Vision SDK TDA2xx

(v03.06.00)

User Guide

IMPORTANT NOTICE

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

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

Customers are responsible for their applications using TI components.

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

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

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

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

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

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

TABLE OF CONTENTS

1 Introduction

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

Vision SDK is currently targeted for the TDA2xx 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 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 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 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

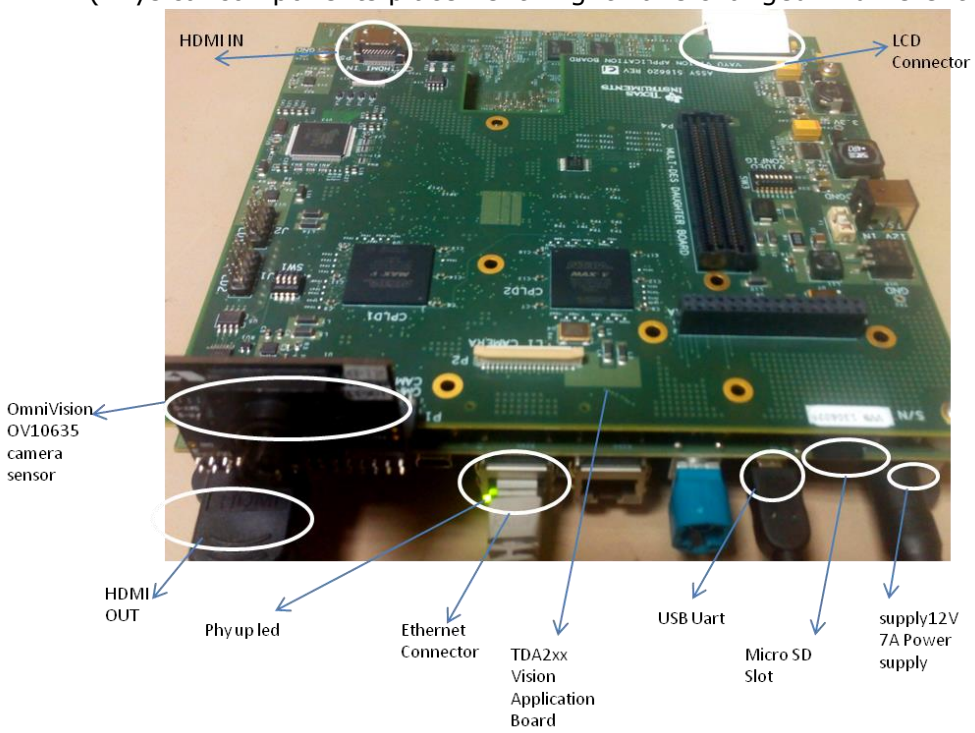
Hardware setup for Single Camera View (SCV), Multichannel AVB Multi-Channel View usecase and LVDS Multi Camera View (LVDS MCV) use-case is described in this section

2.3.1 SCV/AVB Use-case Hardware Setup

SCV/AVB use-case needs the below hardware

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

Setup is shown below
(Physical components placement might have changed in different board versions)



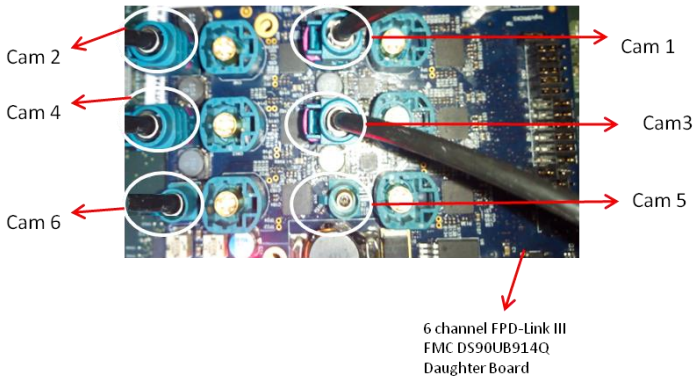
2.3.2 LVDS MCV Use-case Hardware Setup

LVDS MCV use-case needs the below hardware

1. TDA2xx 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:

2.3.2.1 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.*
5th Camera is used for Edge detection currently connected for Cam 6 to enable both AVB and LVDS usecase

IMPORTANT NOTE: *To enable 6th Camera or 6CH LVDS capture, networking MUST be disabled, since there is pinmux conflict between VIP port used for 6th camera capture and Ethernet port. To disable Ethernet and enable 6CH capture do below change,*

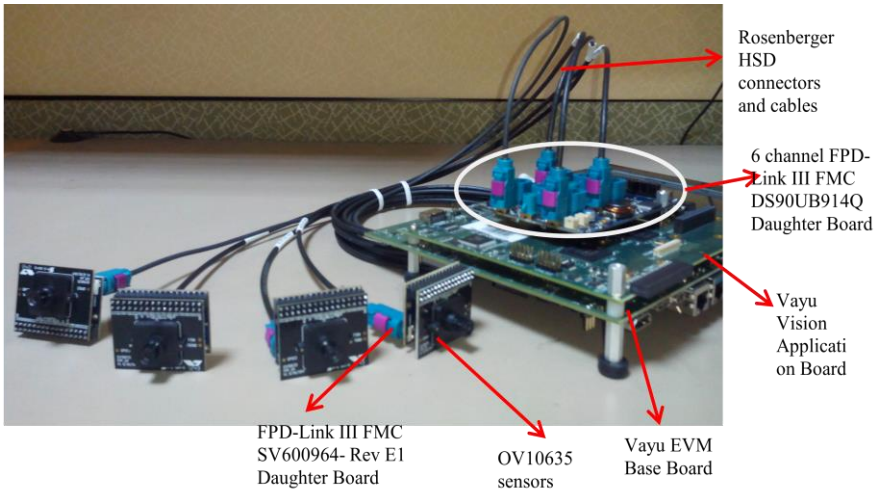
File: \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk
*NDK_PROC_TO_USE=**none***

File: \vision_sdk \$(MAKEAPPNAME)\src\rtos\video_sensor\include\video_sensor.h
*#define VIDEO_SENSOR_NUM_LVDS_CAMERAS (**6**)*

*Do, "gmake -s showconfig" to and check value of **NDK_PROC_TO_USE** to confirm this setting will get applied.*
Do "gmake -s -j depend_ndk_fatfs;gmake -s -j" to do an incremental build with modified settings

For SRV Demo setup Ref: VisionSDK_SurroundView_DemoSetUpGuide.pdf

2.3.2.2 Complete LVDS Setup



2.3.3 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.

2.3.4 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 source to recognize the format and resolution supported by the receiver (TDA2xx SoC). If this step is not done or if this step fails, then TDA2xx 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*.

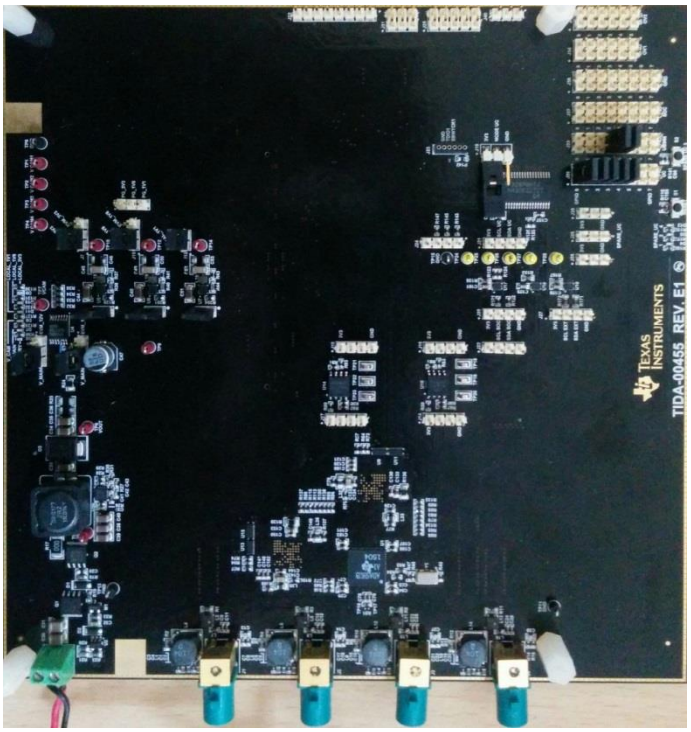
NOTE: Refer Load and Run using CCS for details of running the binaries

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.

2.3.5 Surround view use-case using TIDA00455/OV490

This use-case need following hardware

1. TDA2XX base EVM
2. TIDA00455 daughter card



Modifications needed:

a. SPI flashes on-board TIDA00455 must be configured with correct firmware
b. MSP430 on-board TIDA00455 must be configured with correct firmware
c. Modify R30 on TIDA00455 to ensure correct output on Power-on-coax network as required by camera modules

3. 4 cameras modules sending serialized video data in RAW (Bayer) format connected to TIDA00455 using FPD-Link-III coax-cables.

   DS90UB913EVM/SAT0074 with an appropriate camera module and adapter can be used. For testing, DS90UB813EVM with OV10640 camera's was used.

Modifications needed:

a. If using DS90UB913EVM for camera modules, modify R56 to appropriate value to ensure correct I/O voltage to camera

b. If using DS90UB913EVM for camera modules, modify MSP430 firmware such that GPIO1 is not set to "1" to limit current usage by GPIO1 LED.

Contact your local FAE to get access to appropriate firmware for MSP430 and OV490.

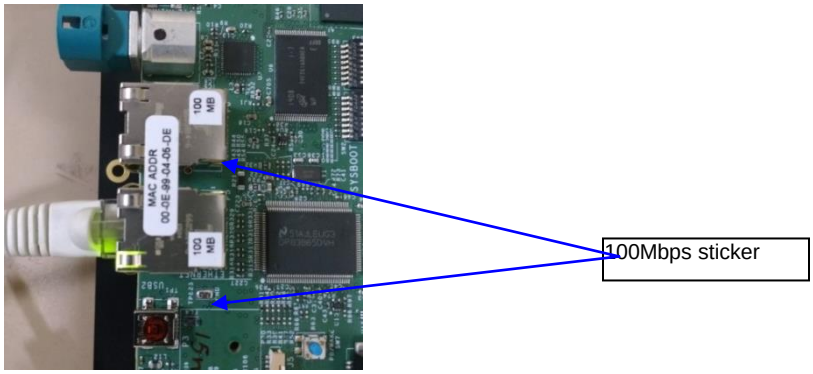
2.3.6 1Gig Ethernet link does not come up on some EVMs

On few of TDA2xx EVMs, we have seen stability issues with 1Gbps Ethernet link. The issue symptoms are

1. Link not detected when connected to 1Gbps switch/router
2. Stability issues with 1Gbps link - multiple connect/disconnect
3. Data transfer not happening due to CRC errors on Ethernet Rx

These are known timing issues due to PHY "wire" side connection and not with the TDAx processor. Spectrum Digital screens all the EVMS and places a sticker labelled 100Mbps if any of EVM fails such test. The EVM boards having TI DP83867 PHY do not have this issue.

Note: Not all failing EVMs have 100Mbps sticker, in order to identify if 1Gbps link is supported run VSDK network application.



2.3.7 INA226 Self power measurement setup

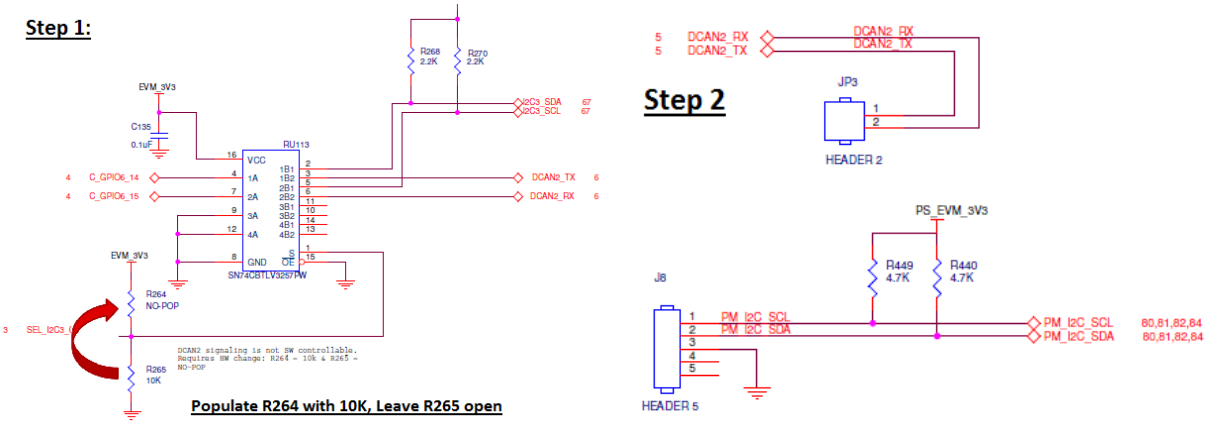
In order to measure power of the TDA2xx voltage rails from the TDA2xx device itself without the need for an extra daughter card the TDA2xx EVM should be modified as shown below. Without this board modification the power measurement software which runs on the device will fail.

STEP 1: Change the select for RU113 multiplexer on the board by making R264 = 10k & R265 = NO-POP. This allows DCAN2 Signaling.

STEP 2: Then perform a blue wiring for the following connections:

- DCAN2_TX (JP3 pin 1) to PM_I2C_SDA (J8 pin 2)
- DCAN2_RX (JP3 pin 2) to PM_I2C_SCL (J8 pin 1)

From Software Configure the PAD configuration registers such that the gpio6_14 and gpio6_15 pads operate as I2C3_SDA and I2C3_SCL respectively. Note that this is taken care from software. Additionally ensure SEL_I2C3_CAN2 is high.



2.4 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.

2.4.1 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.

3.1 Overview of application in release

The Vision SDK supports the following use-cases as examples

- Single channel capture use-cases
  - All single camera usecase.
- Multi-channel LVDS capture use-cases
  - All LVDS camera usecase.
- AVB RX Use-cases,
  - All AVB usecase.
- Dual Display Use cases
  - All Dual display usecase.
- ISS Use cases
  - All ISS capture usecase
- TDA2x Stereo Use cases
  - All Stereo usecase running on TDA2xx
- Network RX/TX Use cases
  - All Network test usecase

Refer to VisionSDK_DataSheet.pdf for detailed description of use-case.

The demos support following devices as capture source

- OV10635 sensor (default)
- HDMI source

The demos support following devices as display devices

- LCD 800x480
- HDMI 1080p60 (default)

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

3.2 AVB Protocol Stack Support in VSDK

Audio Video Bridging(AVB) is a collection of standards that provide "time synchronized low latency streaming services through IEEE 802 networks. TI AVB SW is shim layer of TI RTOS (SYS/BIOS) based stack on the top of NSP driver to showcase the high efficiency and throughput possible on a Cortex M4 cores of TDAxx SoCs.

TI AVB stack support is shown in below table.

Sr. No.	Protocol	Support in TI AVB stack
1.	IEEE 1722 "Layer 2 Transport Protocol	Yes - Support to extract media data from an Ethernet stream conformant to the IEEE 1722 standard. TI AVB solution supports 1722 with MJPEG and H264 video packets extraction (talker and listener). Audio is not supported. Priority tagging & QoS is supported in NSP driver. TI AVB solution uses 802.1Q tagged packets for receiving video.
2.	802.1AS Timing and Synchronization for Time-Sensitive Applications (gPTP)	No gPTP is not supported in the AVB stack. The Ethernet driver (NSP) supports hardware timestamping; this needs to be used by customer PTP stack.
3.	802.1Qat: Stream Reservation Protocol(SRP)	No
4.	802.1Qav: Forwarding and Queuing for Time-Sensitive Streams (FQTSS).	**Partial** - Traffic shaping using credit based scheduling

3.3 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=tda2xx_evm_bios_all**
3. Open file \vision_sdk\$(MAKEAPPNAME)\configs\tda2xx_evm_bios_all\cfg.mk
    a. For Building AVB application
       NDK_PROC_TO_USE is to be set for ipu1_1
4. 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)

5. Under vision_sdk directory
    a. 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.

    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.

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

```
\vision_sdk\binaries\$( MAKEAPPNAME)\$(MAKECONFIG)\vision_sdk\bin\tda2xx-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
```

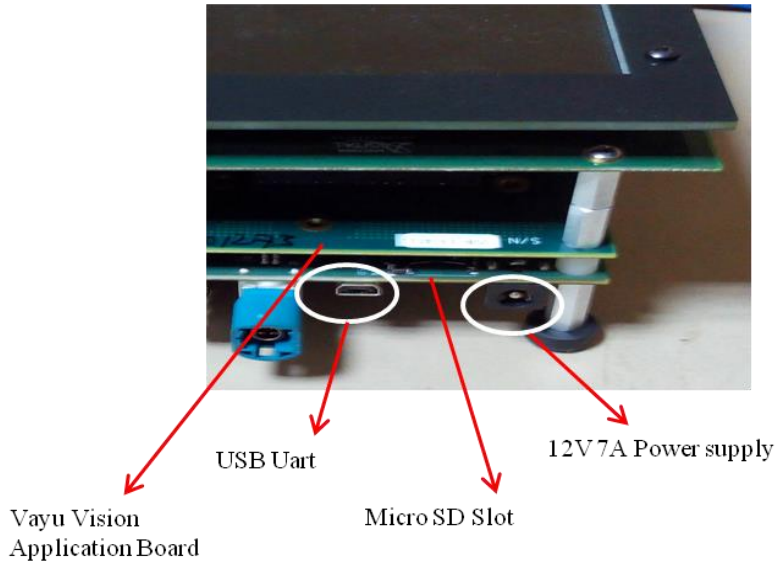
7. To speed up the incremental builds the following can be done as required. The number of processors included in the build can be changed by modifying below values in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk. A value of "no" means CPU not included in build, value of "yes" means CPU included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

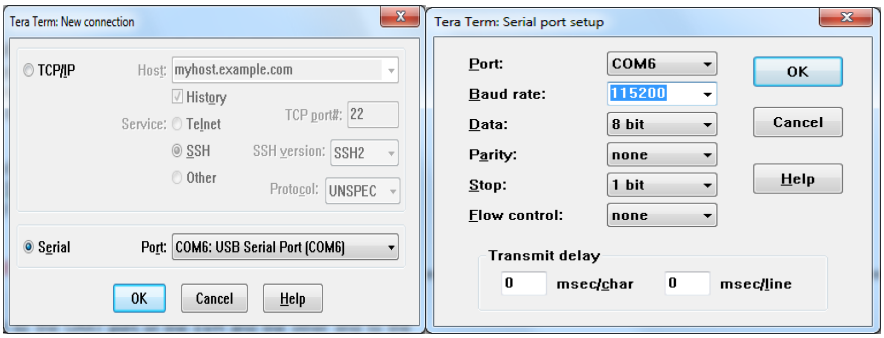
```
PROC_DSP1_INCLUDE=yes
PROC_DSP2_INCLUDE=yes
PROC_EVE1_INCLUDE=yes
PROC_EVE2_INCLUDE=yes
PROC_EVE3_INCLUDE=yes
PROC_EVE4_INCLUDE=yes
PROC_A15_0_INCLUDE=yes
PROC_IPU1_0_INCLUDE=yes
PROC_IPU1_1_INCLUDE=yes
PROC_IPU2_INCLUDE=yes
```

8. 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)`

3.4 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.





3.5 Boot Modes

Supported boot modes on TDA2xx ES1.1 device:

Boot Mode	EVM Switch Setting SYSBOOT(SW2)[1:16]	EVM Switch Setting SW5[1:10]
QSPI_1	01101100 10000001	1110100000
QSPI_4	11101100 10000001	1110100000
NOR	10101100 10000101	0100100000
SD	00001100 10000001	0001100000
Debug/CCS	00000000 10000001	XXXXXXXX

Supported boot modes on TDA2xx ES1.0 device:

Boot Mode	EVM Switch Setting SYSBOOT(SW2)[1:16]	EVM Switch Setting SW5[1:10]
QSPI	01101100 10000001	1110100000
NOR	10101100 10000101	0100100000
SD	11100000 10000001	0001100000
Debug/CCS	00000000 10000001	XXXXXXXX

3.6 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.

3.6.1 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.

3.6.2 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

3.6.3 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).

3.6.4 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

3.6.5 SD Card setup

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

3.6.6 Hardware Pin settings for SD Boot

Make sure the Boot Mode Select Switch is set for the SD boot mode **on TDA2xx Base EVM**. This is done by setting the pins SYSBOOT (SW2+SW3)

- Please refer **Boot Modes**

3.7 Load using QSPI

3.7.1 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.

3.7.2 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.**

3.7.3 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->DRA7xx MULTICORE Initialization DRA7xx_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 TDA2x 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
```

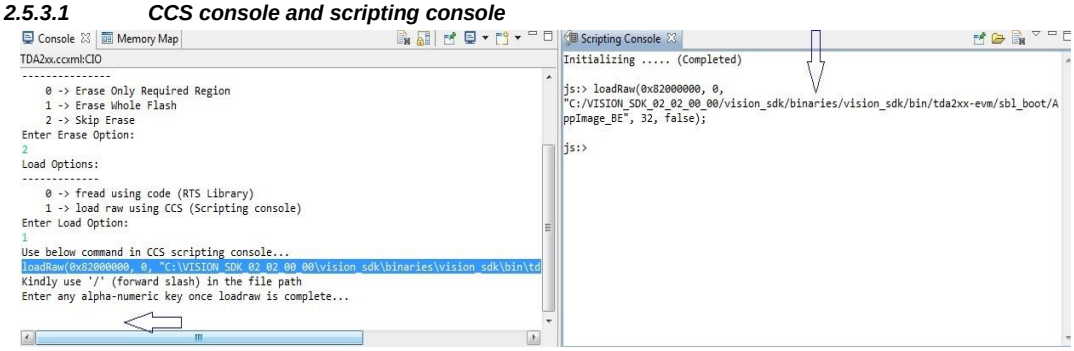
```
Erase Options:
---------------
        0 -> Erase Only Required Region
        1 -> Erase Whole Flash
        2 -> Skip Erase
Enter Erase Option:   1

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 TDA2x 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.

3.8 Load using NOR

3.8.1 Steps to generate nor writer tools

NOTE: SBL nor image is built from PDK package.
To build nor Run the command **gmake -s sbl** from vision_sdk\build dir
This generates all required tools under
vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\nor\$(OPP)\$(PLATFORM)
vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\nor_flash_writer\$(PLATFORM)

1. sbl_nor_$(OPP_MODE)_a15_0_release.bin
2. nor_flash_writer_ipu1_0_release.xem4

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

3.8.2 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

3.8.3 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

   Load image

   vision_sdk\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)\sbl\nor_flash_writer\$(SOC)\nor_flash_writer_ipu1_0_release.xem4

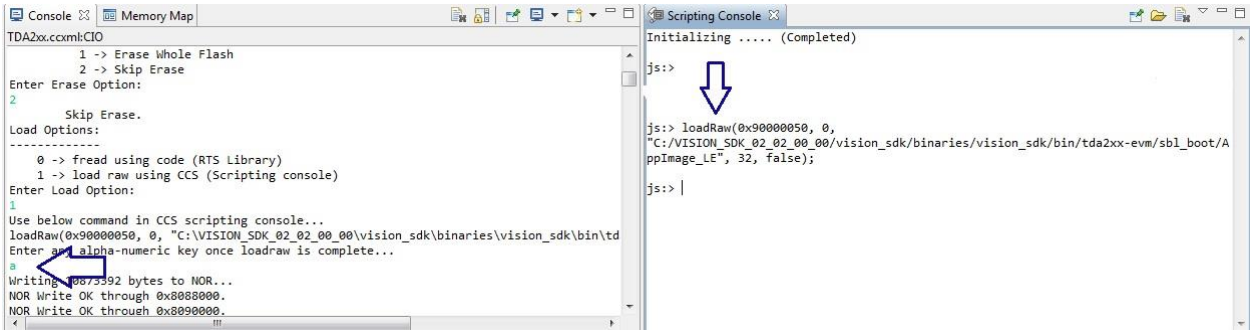
2. Run the core.

[CortexA15_0] Starting NOR Flash Writer. *CFI Query...passed.* *NOR Initialization:* *Command Set: Spansion* *Manufacturer: SPANSION* *Size: 0x40 MB* *Enter the File Name* **<PATH>\sbl_nor_ $(OPP_MODE)_a15_0_release.bin** *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*	Load Options: --- 0 -> fread using code (RTS Library) 1 -> load raw using CCS (Scripting console) Enter Load Option: 0 Reading 71896 bytes from file... Read 16384 bytes [22%] from file... Read 32768 bytes [45%] from file... Read 49152 bytes [68%] from file... Read 65536 bytes [91%] from file... Read 71896 bytes [100%] from file. Done!! Writing 0x118D8bytes to NOR... NOR Write OK through 0x8008000. NOR Write OK through 0x8010000. NOR Write OK through 0x80118D8. Done. !!! Successfully Flashed !!! NOR boot preparation was successful!

3. Reset the board and Repeat step 1 & 2
4. 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* **<PATH>\sbl_nor_ $(OPP_MODE)_a15_0_release.bin** *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 Use below command in CCS scripting console... loadRaw(0x90000050,0,"<PATH>/AppImage_LE", 32, false); Kindly use '/' (forward slash) in the file path Enter any alpha-numeric key once loadraw is complete... s1 Writing 0x1175F10bytes to NOR... NOR Write OK through 0x8088000. . . NOR Write OK through 0x91F5F10. Done. !!! Successfully Flashed !!! NOR boot preparation was successful!

4.6.3.1 CCS console and scripting console



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

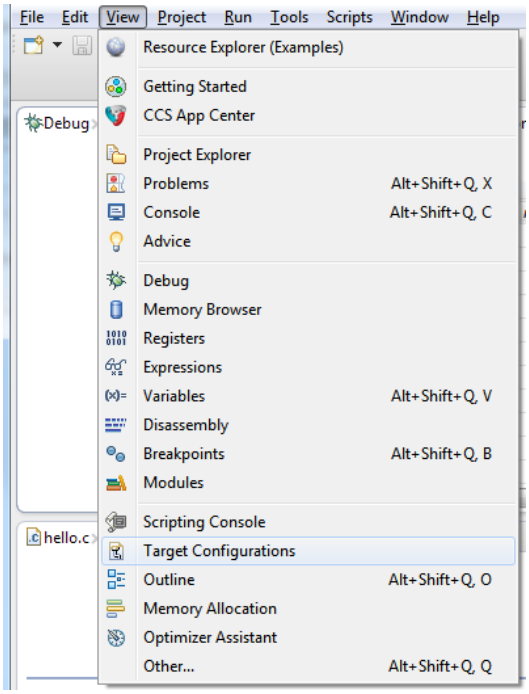
3.9 Load using CCS

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

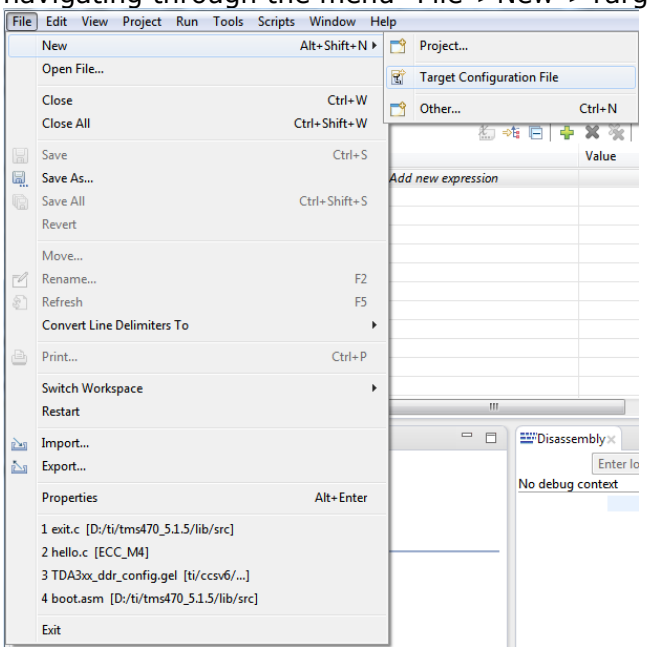
1. GELs are available in
   <Install_dir>\ti_components\ccs_csp\auto_device_support_x.x.x.zip
   NOTE:
   - Latest GELs are also be available at
     http://processors.wiki.ti.com/index.php/Device_support_files
     Under Automotive pick
     Automotive vX.X.X
   - To install the new GEL versions, you need to extract the zip to
     <CCS_INSTALL_DIR>/ccsv6/ccs_base

   Change the following GEL files for vision SDK as below,
   - TDA2xx_multicore_reset.gel
     - Set VISION_SDK_CONFIG to 1
     - Set VISION_SDK_CONFIG_OLD to 0
       - For older versions of VisionSDK (2.08 and older) this should be set to 1
     - Set EVE_SW_CONFIG to 0

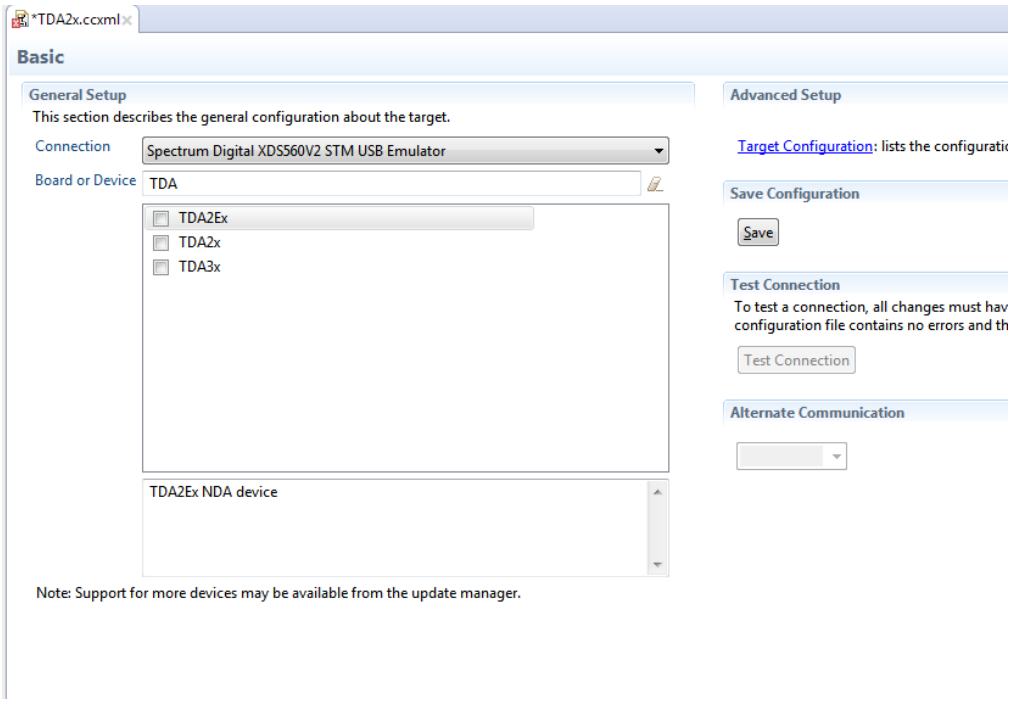
2. CCS Target Configuration creation:
   a. Open "Target Configurations" tab, by navigating through the menu "View ->Target Configurations".



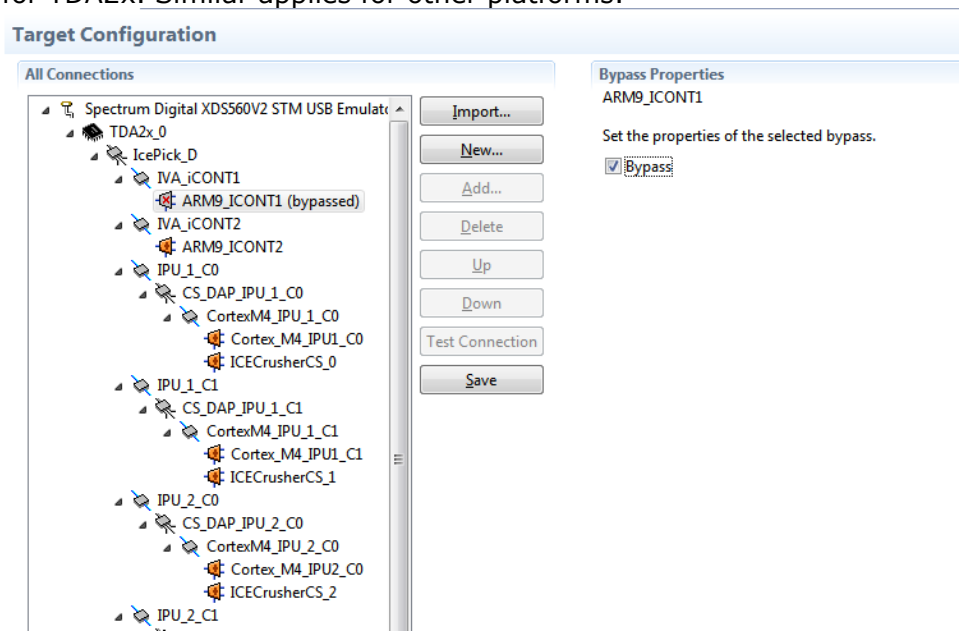
b. Create a new Target Configuration (TDA2xx Target Configuration) by navigating through the menu "File->New->Target Configuration File".



c. Specify Connections as "Spectrum Digital XDS560V2 STM USB Emulator". Specify Board or Device as "**TDA2x**". Then click on "Target Configuration link"

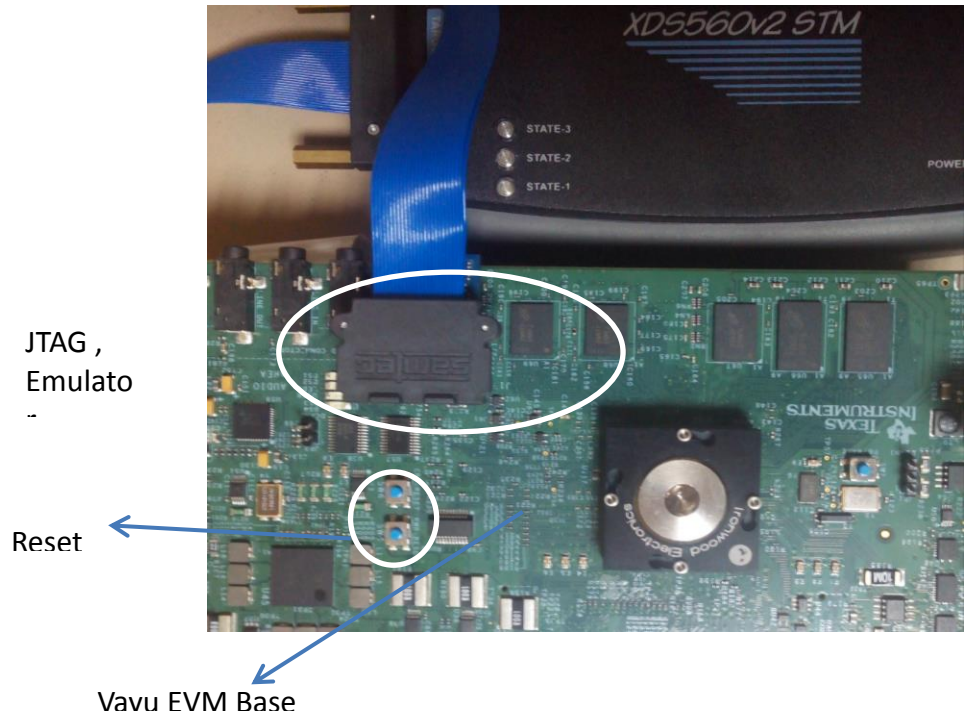


d. ByPass unused cores. Click on the core which needs to be bypassed and check ByPass under Bypass Properties.
The settings is under advance setup tab. Following image is example for TDA2x. Similar applies for other platforms.



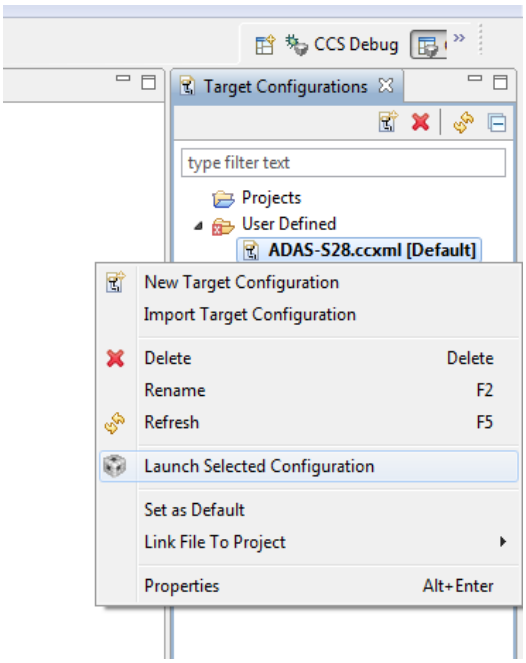
3. Connect JTAG to the board.

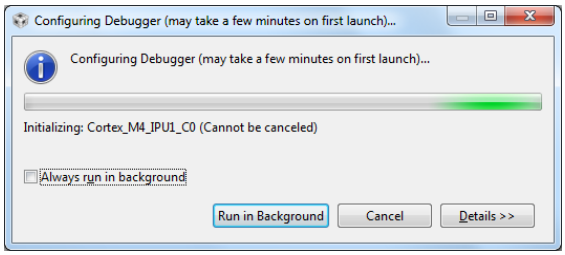
IMPORTANT NOTE: There are two JTAG connectors on the board. The one shown below MUST be used for CCS debug.



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

4. Now launch the previously created TDA2xx Target Configuration.

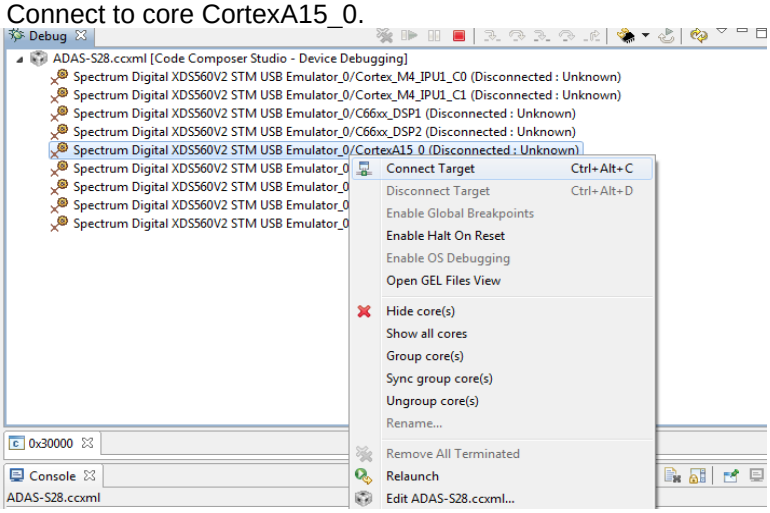




5. Once the target configuration is launched successfully, the following log should be observed on the CCS console:

CortexA15_1: GEL Output: --->>> DRA7xx Cortex A15 Startup Sequence DONE! <<<---

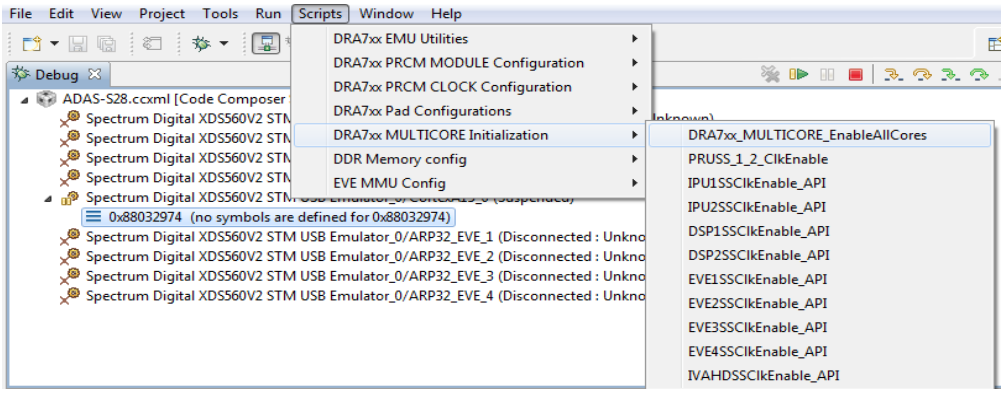
6. Connect to core CortexA15_0.



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

CortexA15_0: GEL Output: --->>> DRA7xx Target Connect Sequence DONE !!!!! <<<---

8. Select CortexA15_0, navigate to Scripts->DRA7xx MULTICORE Initialization DRA7xx_MULTICORE_EnableALLCores



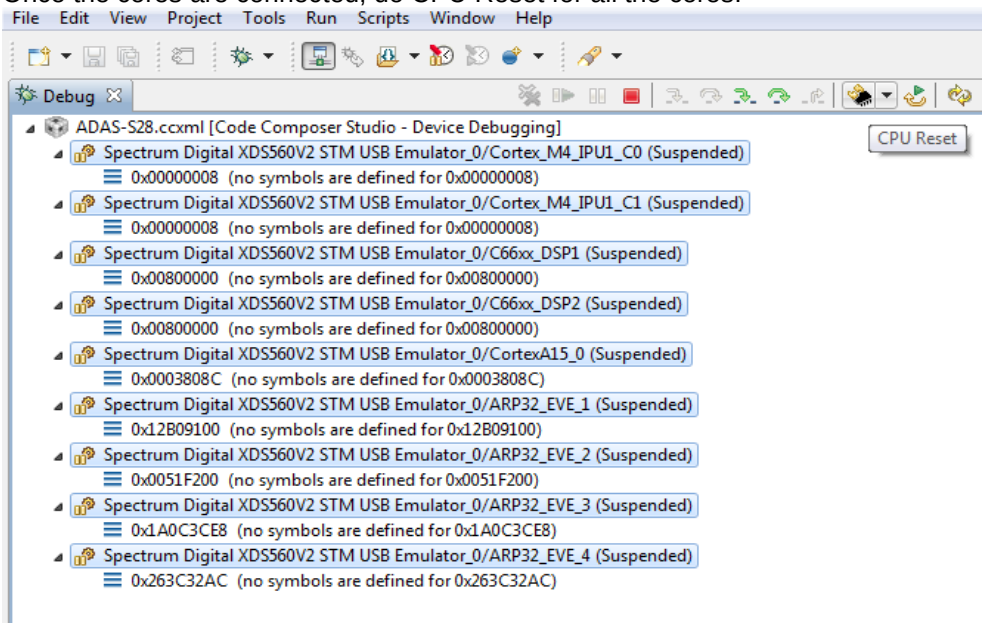
9. On successful script execution, the following log appears on CCS console:

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

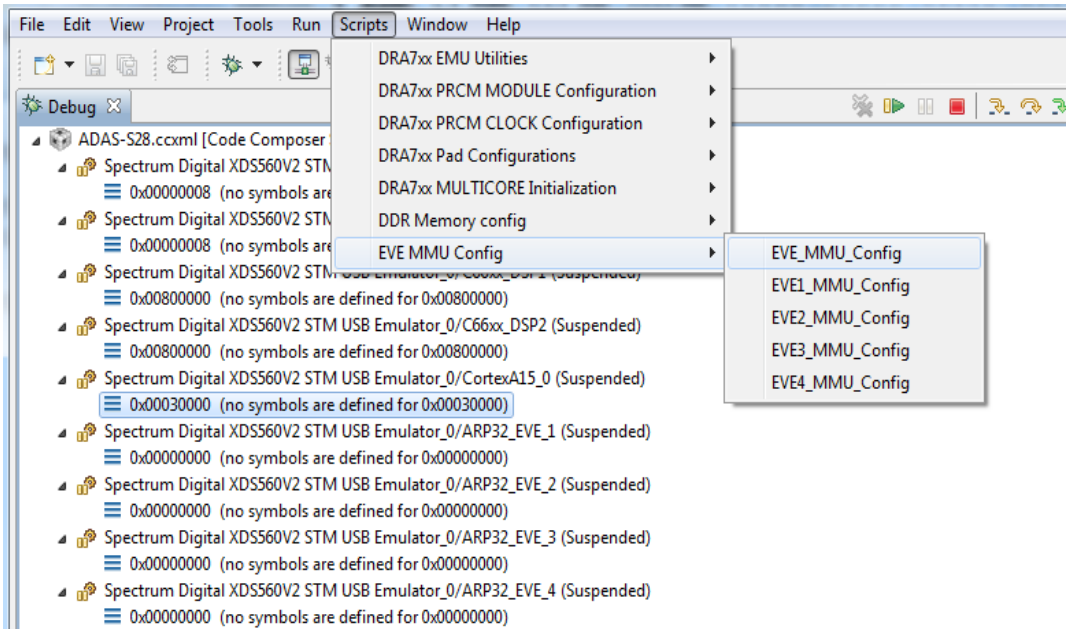
10. Now connect the core shown below,

ARP32_EVE_1, ARP32_EVE_2, ARP32_EVE_3, ARP32_EVE_4
C66xx_DSP1, C66xx_DSP2, Cortex_M4_IPU1_C0, Cortex_M4_IPU1_C1.

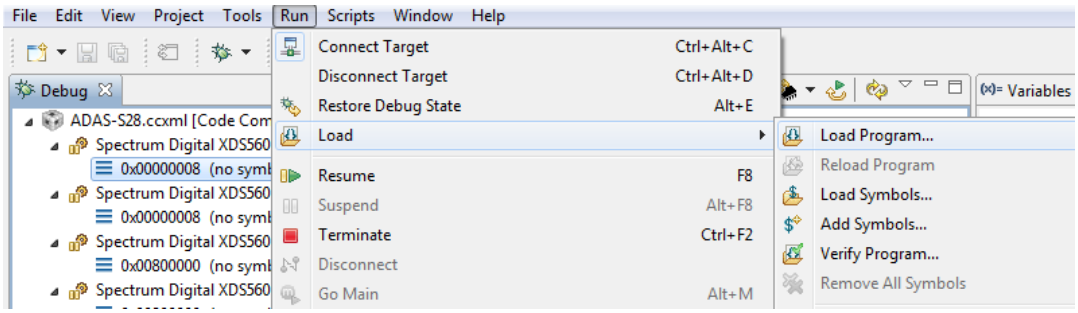
11. Once the cores are connected, do CPU Reset for all the cores.



12. For VisionSDK 2.6 and older, on CortexA15_0, run the GEL “Scripts -> EVE MMU Config -> EVE_MMU_Config”.
IMPORTANT NOTE: If this step is not done you will not be able to load executables on the EVE cores



13. On the cores load the binaries as mentioned below



On ARP32_EVE_4, load the binary, “vision_sdk_arp32_4_release.xearp32F”.
On ARP32_EVE_3, load the binary, “vision_sdk_arp32_3_release.xearp32F”.
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.

3.10 Run the demo

3.10.1 Single channel demos with HDMI input

IMPORTANT NOTE: To demonstrate better output all single channel usecases that require HDMI input should use video clips mentioned in the table below. These clips are part of PROCESSOR_SDK_VISION_03.XX.XX.XX_INPUTS.tar.gz

Usecase No. "Runtime Menu"	**Usecase**	**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 is part of VISION_SDK_02.XX.XX.XX_INPUTS.tar.gz

3.10.2 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 uart menu.

IMPORTANT NOTE: Make sure you select SCV (1Ch VIP capture) use-case or LVDS MCV (4CH LVDS VIP capture) use-case depending on the hardware you run the application.

IMPORTANT NOTE: For AVB MCV Demo Ethernet port must be connected as shown in SCV/AVB Use-case Hardware Setup

Data is streamed from Linux talker (Ref: AVB Used guide for building talker binaries)

VMware can also be used but the throughput of talker is not as desired and depends on PC configurations.

3.11 Autosar Demo

This application send message from IPU1_0 core running vision SDK to IPU2 core running non links and chain framework application and using MCAL/Autosar library for IPC communication.

3.11.1 Build the Application with IPC Lib

1. Open vision_sdk\apps\configs\tda2xx_evm_bios_all\cfg.mk and set

   AUTOSAR_APP = yes

   IPC_LIB_INCLUDE = yes

2. Under the vision_sdk/build directory
   a) Build all cores except IPU2 by disabling IPU2 core
   b) Build only IPU2 by disabling all cores except IPU2
   c) Build multicore appimage with IPU2 and all cores enabled in option a.

3.11.2 Build the Application with BIOS IPC

1. Open vision_sdk\apps\configs\tda2xx_evm_bios_all\cfg.mk and set

   AUTOSAR_APP = yes

2. Apply the IPC Lib patch.

   ```
   $> cd $INSTALL_DIR/vision_sdk/
   $> cp docs/Patches/IPClib_Autosar_with_Bios.patch ../ti_components/drivers/pdk/packages/ti/drv/ipc_lite/
   $> cd ../ti_components/drivers/pdk/packages/ti/drv/ipc_lite/
   $> git apply IPClib_Autosar_with_Bios.patch
   ```

3. Under the vision_sdk/build directory
   a) Build all cores except IPU2 and IPU1_1 by disabling IPU2 and IPU1_1 core
   b) Build only IPU2 by disabling all cores except IPU2
   c) Build multicore appimage with IPU2 and all cores enabled in option a.

Important Note: Always build depend when changing any code.

3.11.3 Run 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 option 1: Single Camera Usecases
3. Select option o: 1CH VIP capture + Display + Autosar
4. Press ‘m’ to send a message to Autosar and receive the acknowledgement.

4 Revision History

Version	Date	Revision History
1.0	21th August 2013	Initial Version
1.1	26th September 2013	Updated for release 1
1.2	4th March 2014	Updated for release 2
1.3	5th Oct 2015	Updated for release 2.8
1.4	1st July 2016	Updated for release 2.10
1.5	2nd Nov 2016	Updated for release 2.11
1.6	8th Feb 2017	Updated for release 2.12
1.7	19th June 2017	Updated linux installer
1.8	27th June	Updated for release 3.0
1.9	12th Oct 2017	Updated for release 3.1
2.0	20th Dec 2017	Updated for release 3.2
2.1	2nd April 2018	Updated for release 3.3
2.2	21st September 2018	Updated for release 3.5
2.3	21st December 2018	Updated for release 3.6

«««« § »»»»