Vision SDK TDA2Px

(v03.01.00)

User Guide

IMPORTANT NOTICE

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 family of SoCs

1.1 References

Refer the below additional documents for more information about Vision SDK

Document	Description
VisionSDK_ReleaseNotes.pdf	Release specific information
VisionSDK_UserGuide.pdf	This document. Contains install, build, execution information
VisionSDK_DataSheet.pdf	Summary of features supported, not supported in a release. Performance and benchmark 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_SurroundView_DemoSetUpGuide.pdf	Document contains the steps for hardware setup for calibrated surround view demo
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 linaro 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 in "<install dir>/ti_components/cg_tools/windows/gcc-arm-none-eabi-4_9-2015q3" 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

2.1.3 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 its 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 linaro tools should be downloaded from below link

https://releases.linaro.org/components/toolchain/binaries/5.3-2016.02/arm-linux-gnueabihf/gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf.tar.xz

The tools need to be installed in $INSTALL_DIR/ti_components/os_tools/linux/linaro location.

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/os_tools/linux/linaro
```

```
$> tar –xvf gcc-linaro-5.3-2016.02-x86_64_arm-linux-gnueabihf.tar.xz
```

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. TDA2Px 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.

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

IMPORTANT NOTE: Only on-chip HDMI output is supported on TDA2Px EVM.

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

SRV use-case needs the below hardware

1. TDA2Px EVM, power supply (12V 5 AMP)
2. UB960/964 Application Board, power supply (12V 5 AMP)
3. IMI modules (OV10640 Rev E sensor) & LVDS cables to connect camera modules to UB960 application board
    a. Details of IMI camera module: http://www.global-imi.com/media/2015/10/Generic-Minicube-Catalogue-1020151.pdf
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.

IMPORTANT NOTE: Refer to the TDA3xx Surround View Demo userguide VisionSDK_UserGuide_3D_SurroundView_TDA3xx_Demo.pdf for the required input tables to be stored in SD card

2.3.3 Sensor Interface

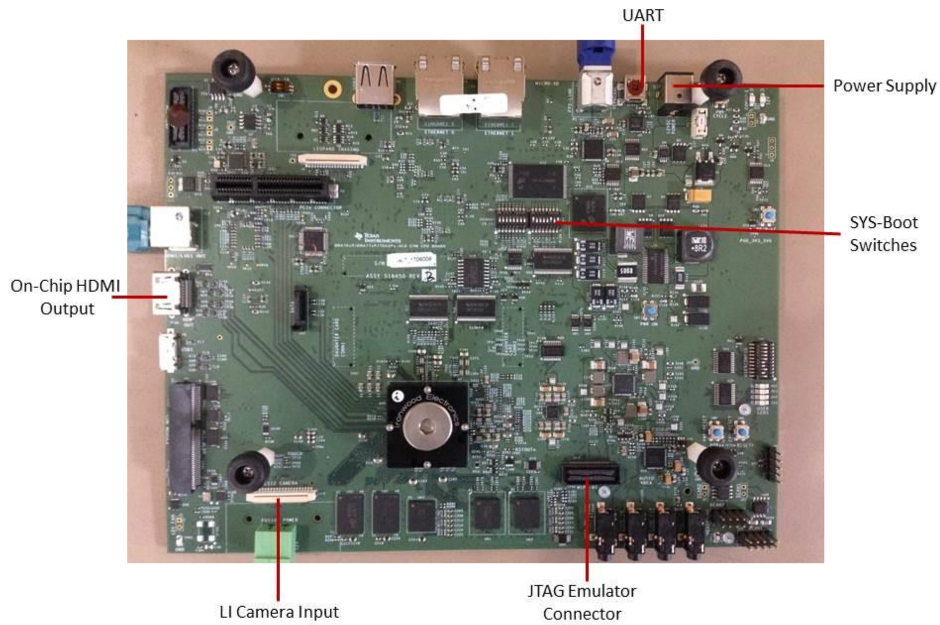


Figure 2.3.3.1 TDA2Px EVM Front Side

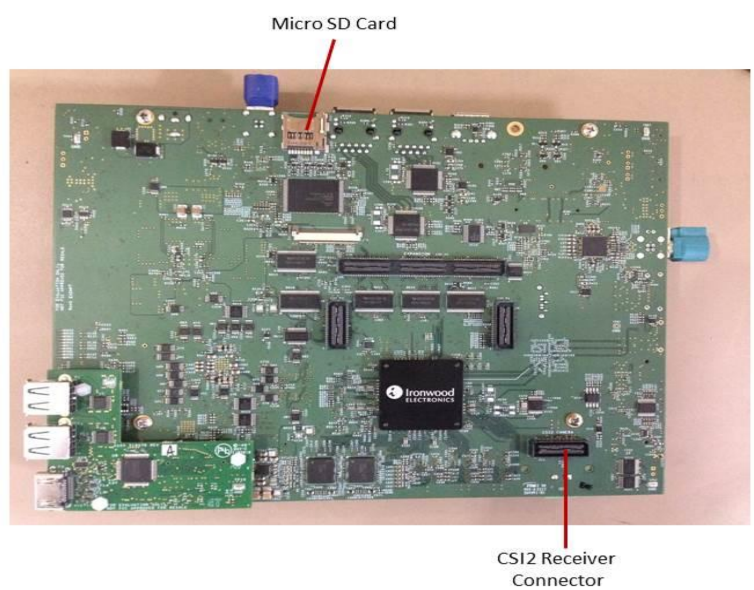


Figure 2.3.3.2 TDA2Px EVM Bottom Side

2.4 Required H/W modification / Configurations

2.4.1 Changes required on UB960 Application board

Refer to the VisionSDK_UserGuide_TDA3xx.pdf for the UB960 Application board changes.

2.5 Supported Sensors

The Table below lists the sensors supported on TDA3xx platform. Please refer section 2.3.3, this details the interfaces that a sensor could be connected to.

Sensor	Details	Comments
OV10640 MIPI REV E (CSI2)	OV10640 CSI2 Leopard imaging (LI) sensor on **LI Camera CSI2 Interface** (for ISS use-cases only)	It outputs RAW 12 in bayer format and ISP is used to convert to required format.
IMI (OV10640)	Connected to UB960 EVM, this module has OV10640 sensor and 913 Serializer	The ISP is used to convert from RAW12 to YUV420. Algorithms (AEWB) is tuned for this sensor
OV2775 CSI2	OV2775 CSI2 sensor on CSI2 receiver connector	It outputs RAW 12 in bayer format and ISP is used to convert to required format. Algorithms (AEWB) is not tuned for this sensor.

Refer to the section 2.5 in Vision_SDK_TDa3xx.pdf for the supported feature on each of the above 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 got 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

- ISS Use-cases
  - 1CH ISS capture + ISS ISP + ISS LDC+VTNF + Display

    Single Channel ISS Usecase
  - 3D + 2D SRV 4CH ISS capture + ISS ISP + DeWarp + Synthesis (DSP1) + Display

    Multi-Channel ISS 3D and 2D SRV usecase using LDC/DeWarp link and DSP
  - 3D SRV 4CH ISS capture + ISS ISP + DeWarp + Synthesis (DSP1) + Display

    Multi-Channel ISS 3D SRV Usecase using LDC/DeWarp link and DSP
  - Surround View Calibration

    ISS SRV calibration usecase

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

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

The demos support following devices as display devices

- HDMI 1080p60 (default)

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

3.2 Building the application

1. On windows command prompt, go inside the directory PROCESSOR_SDK_VISION_03_xx_xx_xx\vision_sdk\build.
2. Open file \vision_sdk\build\Rules.make and set required config **MAKECONFIG=tda2px_evm_bios_all**
3. 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

Ensure that gmake is picked from vision sdk xdc path only.

Use which gmake or where gmake depending upon the git bash or win cmd

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 build/tools_path.mk.

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 to build. Refer** https://support.microsoft.com/en-in/kb/830473 **for more details**.

Git version used for testing is 2.13

(Always point to xdc path gmake only)

4. Under vision_sdk directory
    a. When building first time run the below sequence of commands
       > gmake -s -j depend
       > gmake -s -j
    b. 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
- when some links are added or removed

If "gmake -s -j depend" 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.

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

```
\vision_sdk\binaries\$( MAKEAPPNAME)\$(MAKECONFIG)\vision_sdk\bin\tda2px-evm
```

```
vision_sdk_a15_0_release.xa15fg
vision_sdk_arp32_1_release.xearp32F
vision_sdk_arp32_2_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
```

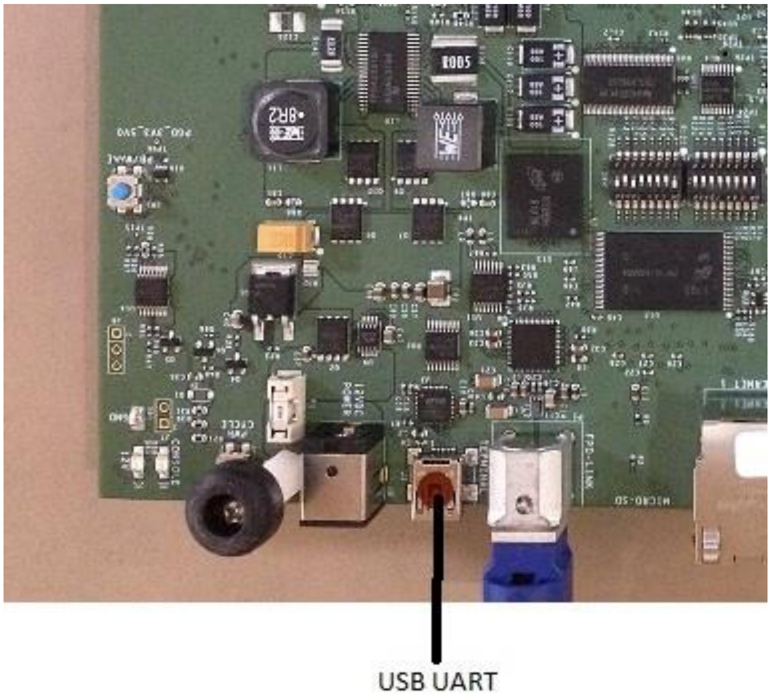
6. To speed up the incremental builds the following can be done as required. The number of processors included in the build can be changed by modifying below values in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk. A value of "no" means CPU not included in build, value of "yes" means CPU included in build. Make sure to do "**gmake -s -j depend**" before "**gmake -s- j**" when number of CPUs included is changed

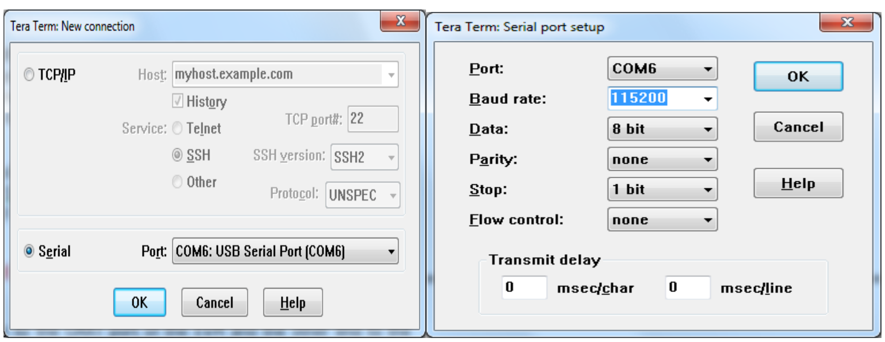
```
PROC_DSP1_INCLUDE=yes
PROC_DSP2_INCLUDE=yes
PROC_EVE1_INCLUDE=yes
PROC_EVE2_INCLUDE=yes
PROC_A15_0_INCLUDE=yes
PROC_IPU1_0_INCLUDE=yes
PROC_IPU1_1_INCLUDE=yes
PROC_IPU2_INCLUDE=yes
```

7. The build config that is selected in config file can be confirmed by doing below
   `> gmake -s showconfig`
   Cleaning the build can be done by following command
   `> gmake -s clean`
   Built binaries need to deleted by
   `> rm -rf ..\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)`

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.





3.4 Boot Modes

Supported boot modes on TDA2px ES1.0 device:

Boot Mode	EVM Switch Setting SYSBOOT(SW3)[1:16]	EVM Switch Setting USER Config
QSPI_1	01101100 10000001	0001100000
QSPI_4	11101100 10000001	0001100000
SD	00001100 10000001	0001100000
Debug	00000000 10000001	XXXXXXXX

3.5 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.

3.5.1 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.

3.5.2 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

3.5.3 Steps to generate MLO

NOTE: SBL MLO image is built from PDK package.
To build MLO Run the command **gmake -s sbl** from vision_sdk\build dir
This generates an MLO under
vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\sd

To build the mlo for different memory map, select required configuration in Makefile under vision_sdk (follow comments from Makefile under SBL build Targets).

3.5.4 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 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 file

3.5.5 SD Card setup

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

3.5.6 Hardware Pin settings for SD Boot

Make sure the Boot Mode Select Switch is set for the SD boot mode **on TDA2px Base EVM**. This is done by setting the pins SYSBOOT (SW2+SW3)

- Please refer **Boot Modes**

3.6 Load using QSPI

3.6.1 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 dir
This generates all required tools under
vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi\$(OPP)\$(PLATFORM)
vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\QSPI_flash_writer\$(PLATFORM)

1. sbl_qspi_$(OPP_MODE)_a15_0_release.tiimage
2. qspi_flash_writer_ipu1_0_release.xem4

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: Refer section "***Board Modification***" under SBL_userguide for hardware modifications if required.

3.6.2 Steps to generate appImage

Following steps need to be followed to generate the application image

Make sure the executables are built as shown in Building the application

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 file

- **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.**

3.6.3 Flashing steps

Flashing pin settings:

- Please refer **Boot Modes**

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

1. Connect A15. Do CPU reset

   Select CortexA15_0, navigate to Scripts->TDA2Px MULTICORE Initialization TDA2Px_MULTICORE_EnableALLCores
2. Connect M4 (IPU)

   Halt A15 core, and Load image on M4

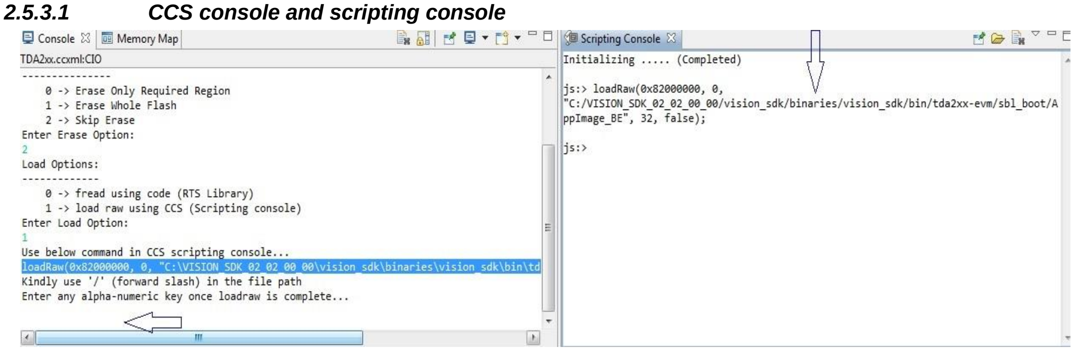
   vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\qspi_flash_writer\$(SOC)\qspi_flash_writer_ipu1_0_release.xem4
3. Run the core. You would see below console logs

[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 TDA2Px EVM, press '2'.** MID - 1 DID - 18 Enter 0 for Erase-Only (without flashing any image) Note : File size should be less than 33554432 Bytes. Enter the file path to flash: **<PATH>\sbl_qspi_$(OPP_MODE)_a15_0_release.tiimage** **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**************

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

[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 TDA2Px EVM, press '2'.*** *MID - 1* *DID - 18* *Enter the File Name* ***C:\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***	*Load Options:* *-------------* *0 -> fread using code (RTS Library)* *1 -> load raw using CCS (Scripting console)* *Enter Load Option:* ***1*** *Open Scripting console window by clicking "Menu -> View -> Scripting console" and enter below command on scripting console as shown 3.5.3.1* ***loadRaw(0x80500000,0, "C:/vision_sdk/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********

2.5.3.1 CCS console and scripting console



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

3.7 Load using CCS

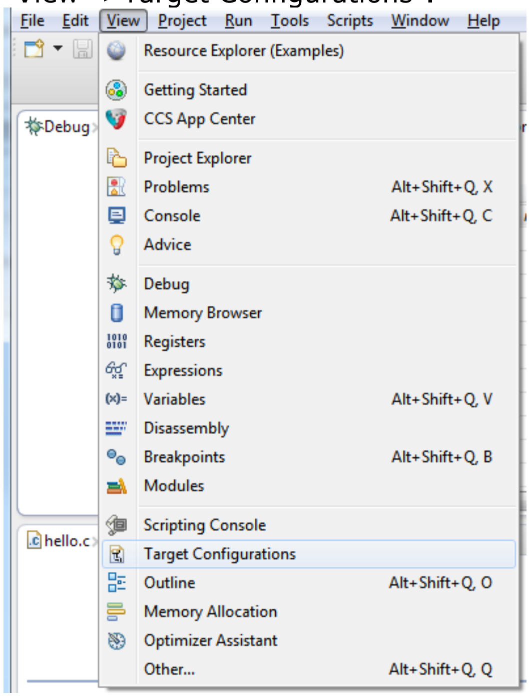
After installing CCS, follow below steps to complete the platform setup,

1. GELs are available in <Install_dir>\ti_components\ccs_csp\auto_device_support_x.x.x.zip
   NOTE:
   - Latest GELs are also be available at http://processors.wiki.ti.com/index.php/Device_support_files
     Under Automotive pick

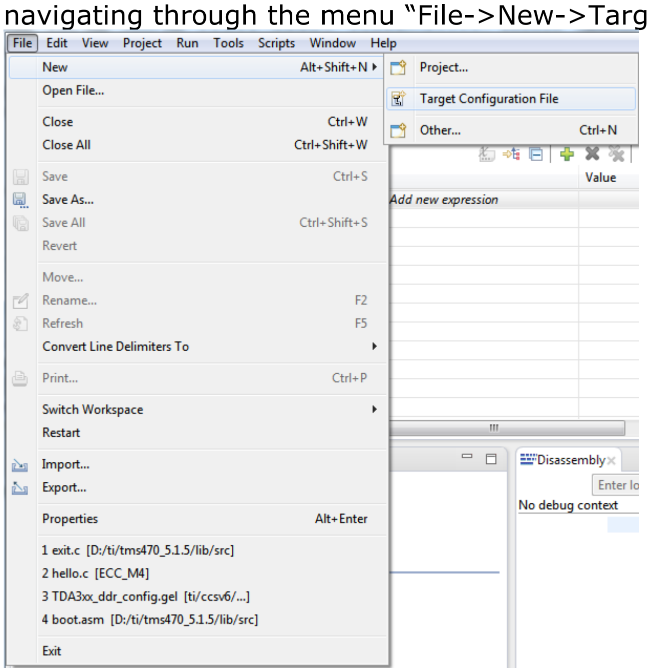
Automotive vX.X.X

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

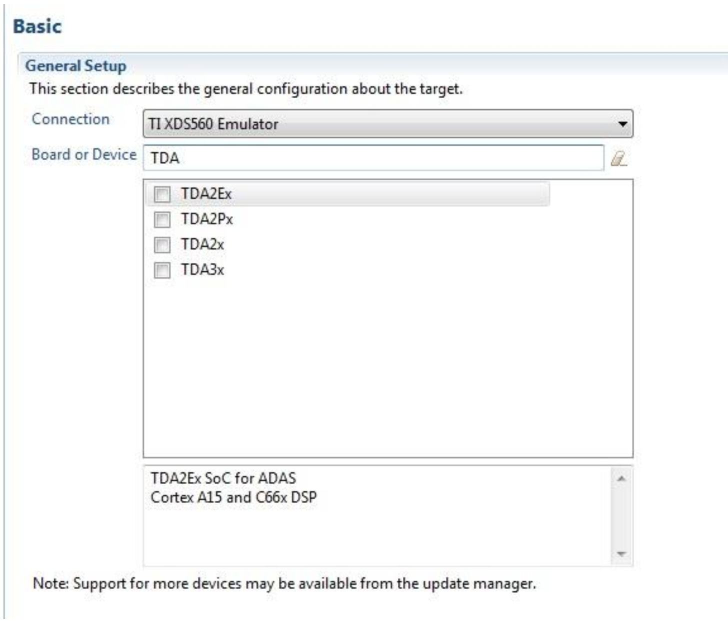
2. CCS Target Configuration creation:
    a. Open “Target Configurations” tab, by navigating through the menu “View ->Target Configurations”.



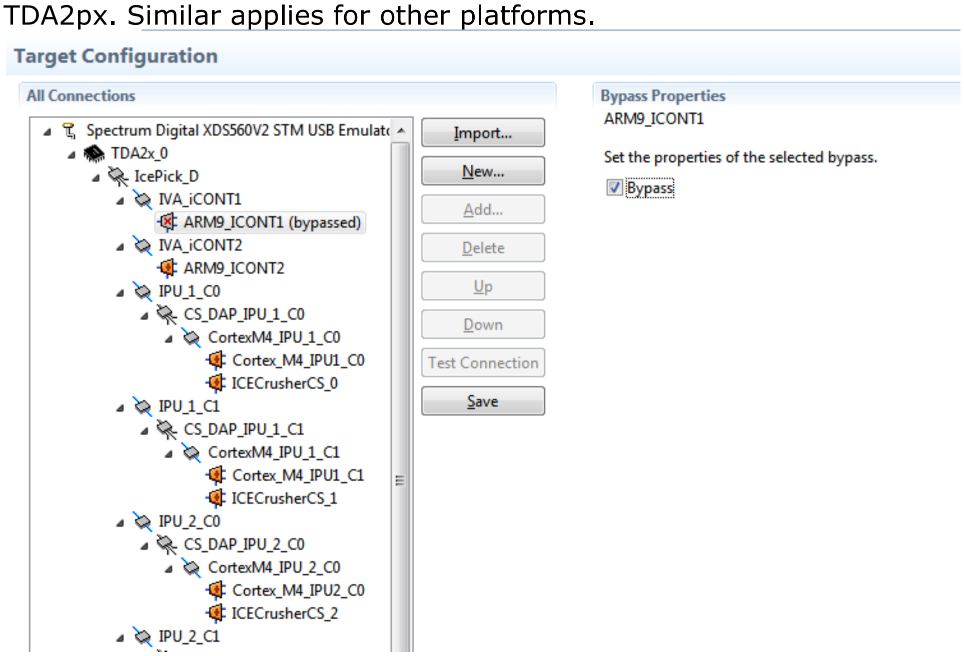
    b. Create a new Target Configuration (TDA2Px Target Configuration) by navigating through the menu “File->New->Target Configuration File”.



    c. Specify Connections as “Spectrum Digital XDS560V2 STM USB Emulator”. Specify Board or Device as “**TDA2Px**”. Then click on “Target Configuration link”

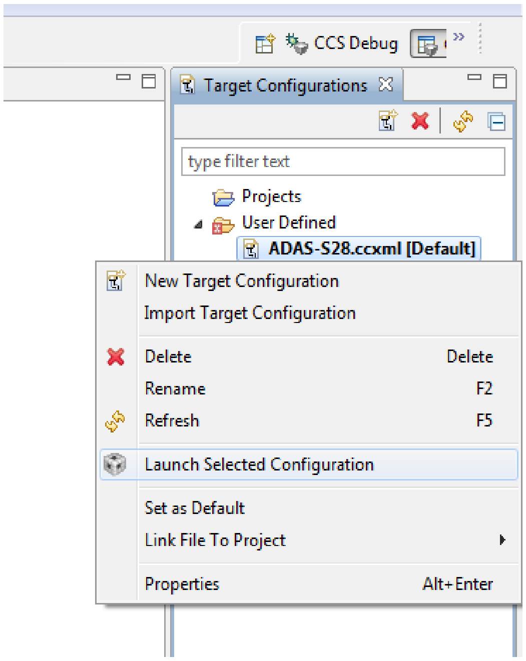


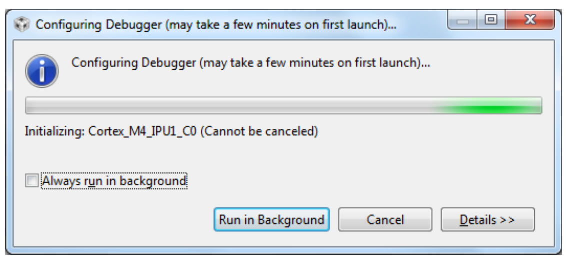
d. Bypass unused cores. Click on the core which needs to be bypassed and check Bypass under Bypass Properties.
The setting is under advance setup tab. Following image is example for TDA2px. Similar applies for other platforms.



3. Connect JTAG to the board.
JTAG emulation connection is shown in the section 2.3.3

4. Now launch the previously created TDA2Px Target Configuration.

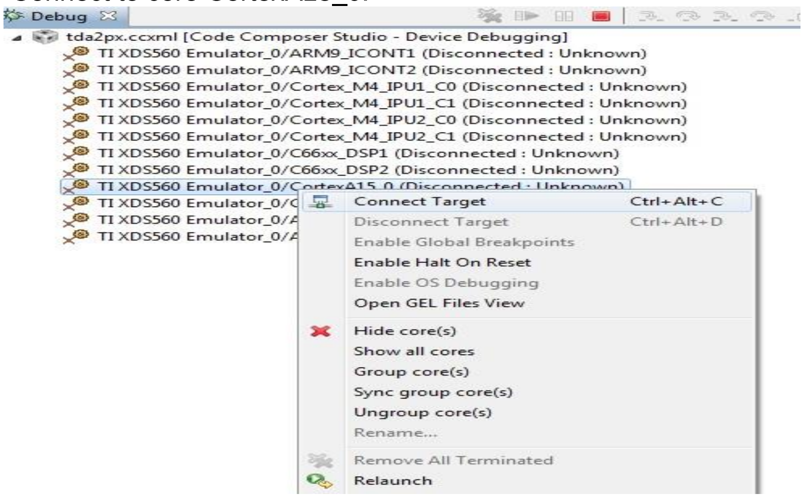




5. Once the target configuration is launched successfully, the following log should be observed on the CCS console:

```
CortexA15_0: GEL Output: --->>> TDA2Px Cortex A15 Startup Sequence DONE! <<<---
CortexA15_1: GEL Output: --->>> TDA2Px Cortex A15 Startup Sequence In Progress... <<<---
CortexA15_1: GEL Output: --->>> TDA2Px Cortex A15 Startup Sequence DONE! <<<---
ARP32_EVE_1: GEL Output: --->>> Configuring EVE Memory Map <<<---
ARP32_EVE_1: GEL Output: --->>> EVE Memory Map Done! <<<---
ARP32_EVE_2: GEL Output: --->>> Configuring EVE Memory Map <<<---
ARP32_EVE_2: GEL Output: --->>> EVE Memory Map Done! <<<---
```

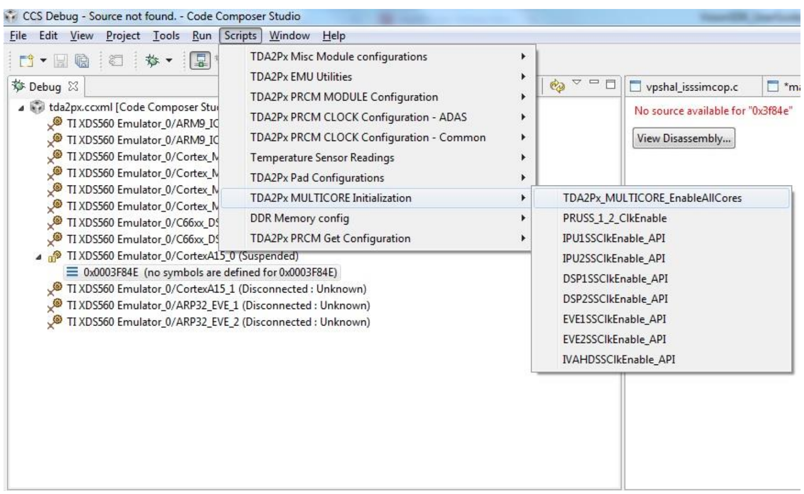
6. Connect to core CortexA15_0.



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

CortexA15_0: GEL Output: --->>> TDA2Px Target Connect Sequence DONE !!!!! <<<---

8. Select CortexA15_0, navigate to Scripts->TDA2Px MULTICORE Initialization TDA2Px_MULTICORE_EnableAllCores

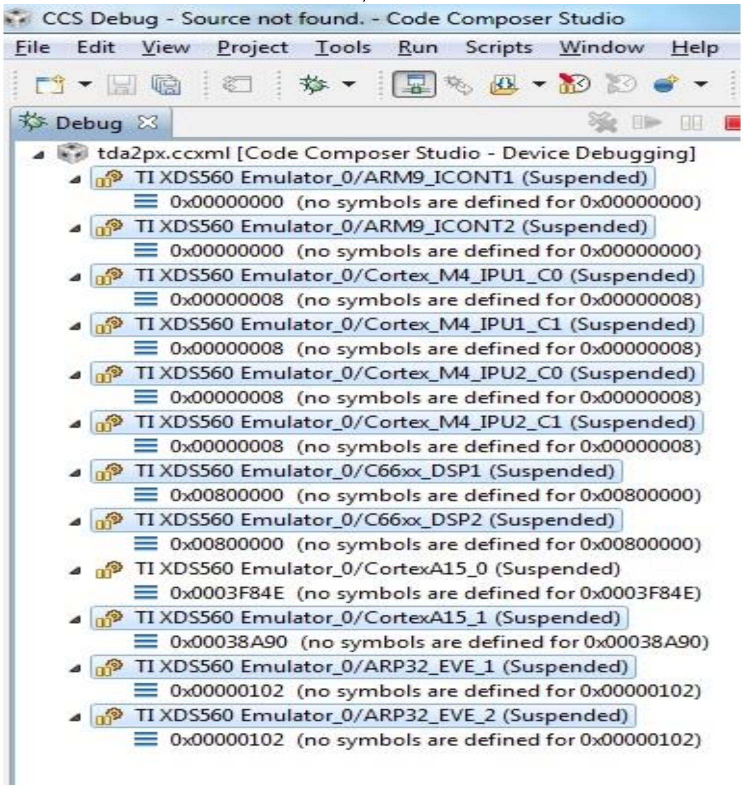


9. On successful script execution, the following log appears on CCS console:

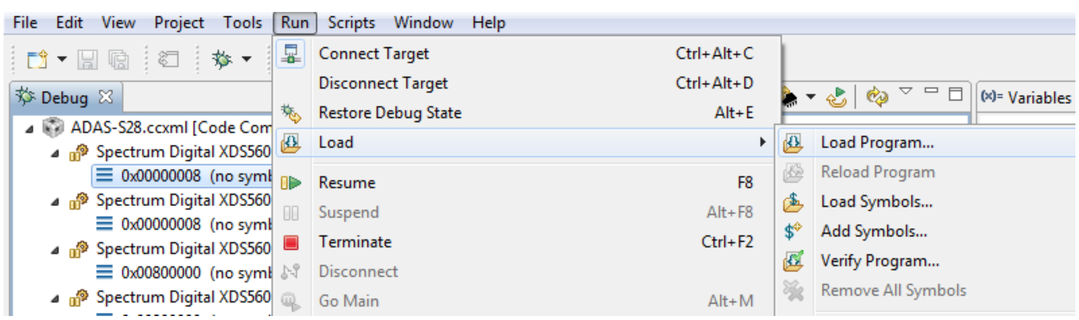
CortexA15_0: GEL Output: --->>> PRUSS 1 and 2 Initialization is in complete ... <<<---

10. Now connect the core shown below,
ARP32_EVE_1, ARP32_EVE_2, C66xx_DSP1, C66xx_DSP2, Cortex_M4_IPU1_C0, Cortex_M4_IPU1_C1.

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



12. On the cores load the binaries as mentioned below



On ARP32_EVE_2, load the binary, “vision_sdk_arp32_2_release.xearp32F”.
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”.
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

3.8.1 Steps to run

1. Power-on the Board after loading binaries by (SD, QSPI, 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.

4 Revision History

Version	Date	Revision History
1.0	13th October 2017	Initial Version
1.1	16th October 16, 2017	Addressed review comments

««« § »»»