer how to effectively use different sub-systems in 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 TDA2Ex EVM. Refer to VisionSDK\_UserGuide\_TDA2xx.pdf and VisionSDK\_UserGuide\_TDA3xx.pdf for respective EVM related Hw/SW setup

### 1.1 References

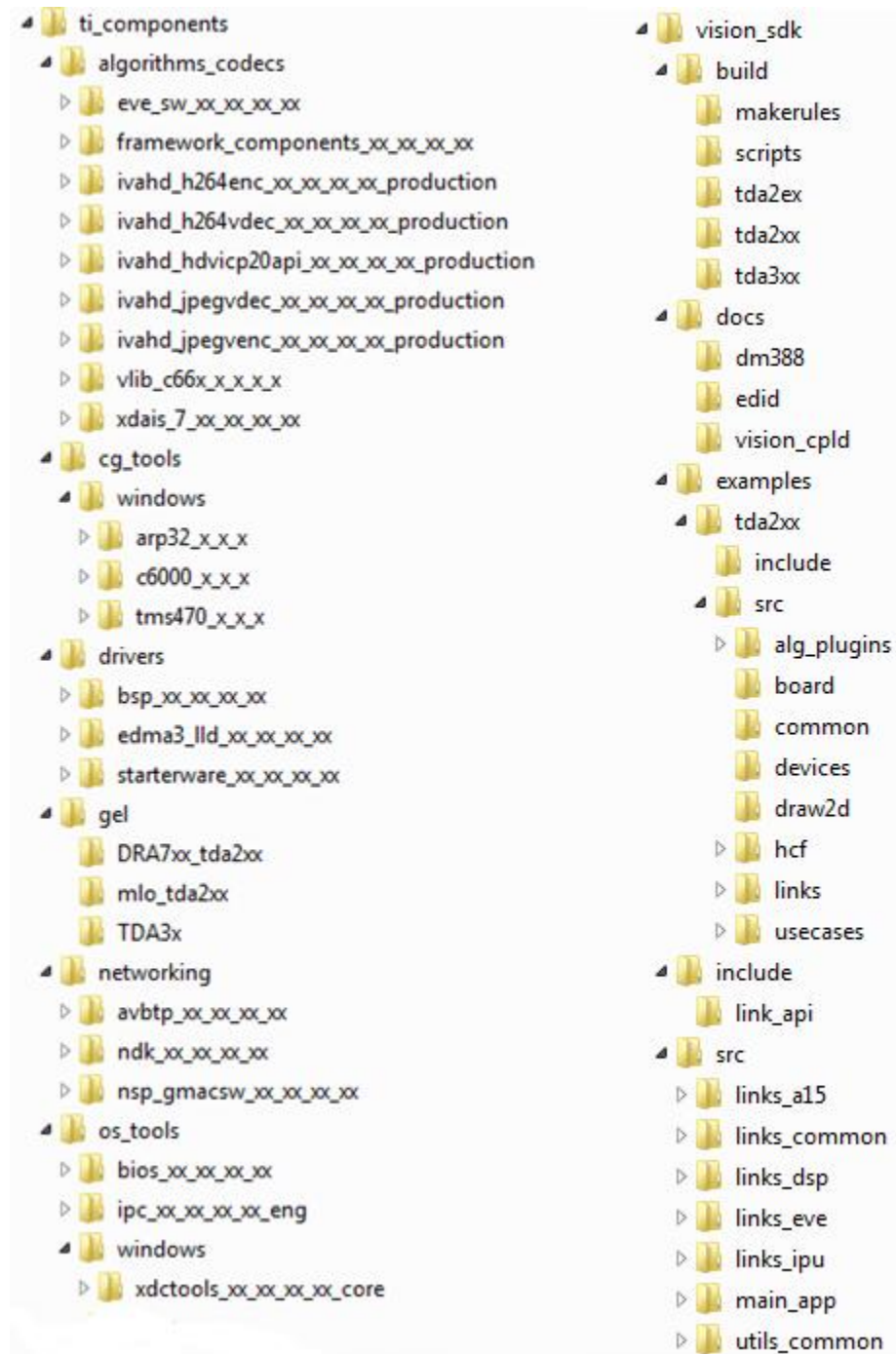
Refer the below additional documents for more information about Vision SDK

Document	Description
VisionSDK_ReleaseNotes.pdf	Release specific information
VisionSDK_UserGuide_TDA2Ex.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; there will be two main directories installed – vision\_sdk and ti\_components.

**NOTE: vision\_sdk\examples\tda2xx contains examples for both TDA3xx and TDA2xx,TDA2Ex**



## **2 Installation Overview**

This chapter provides a brief description on 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 A15 Compiler, Linker**

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

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

The tools need to be installed in "<install dir>/ti\_components/cg\_tools/windows/gcc-arm-none-eabi-4\_7-2013q3" location.

**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**

#### **2.2.2 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 should be installed.

[http://processors.wiki.ti.com/index.php/Download\\_CCS](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

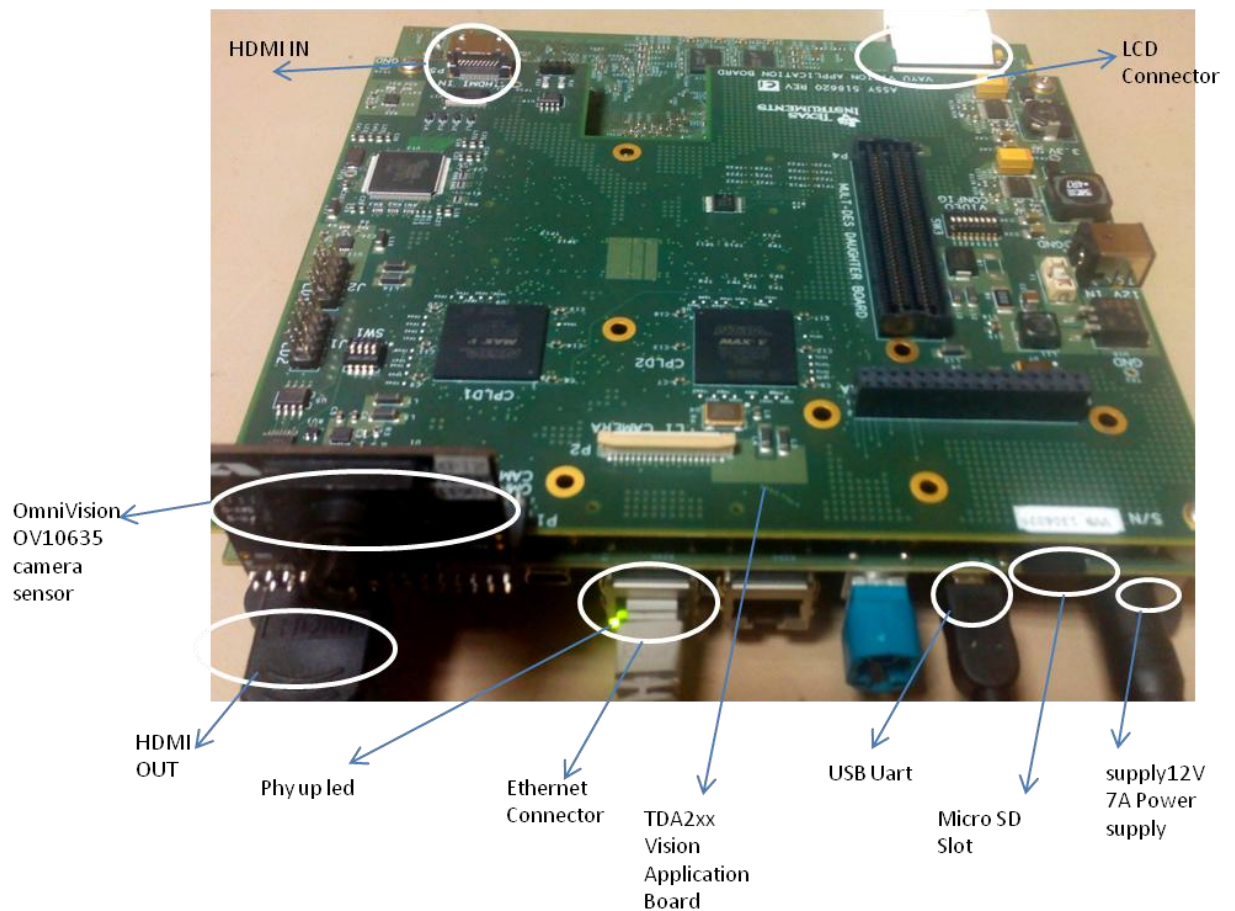
Hardware setup for Single Camera View (SCV) and LVDS Multi Camera View (LVDS MCV) use-case is described in this section

### 2.3.1 SCV Use-case Hardware Setup

SCV use-case needs the below hardware

1. TDA2Ex EVM (Rev D)
2. TDA2xx Vision Application Board (Rev C)
3. OV10635 Sensor (for SCV only)
4. WVGA LCD DC from Spectrum Digital (part #703663) OR
5. HDMI 1080p60 capable Display Monitor

Setup is shown below



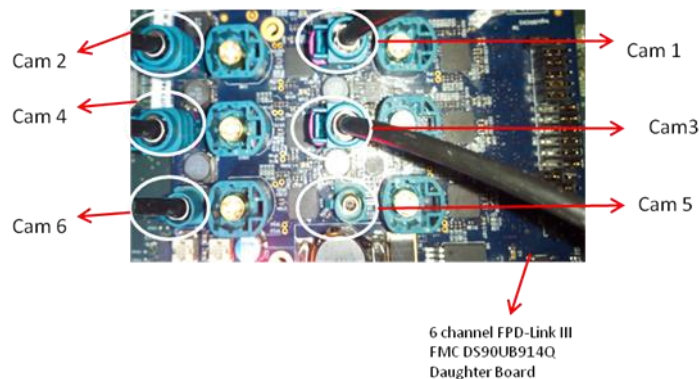
## LVDS MCV Use-case Hardware Setup

LVDS MCV use-case needs the below hardware

1. TDA2Ex EVM (Rev D)
2. TDA2xx Vision Application Board (Rev C)
3. 6 channel FPD-Link III FMC SV600964 Daughter Board (Rev E1)
4. 4 x DS90UB913A EVMs (Rev A)
5. 4 x OV10635 Sensor (additional 5th DS90UB913A EVM, OV10635 sensor and cable would be required in order to run the Surround view demo).
6. 4 x Rosenberger HSD connectors and cables
7. WVGA LCD DC from Spectrum Digital (part #703663) OR
8. HDMI 1080p60 capable Display Monitor

The LVDS MCV use case setup is shown in the snapshots below:

### 2.3.1.1 DeSerializer board

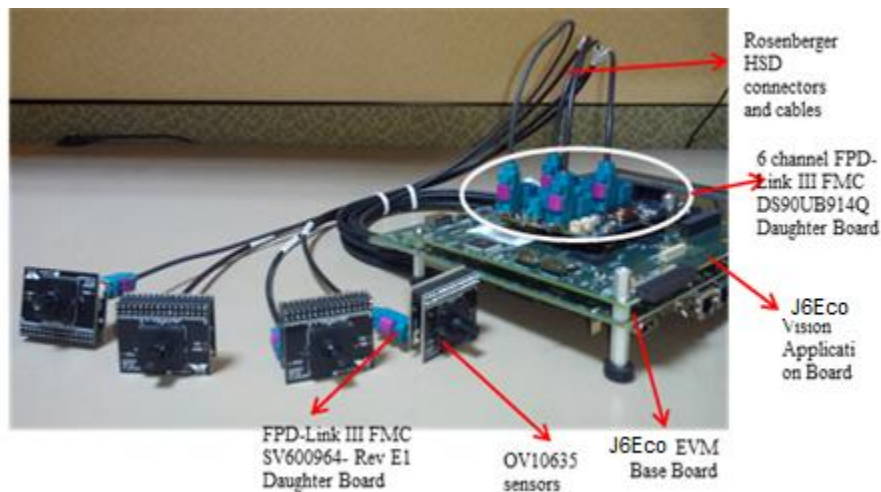


**IMPORTANT NOTE:** Camera 1, Camera 2, Camera 3 and Camera 4 are used for 4 channel LVDS use-case and **MUST** be connected as shown in above figure.

**IMPORTANT NOTE:** There is a board modification required for LVDS capture on TDA2EX boards. Refer to BSP user guide under section "**Changes needed for 4 Channel Multi-deserializer on TDA2EX EVM**" for more information.



### 2.3.1.2 Complete LVDS Setup



### 2.3.2 Capture Pin Settings

Video Config pins needs to set for different capture inputs



**VIDEO CONFIG switch settings (SW3 on TDA2xx Vision Application Board (set for Ov10635 in Original version of CPLD))**

Capture Type	Hardware controlled pin settings Vision Application Board (Rev C CPLD) (default cpld image)								Software controlled pin settings New Version Of CPLD flashed (cpld_1_cam3_shift.pof)							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
OV10635	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
LVDS	OFF	OFF	ON	OFF	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
HDMI	OFF	OFF	ON	ON	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Cpld image is required for VIP input Muxing,

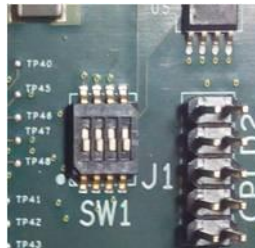
With the cpld\_1\_cam3\_shift.pof and later the software control will work,

On default Rev C board the captured image won't be proper. Either program the Cpld with new image or use hardware controlled pin settings.

### 2.3.3 EDID Programming for HDMI Capture.

EDID information needs to be programmed on the EEPROM present on Vision Application Board. This is required for the HDMI capture source to recognize the format and resolution in which to send data to the TDA2Ex SoC. If this step is not done or if this step fails, then TDA2Ex SoC will not be able to receive data via HDMI capture source

1. Change pins 1 and 2 of SW1 (on vision application board, near CPLD2 connector) to ON.



2. Connect usb cable from board to PC and setup UART for logs (Ref [Uart settings](#))
3. Connect to a15 core and load 'edid\_programming\_1080p\_60.xa15fg' binary using CCS (Refer [Load using CCS](#) till step 8).  
Binary placed under  
vision\_sdk\docs\edid\edid\_programming\_1080p\_60.xa15fg
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 SW1 to OFF (towards numbers 1 and 2).

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

**IMPORTANT NOTE:** If LVDS Setup is connected, EDID programming may fail. Disconnect the LVDS Daughter board and then program EDID. If EDID is not programmed correctly detect video will fail when HDMI capture is done.

## 2.4 Software Installation

vision\_sdk\_02\_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\_02\_xx\_xx\_xx would have got created in the installation path.

### 2.4.1 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 as examples

- Single channel capture use-cases
  - Single channel capture, display use case
  - Single channel capture, frame copy algorithm on DSP1, display use case
  - Single channel capture, lane detect algorithm on DSP1, display use case
- Multi-channel LVDS capture use-cases
  - 4Ch LVDS capture, VPE scale, Timestamp Sync, 2x2 DMA SW Mosaic on IPU1-0, display use case

Refer to VisionSDK\_DataSheet.pdf for detailed description of each use-case.

The demos support following devices as capture source

- OV10635 Sensor 720P30 (default) (VIP use-cases only)
- HDMI Capture 1080P60

The demos support following devices as display devices

- LCD 7-inch 800x480@60fps
- LCD 10-inch 1280x720@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 TDA2Ex platform, set VSDK\_BOARD\_TYPE := TDA2EX\_EVM in vision\_sdk\Rules.make

**NOTE:** For LVDS usecase please set NDK\_PROC\_TO\_USE= none.

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\tda2ex-evm  
vision_sdk_a15_0_debug.xa15fg  
vision_sdk_c66xdsp_1_release.xe66  
vision_sdk_ipul_0_release.xem4  
vision_sdk_ipul_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.

```
PROC_DSP1_INCLUDE=yes  
PROC_A15_0_INCLUDE=yes
```

```
PROC_IPU1_0_INCLUDE=yes  
PROC_IPU1_1_INCLUDE=yes
```

The build time can be made faster by following below steps.  
BUILD\_DEPENDANCY\_ALWAYS = no in Rules.make

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

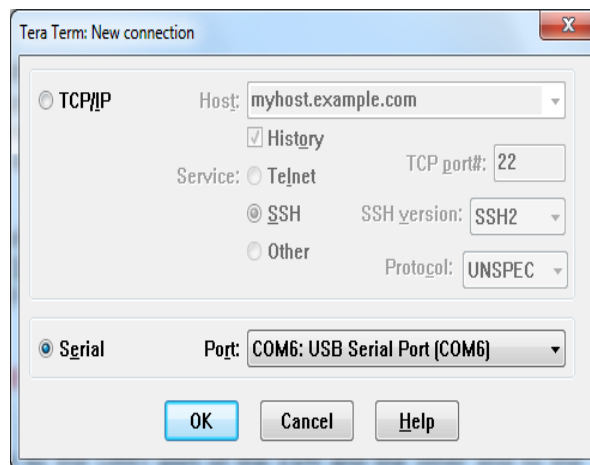
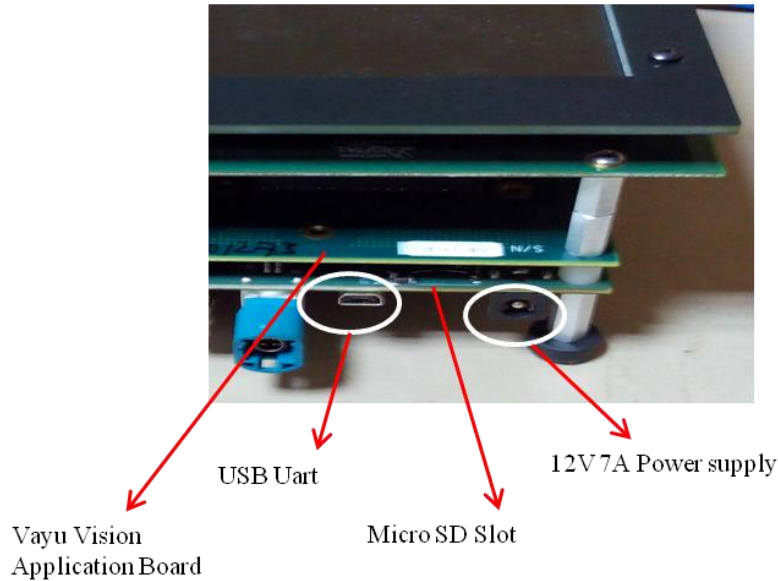
If this "`gmake -s depend`" not done then build or execution may fail.

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.



Tera Term: Serial port setup

Port: COM6

Baud rate: 115200

Data: 8 bit

Parity: none

Stop: 1 bit

Flow control: none

Transmit delay

0 msec/char 0 msec/line

OK

Cancel

Help



### 3.4 Load using SD card

NOTE: The application can be run using SD card and SD card boot or using CCS. This section shows how to run using SD card boot.

Application image is run on the SoC via Secondary Boot Loader (SBL) present in SD card.

#### 3.4.1 Option 1: Steps to prepare a bootable SD card

- Ensure Empty SD card (at least 256MB, preferably 4GB SDHC) is available.
- Ensure SD memory card reader is available.
- Create a primary FAT partition on MMC/SD card (FAT32 format with sector size 512) and mark it as Active. A partition manager utility has to be used for the same.
- Format SD card from DOS command line as below.  
"format <drive> /A:512 /FS:FAT32"

**Make SD card partition as active using below tool**

<http://www.pcdisk.com/download.html>

**IMPORTANT NOTE:** Create a primary FAT partition on MMC/SD card (FAT32 format with sector size 512 bytes mark the partition as active.

#### 3.4.2 Option 2: Steps to prepare a bootable SD card using DISKPART

- Open windows 7 Command prompt and Run as Administrator mode
  - Enter command "diskpart.exe"  
C:\Windows\system32>diskpart.exe will take you DISKPART prompt
- Warning:** Enter below command carefully w.r.t your computer/laptop SD card disk number. Choosing wrong disk number may delete data present in other drive

To list all disk drive present on computer

DISKPART> list disk

Select the SD card disk number, in my case it is disk 1

DISKPART> select disk 1

Now all next command applicable only to disk 1(SD card)

Delete entire partition

DISKPART> clean

To create Primary partition

DISKPART> create partition primary

To list partition

DISKPART> list partition

Select partition

DISKPART> select partition 1

To list volume

DISKPART> list volume

Select volume associated with SD card, In my case its 3

DISKPART> select volume 3

Format SD card, please wait this may take few seconds

DISKPART>format quick fs=fat32 unit=512 label=SD\_BOOT

Make disk active

DISKPART> active

To exit utility

DISKPART> exit

### 3.4.3 Steps to generate MLO

NOTE: SBL MLO image is built from starterware package.

To build MLO Run the command **gmake -s sbi\_sd** from vision\_sdk root dir

And run the **sbi\_mlo\_create.bat** placed at vision\_sdk\build\scripts

This generates an MLO under vision\_sdk\build\scripts\mlo

To build the mlo for different memory map, select required configuration in Makefile under vision\_sdk (follow comments from Makefile under [SBL build Targets](#)).

### 3.4.4 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.bat

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

REM is the comment used to comment out in .bat file

REM set App\_IPU1\_CPU1 is sufficient

### 3.4.5 SD Card setup

Once the AppImage and MLO are generated , Copy the MLO and AppImage at root folder of formatted SD Card

### 3.4.6 Hardware Pin settings for SD Boot

Make sure the Boot Mode Select Switch is set for the SD boot mode **on TDA2Ex Base EVM**. This is done by setting the pins SYSBOOT (SW2+SW3) [0:15] to the below shown position



- 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)

Pre-build MLO is located at C:\VISION\_SDK\_XX\_XX\_XX\_XX\vision\_sdk\prebuilt\tda2ex-evm\sbl\_boot

## 3.5 Load using QSPI

### 3.5.1 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.bat** placed at vision\_sdk\build\scripts

This generates all required tools under vision\_sdk\build\scripts\qspi

1. sbl\_a15host\_release.xa15fg
2. qspiFlashWriter\_m4\_release.xem4
3. sbl\_qspi

To build the qspi for different memory map, select required configuration in Makefile under vision\_sdk (follow comments from Makefile under **SBL build Targets**).

**IMPORTANT NOTE:** For QSPI to work on NOR modified board, refer section "TDA2Ex Board Modification for NOR and QSPI BOOT Mode" under starterware user guide.

### 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 the batch file shown below  
vision\_sdk\MulticoreImageGen.bat

**IMPORTANT NOTE:** If some cores are disabled from build, comment them from MulticoreImageGen.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 A15.

Select CortexA15\_0, navigate to Scripts->TDA2Ex MULTICORE Initialization  
TDA2Ex\_MULTICORE\_EnableALLCores

2. Connect M4 (IPU)

Halt A15 core, and Load image on M4

**C:\VISION\_SDK\_XX\_XX\_XX\_XX\vision\_sdk\build\scripts\qspi\  
qspiFlashWriter\_m4\_release.xem4**

Run the core.

Console outputs

[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\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\*\*\*\*\*

### 3. Reset the board and Repeat step 1 and 2.

Console outputs

```
[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\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*****
```

### 4. Reset the board and Repeat step 1 and 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\tda2ex-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: 2
```

Load Options:

- 
- 0 -> fread using code (RTS Library)
  - 1 -> load raw using CCS (Scripting console)

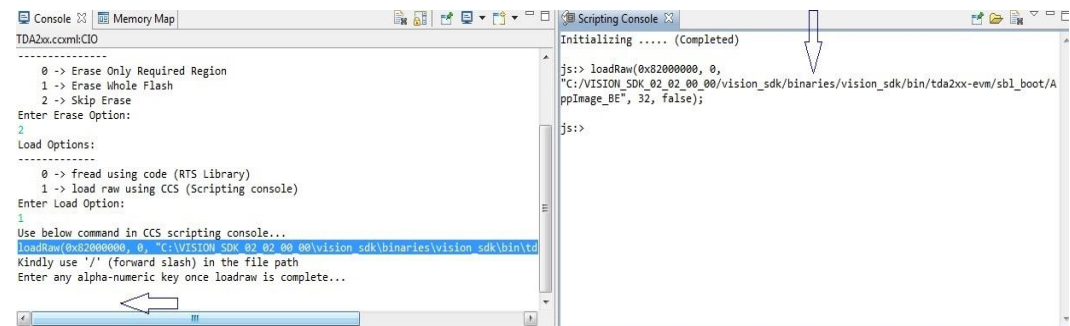
Enter Load Option: **1**

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

**loadRaw(0x82000000, 0,  
"C:/VISION\_SDK\_XX\_XX\_XX\_XX/vision\_sdk/binaries/vision\_sdk  
/bin/tda2ex-evm/sbl\_boot/AppImage\_BE", 32, false);**

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\*\*\*\*\*

### 5. On completion change the pin setting

Boot mode:



- 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)

## 3.6 Load using NOR

### 3.6.1 Steps to generate qspi writer tools

NOTE: SBL nor image is built from starterware package.

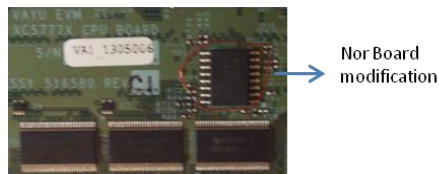
To build nor Run the command **gmake -s sbl\_nor** from vision\_sdk root dir  
And run the **sbl\_nor\_create.bat** placed at vision\_sdk\build\scripts

This generates all required tools under vision\_sdk\build\scripts\nor

1. nor\_flash\_writer\_m4\_release.xem4
2. sbl\_nor

To build the nor for different memory map, select required configuration in Makefile under vision\_sdk (follow comments from Makefile under **SBL build Targets**).

**IMPORTANT NOTE:** There is a board modification required for NOR boot.  
Refer “**TDA2Ex Board Modification for NOR and QSPI BOOT Mode**” under starterware user guide.



**IMPORTANT NOTE: NOR Boot takes ~90min for completion**

### 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 the batch file shown below  
vision\_sdk\MulticoreImageGen.bat

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

REM is the comment used to comment in .bat file

REM set App\_IPU1\_CPU1 is sufficient

### 3.6.3 Flashing steps

Flashing pin settings:



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

(VISION\_SDK\_xx\_xx\_xx\_x\ti\_components\drivers\starterware\_xx\_xx\_xx\_xx\bootloader)

For loading binaries using CCS refer [Load using CCS](#) till step 8.

1. Connect A15.

Load image

**C:\VISION\_SDK\_XX\_XX\_XX\_XX\vision\_sdk\build\scripts\nor\  
nor\_flash\_writer\_m4\_release.xem4**

Run the core.

Console outputs

*[CortexA15\_0] Starting NOR Flash Writer.*

*CFI Query...passed.*

*NOR Initialization:*

*Command Set: Spansion*

*Manufacturer: SPANSION*

*Size: 0x40 MB*

*Enter the File Name*

**C:\VISION\_SDK\_XX\_XX\_XX\_XX\vision\_sdk\build\scripts\nor\sbl\_n  
or**

*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*

*Erasing the NOR Flash*

*Erased through 0x8020000*

*:*

*:*

*Erased through 0x9000000*

2. Reset the board and Repeat step 1.

Console outputs

*[CortexA15\_0] Starting NOR Flash Writer.*

*CFI Query...passed.*

*NOR Initialization:*

*Command Set: Spansion*

*Manufacturer: SPANSION*

*Size: 0x40 MB*

*Enter the File Name*

**C:\VISION\_SDK\_XX\_XX\_XX\_XX\vision\_sdk\build\scripts\nor\sbl\_n  
or**



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**

Reading xxxx bytes from file...

Read XXXX bytes [100%] from file. Done!!

Writing 69156 bytes to NOR...

NOR Write OK through 0x8008000.

NOR Write OK through 0x8010000.

NOR Write OK through 0x8010E24.

Done.

!!! Successfully Flashed !!!

NOR boot preparation was successful!

### 3. Reset the board and Repeat step 1.

[CortexA15\_0] Starting NOR Flash Writer.

CFI Query...passed.

NOR Initialization:

Command Set: Spansion

Manufacturer: SPANSION

Size: 0x40 MB

Enter the File Name

**c:\VISION\_SDK\_XX\_XX\_XX\_XX\vision\_sdk\binaries\vision\_sdk\  
bin\tda2ex-evm\sbl\_boot\AppImage\_LE**

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

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)

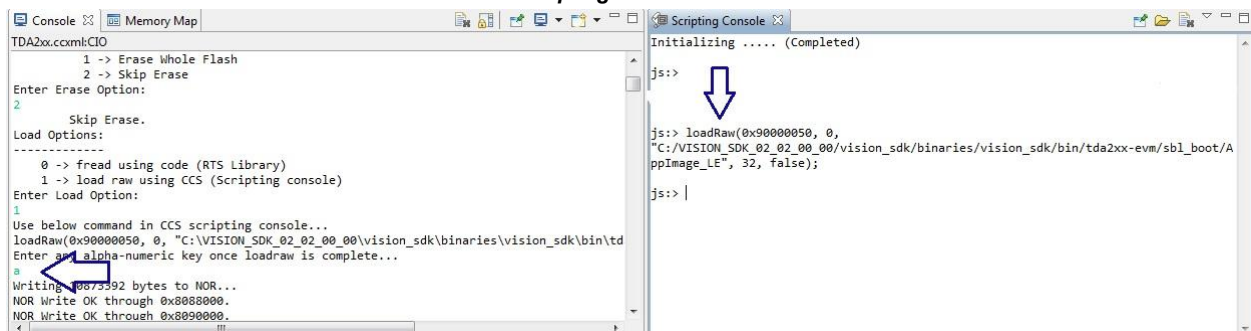
Enter Load Option: **1**

Open Scripting console window by clicking "Menu -> View -> Scripting console" and enter below command on scripting console as shown 3.6.3.1

**loadRaw(0x90000050, 0,  
"C:/VISION\_SDK\_XX\_XX\_XX\_XX/vision\_sdk/binaries/vision\_sdk  
/bin/tda2ex-evm/sbl\_boot/AppImage\_LE", 32, false);**

In CCS console Enter any alpha-numeric key once loadraw is complete... as shown in 3.6.3.1

### 3.6.3.1 CCS console and scripting console



Writing 10873392 bytes to NOR...  
NOR Write OK through 0x8088000.  
NOR Write OK through 0x8090000.

NOR Write OK through 0x8AD8000.  
NOR Write OK through 0x8ADEA30.  
Done.

!!! Successfully Flashed !!!

NOR boot preparation was successful!

4. On completion change the pin setting

Boot mode:



- 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)

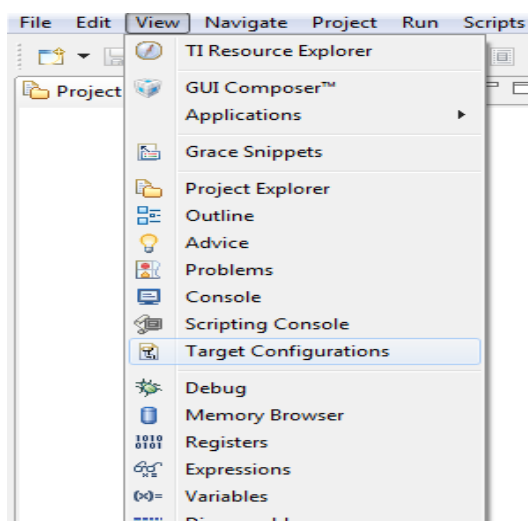
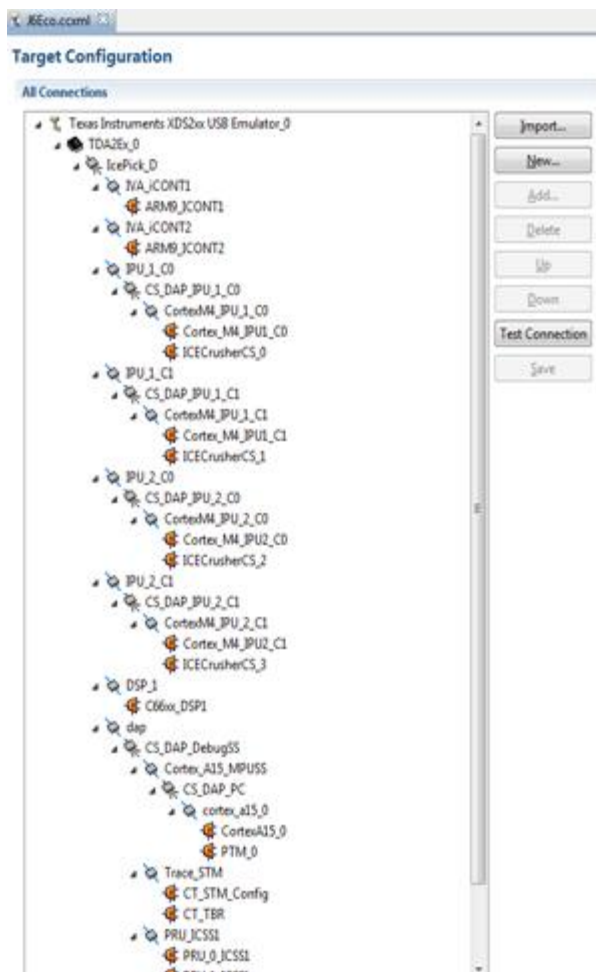
### 3.7 Load using CCS

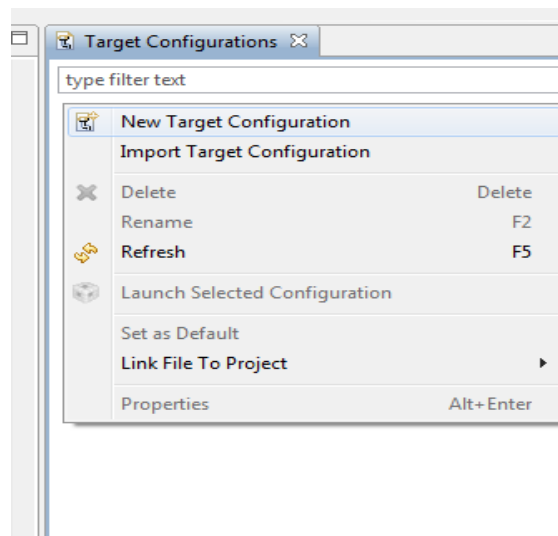
After installing CCS, follow below steps to complete the platform setup,

1. Install Chip Support Package available at location  
<Install\_dir>\ti\_components\ccs\_csp\tda3xx\CCS\_CSP\_TDA2Ex\_SR1.0\_NDA\_TRM\_vK\_gels2.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,

- TDA2Ex\_multicore\_reset.gel
    - o Set VISION\_SDK\_CONFIG to 1 always (even for 256MB mode)
    - o Set VISION\_SDK\_CONFIG\_256M for 256MB mode
- 
2. CCS Target Configuration creation:
    - a. Open "Target Configurations" tab, by navigating through the menu "View ->Target Configurations".
    - b. Create a new Target Configuration (TDA2Ex 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 "TDA2Ex".
    - e. ByPass unused cores. Click on the core which needs to be bypassed and check ByPass under Bypass Properties.  
The settings is under advance setup tab






## Basic

### General Setup

This section describes the general configuration about the target.

Connection Spectrum Digital XDS560V2 STM USB Emulator ▼

Board or Device tda 

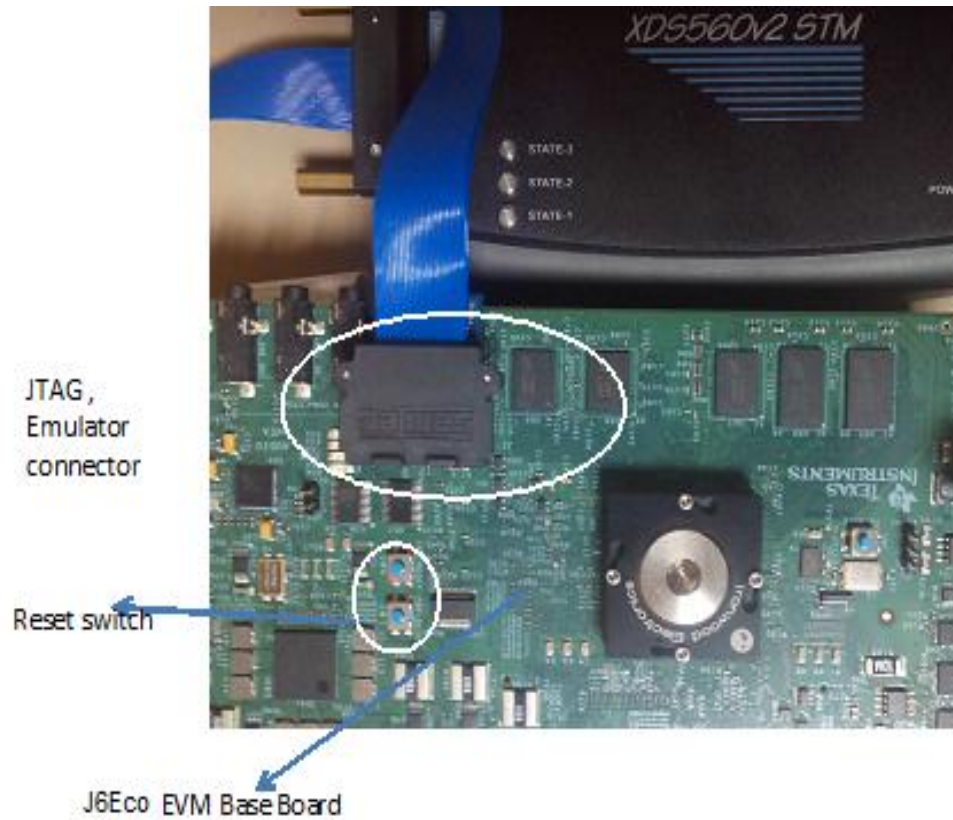
☒ TDA2Ex

☐ TDA3x

TDA2Ex NDA device ▲

3. Connect JTAG to the board.

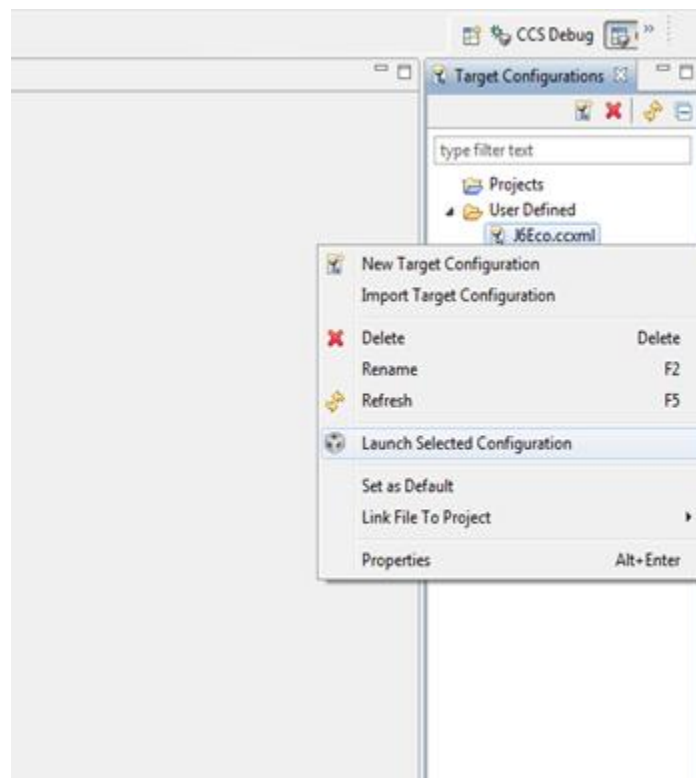
**IMPORTANT NOTE:** There are two JTAG connectors on the board. The one shown below **MUST** be used for CCS debug.

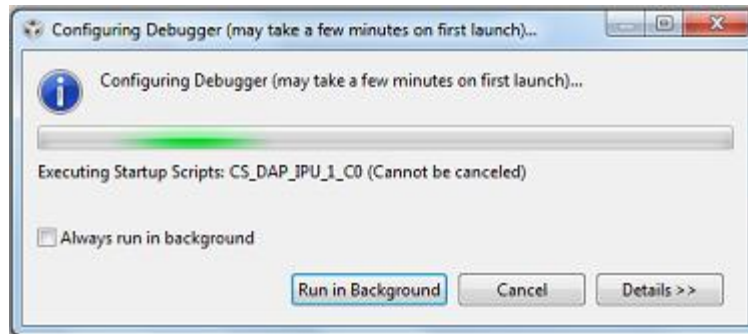


4. Reset EVM through the blue button (SW4, out of two, the one away from the JTAG).



5. Now launch the previously created TDA2Ex Target Configuration.

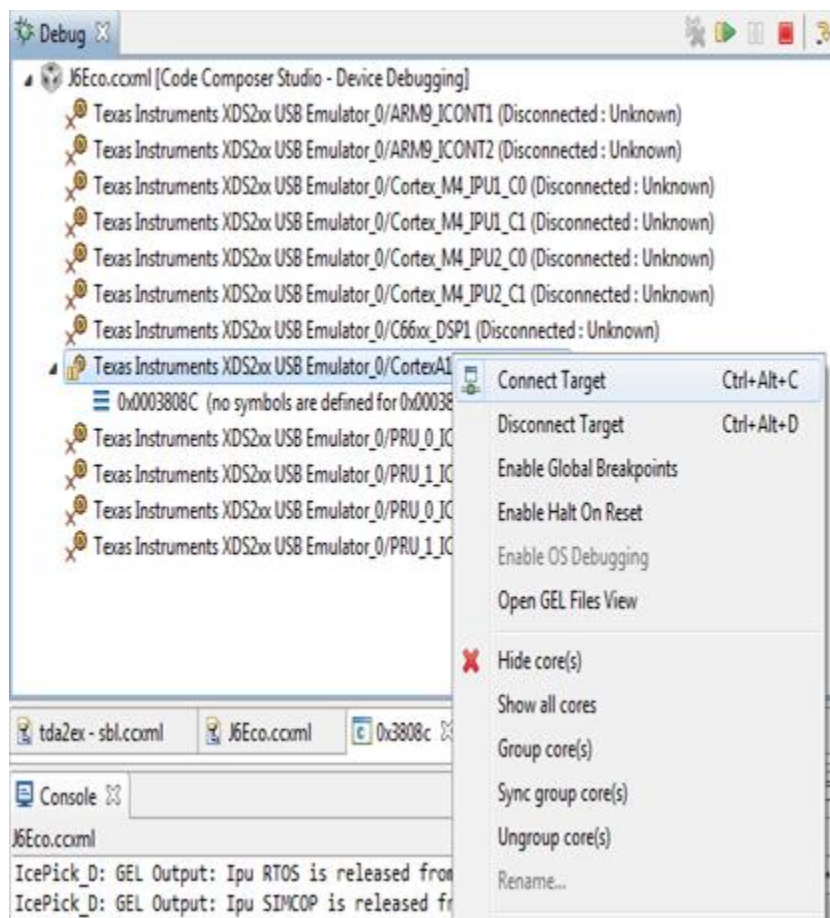




6. Once the target configuration is launched successfully, the following log should be observed on the CCS console:

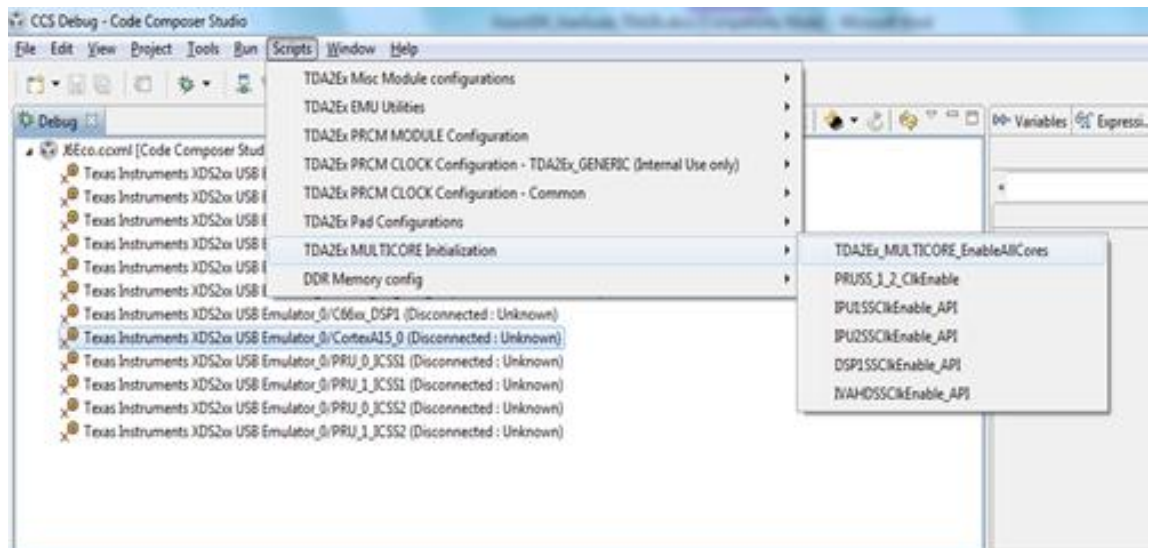
CortexA15\_0: GEL Output: --->>> TDA2Ex Cortex A15 Startup Sequence DONE! <<<--

7. Connect to core CortexA15\_0.



8. On successful connect, the following log appears on CCS console:  
CortexA15\_0: GEL Output: --->>> TDA2Ex Target Connect Sequence DONE !!!!! <<<---
9. Select CortexA15\_0, navigate to Scripts->TDA2Ex MULTICORE Initialization  
TDA2Ex\_MULTICORE\_EnableALLCores

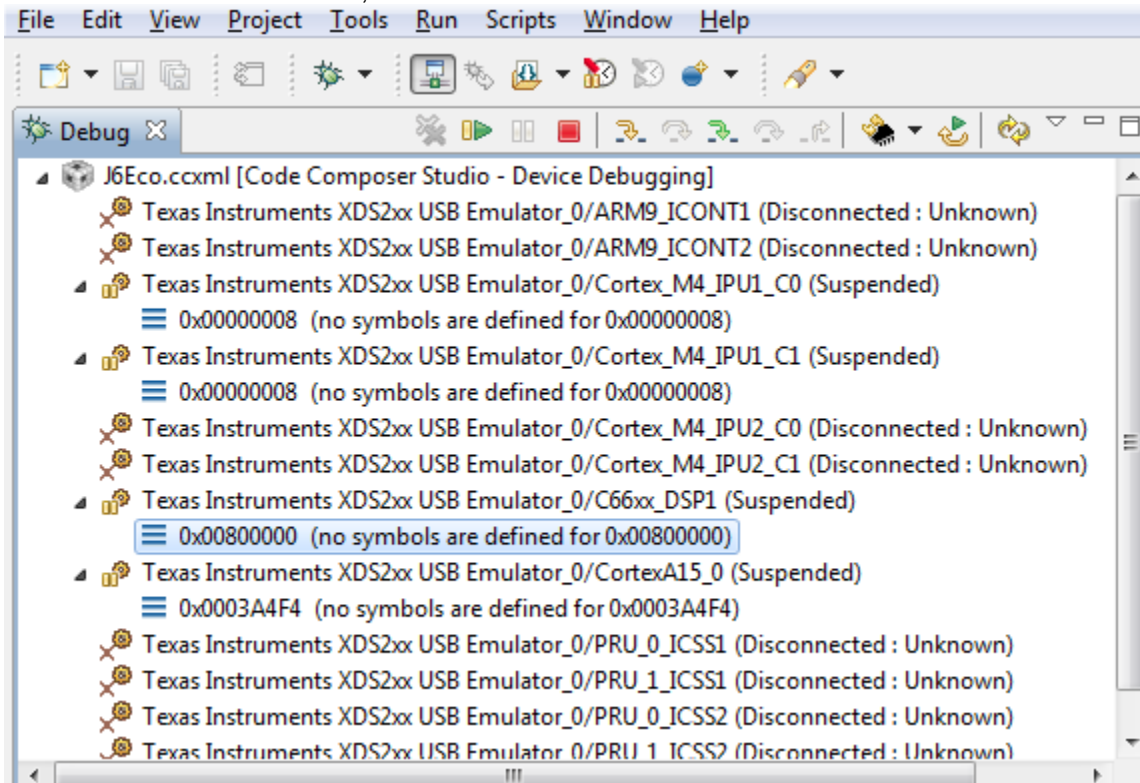




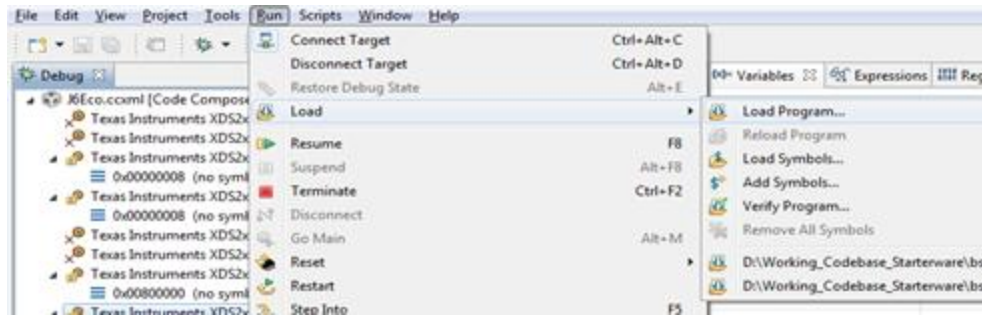
10. On successful script execution, the following log appears on CCS console:  
CortexA15\_0: GEL Output: --->>> PRUSS 1 and 2 Initialization is in complete ... <<<---

11. Now connect the core shown below,  
C66xx\_DSP1,Cortex\_M4\_IPU1\_C0, Cortex\_M4\_IPU1\_C1.

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



13. On the cores load the binaries as mentioned below



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".  
 On CortexA15\_0, load the binary, "vision\_sdk\_a15\_0\_debug.xa15fg"

**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.

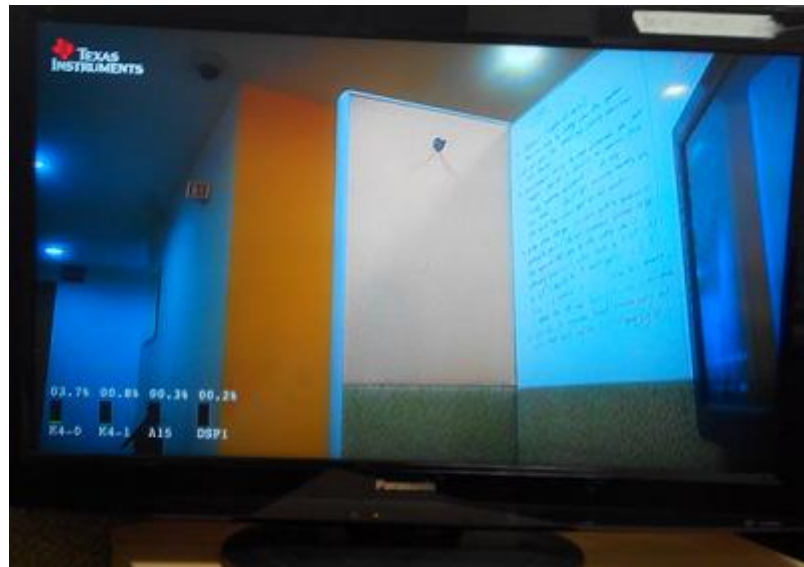
### 3.8 Run the demo

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 uart menu.

**IMPORTANT NOTE:** Make sure you select SCV (1Ch VIP capture) use-case or LVDS (4Ch VIP capture) use-case depending on the camera that is connected

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

- a. SCV use-cases:



- b. LVDS use-cases:



c. Surround View



## 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
Q. I selected a use-case and it displays green or blue color screen instead of video display on LCD
Please ensure you have selected proper display using s (setting option), Display settings.

### 4.2 Build, install, load, run related FAQs

Q. How do I speed up build and load time ?
<b>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.</b>
One or more of the following steps can be used to speed up build time,
<ol style="list-style-type: none"> <li>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.</li> <li>2. Edit Rules.make and set BUILD_DEPENDANCY_ALWAYS=no around line ~472.</li> </ol>

When BUILD\_DEPENDANCY\_ALWAYS=no, an additional step of (g)make -s depend needs to be 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 in 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.

<p>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</p>
<p>Q. Can the version of tools or other components be changed? If yes, how?</p>
<p>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.</p>
<p>Q. Is Linux based build supported?</p>
<p>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.</p>
<p>Q. In Surround View Demo I don't see proper display, whereas video from sensor shows proper data.</p>
<p>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.</p>
<p>Q. Is there any script to automate loading and running of cores via CCS</p>
<p>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.</p>
<p>Q. How to ensure alignment when sharing data structures across cores</p>
<p>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.</p>

### 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.
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. A15 is a debug binary but still is see WFI instruction hit when we pause.
In case of Vision Sdk, A15 is build as debug binary; The PROFILE used to build is Release so PM will be included. The Build Profile needs to debug to remove CPU_IDLE from the build.
Q. Does Vision Sdk use GP timer in their configurations which are those?
Yes, Vision Sdk does use the timers. Timer 1,2 5,6 9,11 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/tda2ex	Rules specifically for TDA2Ex device
vision_sdk/docs	Documentation for vision SDK
vision_sdk/examples	Sample Use case / application folder
<b>vision_sdk/examples/tda2xx</b>	<b>Use case for TDA2xxx,TDA2Ex and TDA3xx device</b>
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_a15	Source files for individual links present on A15
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_ipu	Source files for individual links present on IPU
vision_sdk/src/main_app	Folder for main() functionality
vision_sdk/src/utlis_common	Common utilities used in framework
ti_components	Root directory of tools accompanying vision SDK
ti_components/algorithms_codecs/framework_components_x_xx_xx_xx	Framework Components
ti_components/algorithms_codecs/ivahd_hdvp20api_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/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_ild_xx_xx_xx_xx	Driver for system DMA usage
ti_components/gel/TDA2Ex	Gel files modified for Vision-Sdk
ti_components/networking	Networking related tools
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
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	14 <sup>th</sup> November 2014	Added QSPI+SD boot, GCC and CCSversion
3.0	31 <sup>st</sup> December 2014	Some minor changes
4.0	4 <sup>th</sup> July 2015	Updated for 2.7 release
5.0	16 <sup>th</sup> Oct 2015	Updated for 2.8 release

« « « § » » »