Vision SDK on TDA3X RVP

(v03.05.00)

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 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, TDA3xx family of SoCs.

This document explains the HW/SW setup for TDA3xx Rugged Vision Platform (RVP).

The RVP developed by D3 Engineering is one of the evaluation platforms for Vision SDK. The RVP supports the following Vision SDK data flows:

- Surround View (SRV) Auto Calibration
- SRV 3D
- SRV 2D + 3D
- Fast Boot SRV 3D
- Network Tx/Rx (Limited support: Few command exchanges between PC network tool and RVP are tested)

1.1 References

Refer the below additional documents for more information about Vision SDK

Document	Description
VisionSDK_ReleaseNotes.pdf	Release specific information
VisionSDK_UserGuide_TDA3xx.pdf	This document. Contains install, build, execution information
VisionSDK_ApiGuide.CHM	User API interface details
VisionSDK_SW_Architecture.pdf	Overview of software architecture
VisionSDK_DevelopmentGuide.pdf	Developer Guide

1.2 Directory Structure

Once Vision SDK is installed; there will be two main directories installed - vision_sdk and ti_components.

Refer VisionSDK_UserGuide_TDA3xx.pdf for details of directory structure.

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>
```

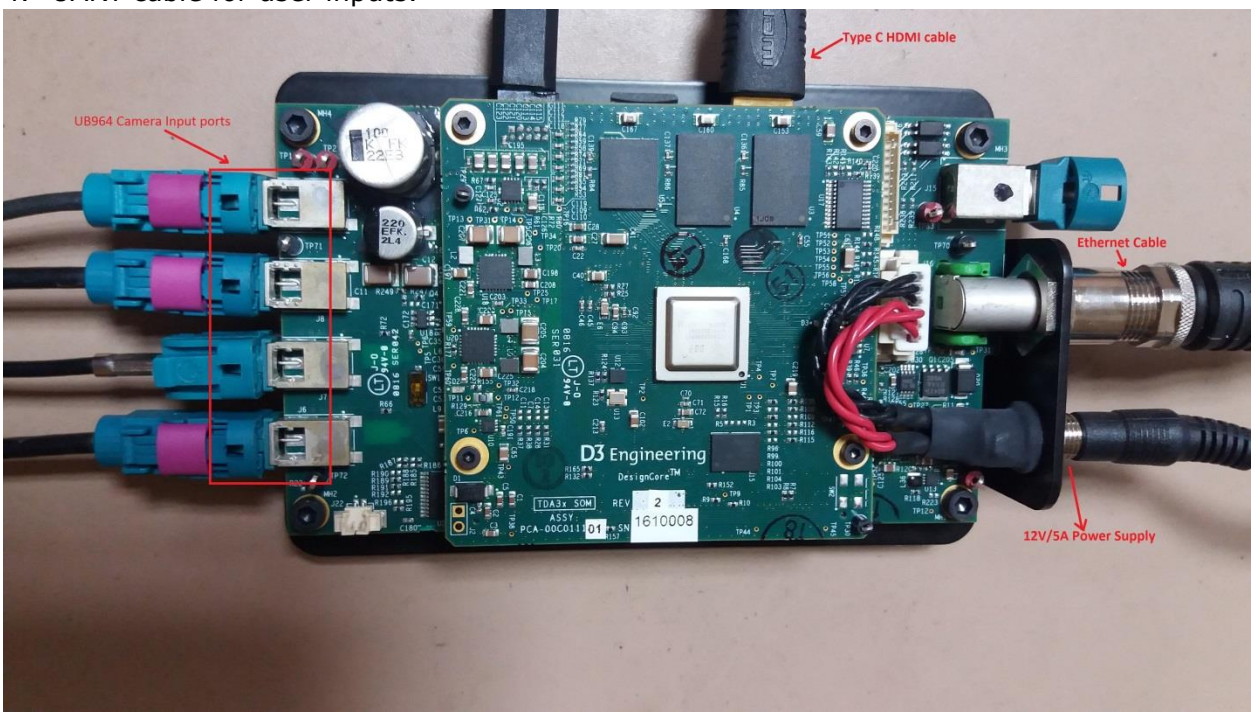
2.3 Hardware Requirements

Hardware setup required for RVP use-cases is described in this section

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

SRV use-cases needs the below hardware

1. TDA3xx RVP, power supply (12V 5 AMP)
2. Four TIDA00262 (AR0140 sensor) modules or IMI (OV10640 sensor) modules and LVDS cables to connect camera modules to UB964 input ports of RVP.
    a. Details of TIDA00262 camera module: http://www.ti.com/tool/TIDA-00262?keyMatch=TIDA-00262&tisearch=Search-EN-Everything#tiDevice
3. HDMI 1080p60 capable Display Monitor and **Type C HDMI cable**.
4. UART cable for user inputs.



2.4 Software Installation

PROCESSOR_SDK_VISION_xx_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 folders by name 'ti_components' and 'vision_sdk' would have been created in the installation path.

2.4.1 Uninstall Procedure

To uninstall, double click on uninstall.exe present in the installation directory.

3 Build and Run

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

3.1 Overview of RVP application in release

The Vision SDK on RVP supports the following use-cases

- SRV 3D
- SRV 2D + 3D
- Fast boot SRV 3D
- Network Tx/Rx (Limited support: Few command exchanges between PC network tool and RVP are tested)

Refer to VisionSDK_DataSheet.pdf for detailed description of the use-cases.

3.2 Building the application

1. On windows command prompt, go inside the directory \vision_sdk\build\.
2. Open file \vision_sdk\build\Rules.make and set

   MAKEAPPNAME=apps and **MAKECONFIG=tda3xx_rvp_bios_all**
3. Open file \vision_sdk\apps\configs\tda3xx_rvp_bios_all\cfg.mk
   a. Additionally to build for Fast Boot SRV 3D set SRV_FAST_BOOT_INCLUDE=yes

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

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

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

   IMPORTANT NOTE: If the installation folder depth is high then windows cmd prompt fails with error that it cannot find a file, even in file is present in mentioned path, this is because Windows has a limitation of 8191 characters for the commands that can execute. In such a situation as a workaround either restrict the folder depth to

d:/ or if it cannot be restricted use git bash to build. Refer https://support.microsoft.com/en-in/kb/830473 **for more details**.

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

5. Under \vision_sdk\build\ directory
   a. When building first time run the below sequence of commands
      > gmake -j depend
      > gmake –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 -j

Executing "**gmake -j depend** " will build all the necessary components (PDK (Platform Development Kit), EDMA drivers) and "**gmake -j"** will build the Vision SDK links framework and apps.

IMPORTANT NOTE: For incremental build, make sure to do "gmake -j depend" before "gmake -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\configs\$(MAKECONFIG) *_cfg.mk
- when any .h or .c file in TI component is installed in ti_components is changed
- when any new TI component is installed in ti_components

If "gmake -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\tda3xx-rvp
 vision_sdk_arp32_1_release.xearp32F
 vision_sdk_c66xdsp_1_release.xe66
 vision_sdk_c66xdsp_2_release.xe66
 vision_sdk_ipu1_0_release.xem4
 vision_sdk_ipu1_1_release.xem4
```

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 -j depend**" before "**gmake -j**" when number of CPUs included is changed

```
PROC_DSP1_INCLUDE=yes
PROC_DSP2_INCLUDE=yes
PROC_EVE1_INCLUDE=yes
PROC_IPU1_0_INCLUDE=yes
PROC_IPU1_1_INCLUDE=yes
```

8. The build config that is selected in config file can be confirmed by doing below

```
> gmake -s showconfig
```

9. Cleaning the build can be done by following command

```
> gmake clean
```

Alternatively, below folder can be deleted to delete all generated files

```
> rm -rf ..\binaries\$(MAKEAPPNAME)\$(MAKECONFIG)
```

3.3 UART settings

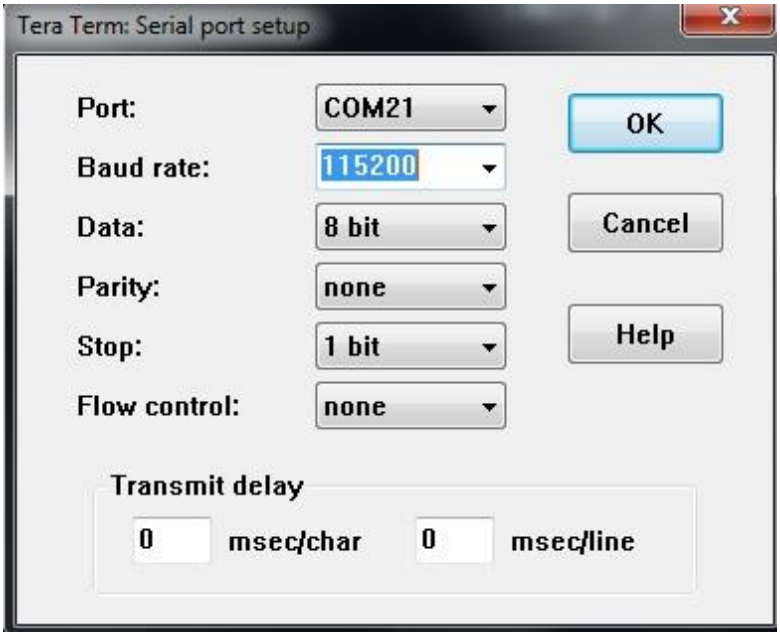
Connect a serial cable to the UART port of the RVP and the other end to the serial port of the PC (*configure the HyperTerminal at 115200 baud rate*) to obtain logs and select demo. RVP detects 4 UART ports, you need to select the 3rd one.

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

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

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


Tera Term: New connection
TCP/IP
Host: 172.24.190.65
History
Service: Telnet
SSH
Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPEC
Serial
Port: COM21: USB Serial Port (COM21)
OK
Cancel
Help


Tera Term: Serial port setup
Port: COM21
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
OK
Cancel
Help
Transmit delay
0 msec/char
0 msec/line

3.4 Load using QSPI and SD boot

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

The prebuilt QSPI flash writer (qspiFlashWriter_m4_release.xem4) is present in the 'vision_sdk\build\rtos\scripts\tda3xx-rvp\' folder.

3.4.1 Steps to generate sbl_qspi_sd

NOTE: SBL qspi_sd image is built from PDK boot package. To build qspi_sd Run the command gmake sbl from \vision_sdk\build\ directory. This generates 'sbl_qspi_sd_opp_nom_ipu1_0_release.tiimage' under \vision_sdk\binaries\apps\tda3xx_rvp_bios_all\sbl\qspi_sd\opp_nom\tda3xx-rvp\ directory.

IMPORTANT NOTE: For RVP default memory map is 1GB , Use the following build time macro to build SBL. These are changes for SBL any change in Memory map changes are to be done in corresponding apps\build\tda3xx\<>.xs Files

Change the macro in apps/configs/<MenuConfig>/cfg.mk file

Memory Map Configuration	VSDK Buid macro **DDR_MEM**	SBL Build macro to be passed **EMIFMODE**
1GB Memory Map	DDR_MEM_1024M	SINGLE_EMIF_1GB
512 MB Memory Map	DDR_MEM_512M	SINGLE_EMIF_512MB

More info Please look at SBL options in build\rtos\makerules\build_pdk.mk

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

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

3.4.2 Steps to generate appImage

Following steps need to be followed to generate the application image

1. Make sure the executables are built as shown in Building the application
2. To generate the application image run below command from "\vision_sdk\build\" folder
   > gmake appimage

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

3.4.3 Flashing steps

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

1. Connect M4 (IPU). Do CPU reset
2. Load image on M4

D:\vision_sdk\build\rtos\scripts\tda3xx-rvp\qspiFlashWriter_m4_release.xem4

3. Run the core.

   You should get below logs on console outputs

[Cortex_M4_IPU1_C0] *QSPI Flash writer application* *Enter Device type to use* *1 - Spansion 1 bit* *2 - Spansion 4 bit* *3 - Micron 1 bit* *4 - Micron 4 bit* *5 - Winbond 1 bit* *6 - Winbond 4 bit* *7 - ISSI 1 bit* *8 - ISSI 4 bit* *$ > 4* *MID - 20* *DID - ba* *Enter 0 for Erase-Only (without flashing any image)* *Note : File size should be less than 33554432 Bytes.* *Enter the file path to flash:* **D:\vision_sdk\binaries\apps\tda3xx_rvp_bios_all\sbl\qspi_sd\opp_nom\tda3xx-rvp\sbl_qspi_sd_opp_nom_ipu1_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:* **0**	*Load Options:* *----------* *0 -> fread using code (RTS Library)* *1 -> load raw using CCS (Scripting console)* *Enter Load Option:* **0** *Reading 76504 bytes from file...* *Read 16384 bytes [17%] from file...* *Read 32768 bytes [34%] from file...* *Read 49152 bytes [51%] from file...* *Read 65536 bytes [68%] from file...* *Read 76504 bytes [100%] from file...* *QSPI Erase started* *QSPI Erase completed* *QSPI file write started: file size 76504* **********QSPI flash completed sucessfully**********

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

3.5 Load using CCS

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

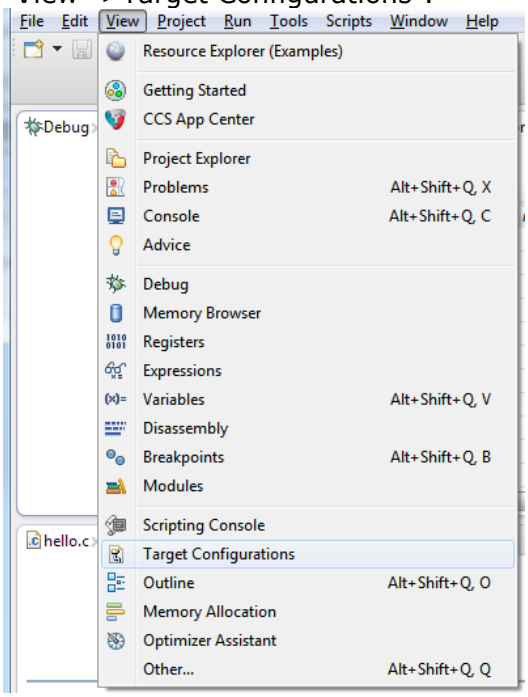
1. GELs are available in
   <Install_dir>\ti_components\ccs_csp \auto_device_support_x.x.x.zip
   NOTE:
   - GELs are also be available at
     http://processors.wiki.ti.com/index.php/Device_support_files
     Under Automotive pick
     Automotive vX.X.X
   - To install the new GEL versions, you need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

   Change the following GEL files for vision SDK as below,
   - TDA3xx_multicore_reset.gel

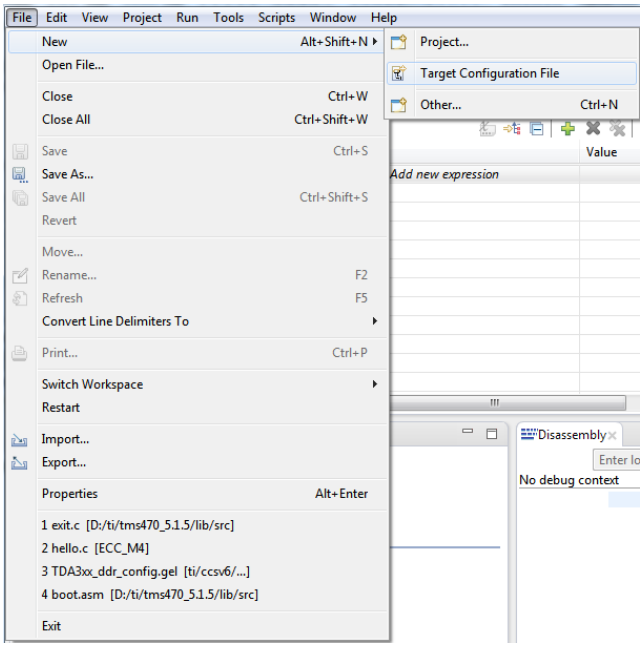
- Set VISION_SDK_CONFIG to 1
- 256MB mode not supported

For RVP, copy the 'TDA3xx_ddr_config.gel' present in the '/vision_sdk/build/rtos/scripts/tda3xx-rvp/' folder to '<CCS_INSTALL_DIR>/ccsv6/ccs_base/emulation/gel/TDA3x/' folder.

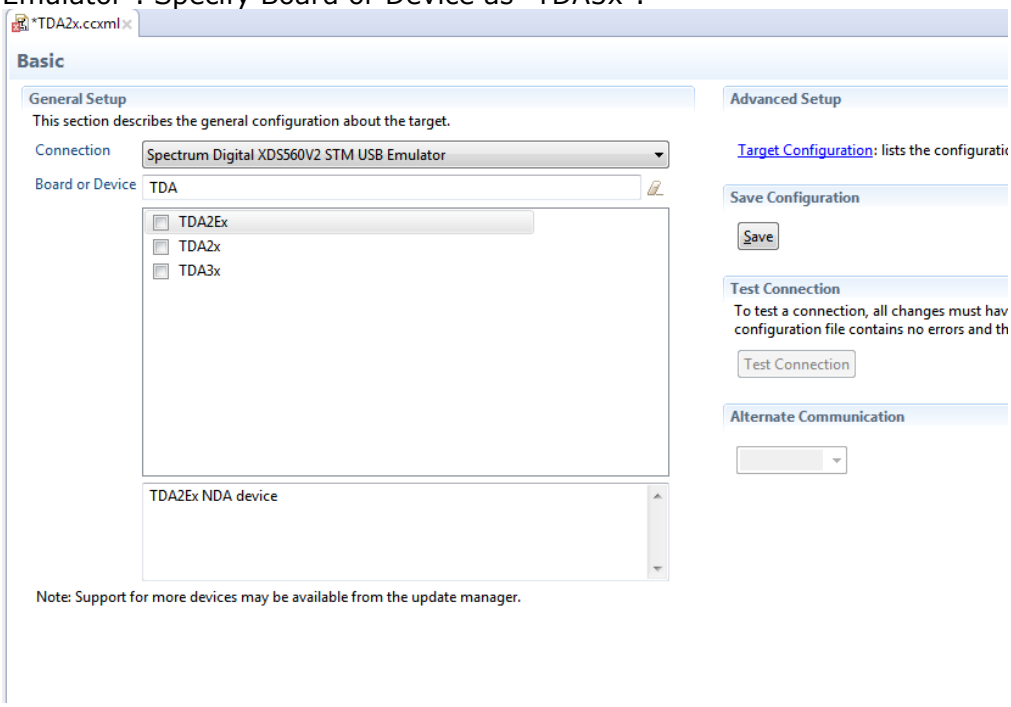
2. CCS Target Configuration creation:
   a. Open "Target Configurations" tab, by navigating through the menu "View ->Target Configurations".



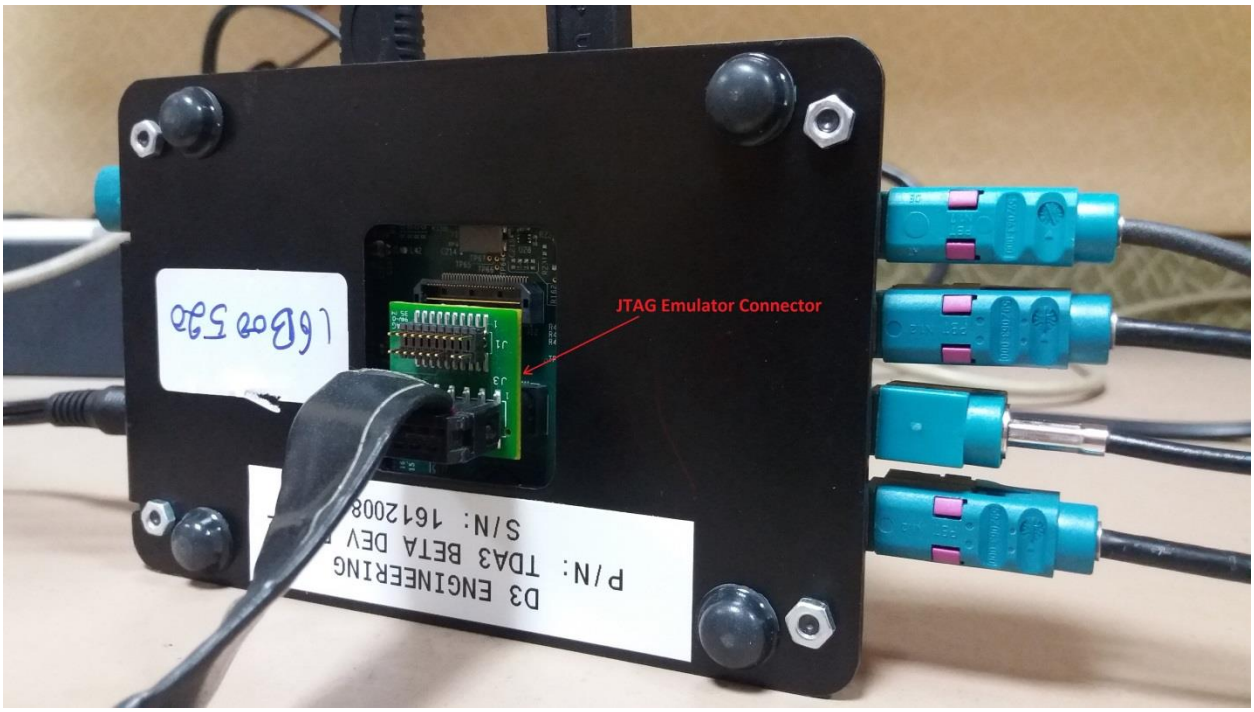
   b. Create a new Target Configuration (TDA3xx Target Configuration) by navigating through the menu "File->New->Target Configuration File".



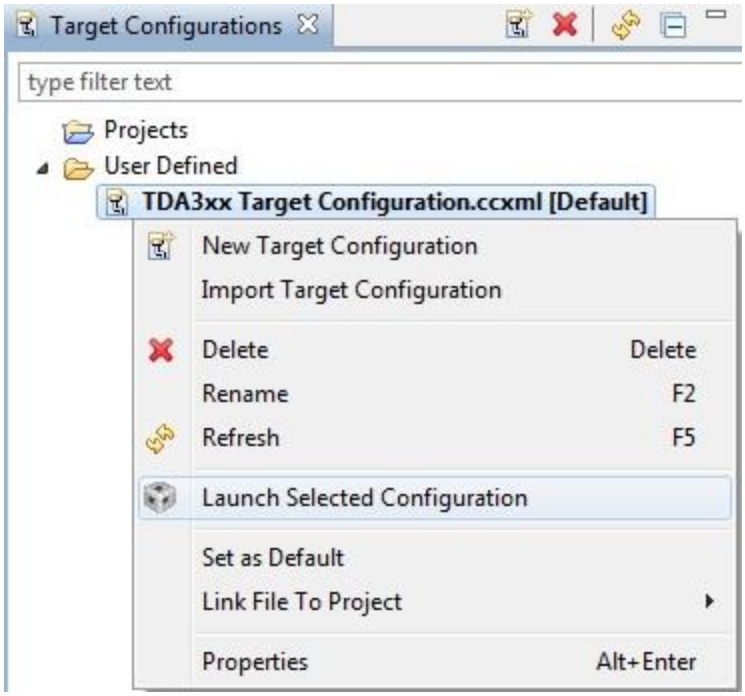
c. Specify Connections as "Spectrum Digital XDS560V2 STM USB Emulator". Specify Board or Device as "TDA3x".

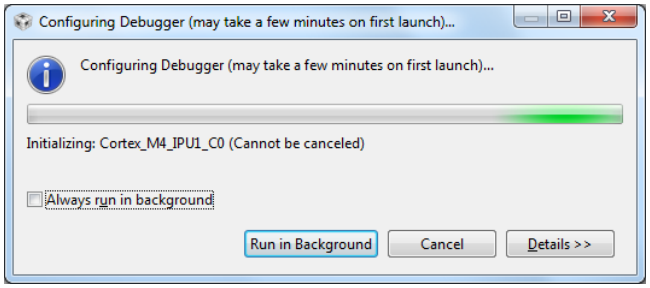


3. Connect JTAG to the board.

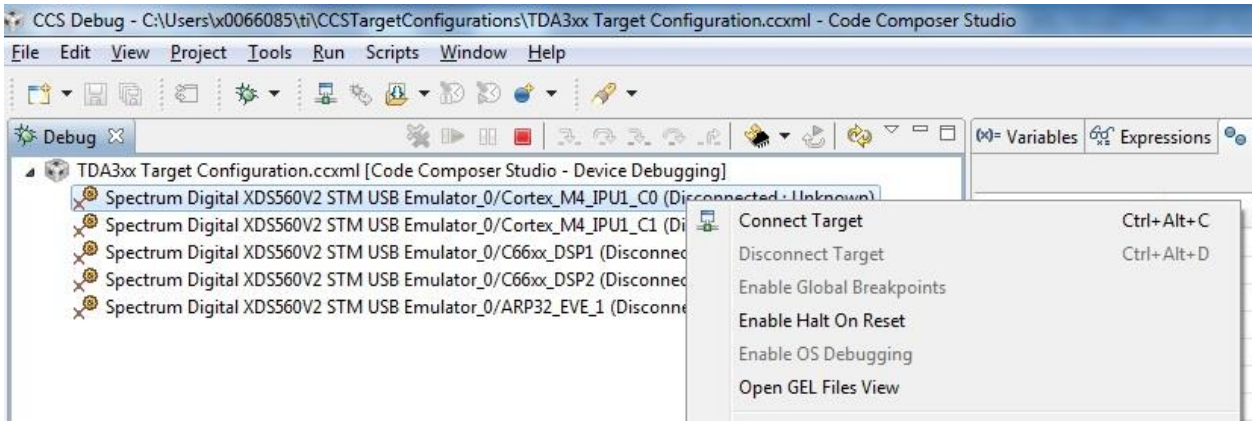


4. Now launch the previously created TDA3xx Target Configuration.





5. Connect to core Cortex_M4_IPU1_C0.



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

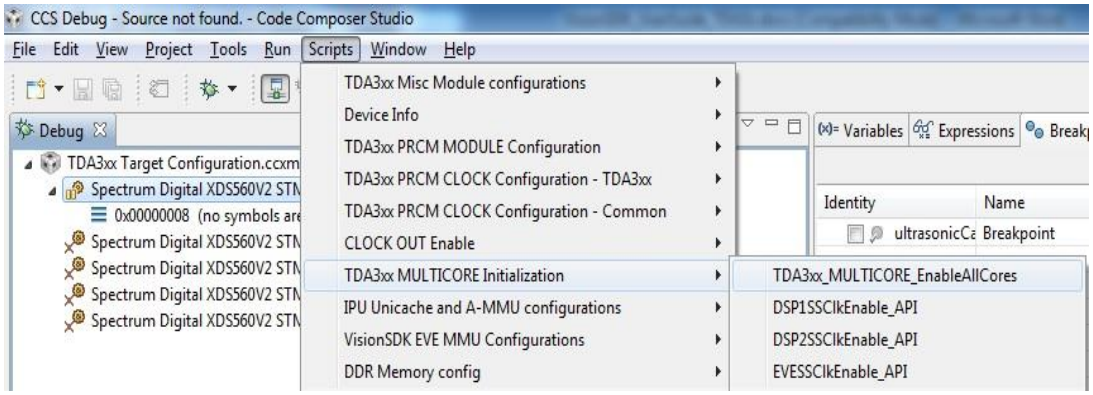
```
Cortex_M4_IPU1_C0: GEL Output: --->>> TDA3xx Target Connect Sequence DONE
!!!!!  <<<---
```

```
NOTE: If you hit the following error message:
        Cortex_M4_IPU1_C0: Trouble Writing Memory Block at 0x55080808 on Page 0 of Length 0x4:
                           Error 0x80002002/-1203 Fatal Error during: Memory, Control,  The DAP
                           access, address 0x55080808, has returned a SLAVE error.
        Cortex_M4_IPU1_C0: GEL: Error while executing OnTargetConnect(): Internal error while
                           writing 0x55080808
                at *((unsigned int *) regAddr)=(unsigned int) 0xA0000000
                        [TDA3xx_multicore_reset.gel:12]
                at AMMU_config() [TDA3xx_startup_common.gel:96]
                at OnTargetConnect()
```

Then,

- Reset the system : Run->Reset->System reset
- Connect once again : Scripts->TDA3xx Misc Module Configurations->OnTargetConnect_API

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

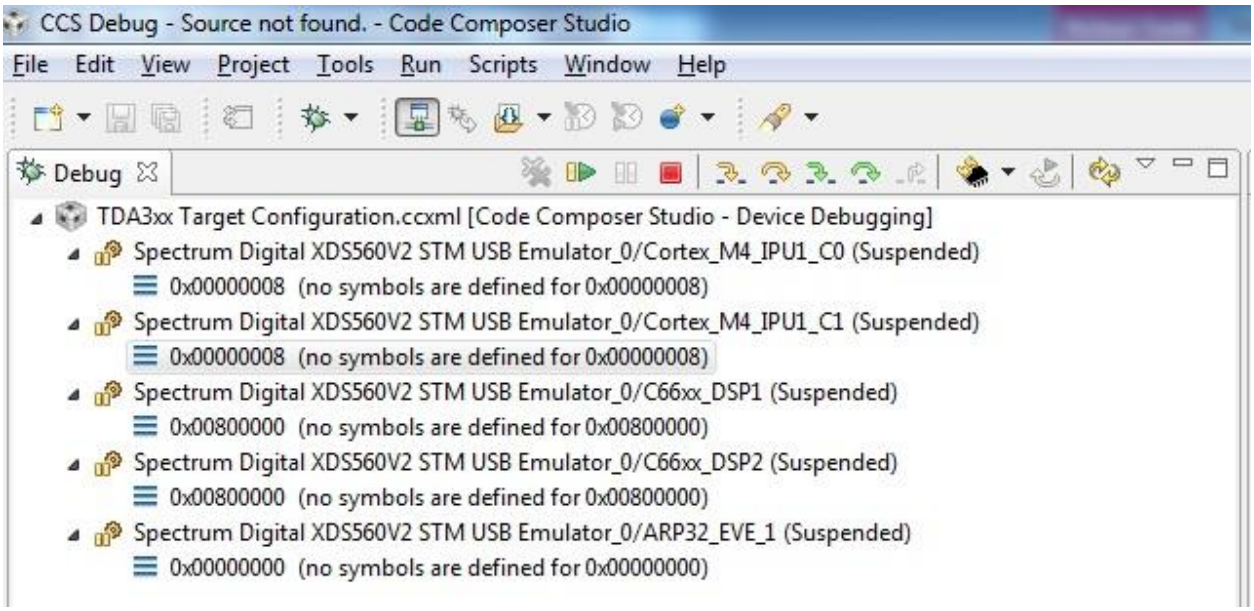


8. On successful script execution, the following log appears on CCS console:

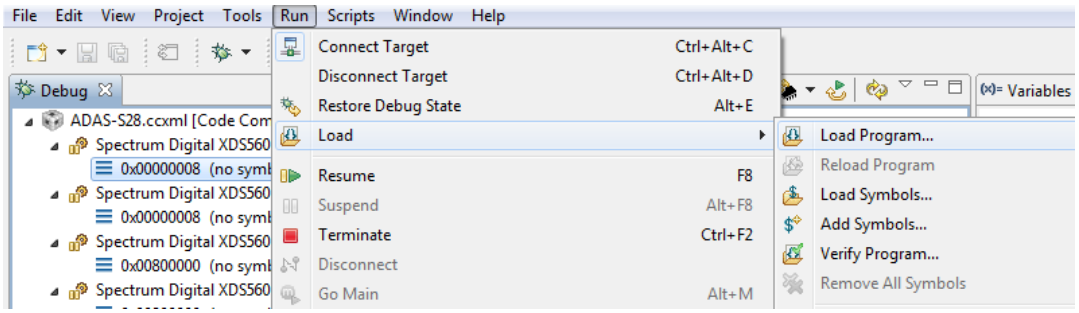
```
Cortex_M4_IPU1_C0: GEL Output: --->>> EVESS Initialization is DONE! <<<---
```

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

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



11. On the cores load the binaries as mentioned below



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

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

3.6 Run the demo

3.6.1 Steps to run

1. Power-on the Board after loading binaries by (SD or CCS) and follow Uart settings to setup the console for logs and selecting demo.

2. Select the Camera Module Types "s: System Settings"-> "Capture Settings" -> "8: AR0140 Sensor for SV - TIDA00262 (TDA3x ONLY)" OR
   "9: OV10640 Sensor for SV - IMI (TDA3x ONLY)"

3. Select the SRV use-cases "5: ISS Use-cases, (TDA3x ONLY)"->
   "5: Surround View Calibration" OR
   "4: 3D SRV 4CH ISS capture + ISS ISP + DeWarp + Synthesis (DSP1) + Display" OR
   "6: 3D + 2D SRV 4CH ISS capture + ISS ISP + DeWarp + Synthesis (DSP1) + Display"

4 Revision History

Version	Date	Revision History
1.0	04th July 2016	Initial Version
1.1	31st October 2016	Updated for Vision SDK ver 2.11
1.2	8th Feb 2017	Updated for Vision SDK ver 2.12
1.3	19th June 2017	Updated linux installer
1.4	29th June 2017	Updated for Vision SDK ver 3.0
1.5	13th October 2017	Updated for Vision SDK ver 3.01
1.6	14th November 2017	Added section for ISS SRV use-cases
1.7	21st December 2017	Updated for Vision SDK ver 3.02
1.8	6th April 2018	Updated release ver 3.3
1.9	25th June 2018	Added 1GB build info

««« § »»»