Vision SDK

(v03.08.00)

Linux 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 multiprocessor 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 family of SoCs

This document particularly explains Linux part of vision SDK where A15 is supposed to run Linux as target OS while other cores in the SOC run SYSBIOS. Build environment is also assumed to be Linux and user should be familiar with basics of Linux to follow this document.

1.1 References

Refer additional documents for more information about Vision SDK.

Refer Index.html under <INSTALL_DIR> folder, this file helps navigating through Vision SDK documentations effectively

2 Installation Overview

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

2.1 PC Requirements

Installation of this release needs a Linux Ubuntu 14.04 machine.

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

2.2 Software Requirements

All software packages required to build and run the Vision SDK are included as part of the SDK release package except for the ones mentioned below.

2.2.1 A15 Compiler, Linker

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

https://developer.arm.com/-/media/Files/downloads/gnu-a/8.3-2019.03/binrel/gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf.tar.xz

The tools need to be installed under

$INSTALL_DIR/ti_components/os_tools/linux/arm/

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/arm
$> tar –xvf gcc-arm-8.3-2019.03-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 libssl-dev

2.2.2 Linux kernel, uboot, sgx driver and target file system

In this vision sdk release kernel, uboot, sgx & target filesystem downloaded/cloned. Links to download/clone are mentioned in software installation section 2.4

2.2.3 Other mandatory software packages for build

Ensure these packages/tools are installed on the installation machine

Ssh, corkscrew, gawk, uname, sed, u-boot-tools, dos2unix, dtrx, 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 lzop

To install

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

On ti-baseline 14.04 ubuntu package can be installed using the script

hlos/scripts/linux/setup-linux-build-env.sh

2.2.4 Code Composer Studio

CCS version 6.0.1.00040 or higher shall be used to debug cores which are running Bios. CCS can be downloaded from the below link-

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

Remove all default gels before connecting the bios core

2.3 Hardware Requirements

Please refer $INSTALL_DIR/vision_sdk/docs/VisionSDK_UserGuide.pdf

IMPORTANT NOTE: This release supports vision sdk only on

TDA2x Rev E or higher version of EVMs

TDA2ex Rev C or higher version of EVMs

TDA2px EVMs

2.4 Software Installation & Setup

Download PROCESSOR_SDK_VISION_XX_XX_XX_XX_setuplinux.bin

```
$> bash
$> ./PROCESSOR_SDK_VISION_XX_XX_XX_XX_setuplinux.bin
```

Accept license agreement and give <installation_directory_absolute_path> e.g. /home/username/foldername/PROCESSOR_SDK_VISION_XX_XX_XX_XX

```
$> cd <installation_directory_absolute_path>
$> export INSTALL_DIR=<installation_directory_absolute_path>
```

2.4.1 One time PC set up

If you are setting up git first time you need to setup your .gitconfig and gitproxy with as shown below

IMPORTANT NOTE: These steps are just indicatory. You may have to figure out arguments (e.g. paths for .gitconfig / git-proxy.sh / arguments to corkscrew) based on your particular system Network. Given below is TI network proxy setting

1. Edit .gitconfig

```
$>vi /home/<username>/.gitconfig
[core]
        gitproxy = none for ti.com
        gitproxy = /home/<username>/git-proxy.sh
```

save and exit (ESC + :wq)

2. Create git-proxy.sh

```
$>vi /home/<username>/git-proxy.sh
exec /usr/bin/corkscrew proxyle01.ext.ti.com 80 $*
```

save and exit (ESC + :wq)

You should see following output if the setup is successful

```
$> git config --list
core.gitproxy=none for ti.com
core.gitproxy=/home/<username>/git–proxy.sh
```

2.4.2 Install Linux Components

2.4.2.1 Essential Components kernel, uboot, sgx, and file system

IMPORTANT NOTE: Steps below are absolutely mandatory and pre-cursors before you move to building vision sdk

You can choose to perform steps 1 – 4 in parallel through different terminals, these need not be sequential, to open terminal use Ctrl + Alt +t

Script at vision_sdk/build/hlos/scripts/linux/setup_linux.sh will do the basic setup (only steps 1 - 3). Run the script from <INSTALL_DIR>/vision_sdk/build

```
$> ./hlos/scripts/linux/setup_linux.sh
```

File System needs to be download, not a part of basic setup setup_linux.sh (refer section)

Note: Script will apply patches only if placed in default location as per release.

$INSTALL_DIR/ti_components/os_tools/linux/kernel/linux-kernel-addon/

2.4.2.1.1 Clone kernel

```
$> cd $INSTALL_DIR/ti_components/os_tools/linux/kernel
$> git clone git://git.ti.com/glsdk/psdkla-kernel.git omap
$> cd omap/
$> git checkout -b kernel_dev tags/REL_VISION_SDK_03_08_00_00
Or
$> git checkout -b kernel_dev 3d03684
```

memcahe.ko source

There is additional kernel module needed for vision_sdk, ensure the source lies under $INSTALL_DIR/ti_components/os_tools/linux/kernel/linux-kernel-addon/memcache, this comes by default with installation.

Cmem package needs to be installed from another git

```
$> cd $INSTALL_DIR/ti_components/os_tools/linux/kernel/cmem
$> git clone git://git.ti.com/ipc/ludev.git
$> cd ludev/
$> git checkout -b cmem_dev 4f970f0
Or
$> git checkout tags/4.16.00.00
```

2.4.2.1.2 Clone u-boot

```
$> cd $INSTALL_DIR/ti_components/os_tools/linux/u-boot
$> git clone git://git.ti.com/glsdk/psdkla-u-boot.git u-boot
$> cd u-boot/
$> git checkout -b uboot_dev tags/REL_VISION_SDK_03_08_00_00
Or
$> git checkout -b uboot_dev f454ae0
```

2.4.2.1.3 Clone sgx drivers

```
$> cd $INSTALL_DIR/ti_components/os_tools/linux/sgx
$> git clone git://git.ti.com/graphics/omap5-sgx-ddk-linux.git
```

```
$> cd omap5-sgx-ddk-linux/
$> git checkout -b sgx_dev 4519ed3
```

2.4.2.1.4 Download and untar file system

Download Linux file system

tisdk-rootfs-image-dra7xx-evm_vsdk_3_8.tar.xz from ti.com - http://software-dl.ti.com/processor-sdk-vision/esd/TDAx/vision-sdk/latest/index_FDS.html

And keep under following directory

$INSTALL_DIR/ti_components/os_tools/linux/targetfs

Untar file system

```
$> bash
$> export INSTALL_DIR=<installation_directory_absolute_path>
$> cd $INSTALL_DIR/ti_components/os_tools/linux/targetfs
$> chmod 777  ../targetfs
NOTE: Only targetfs folder needs to have full permission not the files within
$> tar xf tisdk-rootfs-image-dra7xx-evm_vsdk_3_8.tar.xz
$> exit
```

Note:(for M4 Display): HDMI display and 10.1" LG lcd is only supported.

2.4.2.2 Optional Components ipumm, Codec Engine and Framework components

Certain configurations require ipumm. Ipumm can be cloned from the below mentioned git repo along with codec engine and framework components.
NOTE: These are optional and default vision_sdk_linux build does not need these.
Note: Optional components are not setup as part of setup_linux.sh script

2.4.2.2.1 Clone ipumm

```
$> cd $INSTALL_DIR/ti_components/codecs
$> git clone git://git.ti.com/ivimm/ipumm.git
$> cd ipumm/
$> git checkout -b ipumm_dev 365a9a5402c829710b7e1eefec07d26b3c94c3a9
```

2.4.2.2.2 Codec_Engine

```
$> Install codec engine ver codec_engine_3_24_00_08 in the following folder
<INSTALL_DIR>/ti_components/codecs/
Package can be downloaded from
http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/ce/3_24_00_08/index_FDS.html
```

2.4.2.2.3 Framework Components

Download FC version x_xx_xx_xx from below link and use when set IPUMM=yes in the build configuration. The FC version packaged along with VSDK is a patched version for IVA-HD profiling and that patched version will not work with IPUMM.

Note: FC Path and version supported in release can be refer from tools_path.mk in

<INSTALL_DIR>/vision_sdk/build

http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/fc/3_40_02_07/index_FDS.html

2.4.2.2.4 *Opencl supported package*

These below package will be needed for OpenCL build and debugging dsp opencl code

Kernel repositories required

dsptop package needs to be installed from another git

```
$> cd $INSTALL_DIR/ti_components/os_tools/linux/kernel
$> git clone git://git.ti.com/sdo-emu/dsptop.git
$> cd dsptop/
$> git checkout -b dsptop_dev 0aedcab
```

gdbc6x package needs to be installed from another git

```
$> cd $INSTALL_DIR/ti_components/os_tools/linux/kernel
$> git clone git://git.ti.com/sdo-emu/gdbc6x.git
$> cd gdbc6x/
$> git checkout -b gdbc6x_dev df0b8f6
```

Filesystem patches for opencl

Copy "opencl_fs_patches.tar.gz" from $INSTALL_DIR/ti_components/os_tools/linux/kernel/linux-kernel-addon/fs-patches into $ INSTALL_DIR/ti_components/os_tools/linux/targetfs and untar

```
tar –xvzf opencl_fs_patches.tar.gz
```

2.4.3 Un-installation

```
$> rm -rf $INSTALL_DIR
```

3 Build

Important Note: In this release reference to variations in platform names would be found, basically they resemble same platform for informational-adas

Tda2xx as dra7x or vice versa

Tda2ex as dra72 or vice versa

Tda2ex 17x17 as dra71 or vice versa

Tda2px as dra76 or vice versa

For building binaries follow these steps

3.1 Build Linux Vision SDK for Video Capture and Display use-cases

Note: If you are trying this after 3.2 ensure you do a 'make clean' and manually delete $INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/$(MAKECONFIG) folder and then proceed

1. You must have followed all the steps in software installation and setup section 2.4 before you proceed
2. Select make config in Rules.make

   TDA2XX

   MAKECONFIG=**tda2xx_evm_linux_all**

TDA2EX

MAKECONFIG=**tda2ex_evm_linux_all**

TDA2EX 17x17

MAKECONFIG= **tda2ex_17x17_evm_linux_all**

TDA2PX

MAKECONFIG=**tda2px_evm_linux_all**

3. Select the Lens Module used for the camera:

   For TDA2xx, copy the binary files CALMAT.bin and CHARTPOS_RUBICON.BIN to SD Card from vision_sdk/apps/tools/surround_vision_tools/Srv_LUTs/TDA2X/

   For TDA2PX, copy vision_sdk/apps/tools/Lens_params/LENS_imi.BIN, vision_sdk/apps/tools/surround_vision_tools/Srv_LUTs/TDA2X/CALMAT.bin and vision_sdk/apps/tools/surround_vision_tools/Srv_LUTs/TDA2X/ CHARTPOS_RUBICON.BIN to SD Card.

4. Build the Linux dependencies, this will build kernel, u-boot and sgx drivers & memcache.ko

   NOTE: This step is needed only first time you build or if you make any changes to u-boot/kernel/sgx drivers, otherwise this can be skipped.

```
$> cd $INSTALL_DIR/vision_sdk/build
$> make linux
$> make linux_install
```

5. Build the sdk

```
$>make –s –j depend
$>make –s –j
```

   Executing "**make –s –j depend** "will build all the necessary components (PDK drivers, EDMA drivers) and "**make –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

6. Step 5 will ensure your firmware binaries and linux application's .out are copied into your targetfs i.e. $INSTALL_DIR/ti_components/os_tools/linux/targetfs/lib/firmware and $INSTALL_DIR/ti_components/os_tools/linux/targetfs/opt/vision_sdk respectively.

   It will also execute make linux_install to copy all binaries needed to boot into $INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot and $INSTALL_DIR/ti_components/os_tools/linux/targetfs/boot

7. The file system under $INSTALL_DIR/ti_components/os_tools/linux/targetfs now can be used as either NFS or rootfs on sd card.
    a. Tar the file system and keep it in $INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot folder.

    Important Note: With 4.19 kernel filesystem 4GB and above card is required.

```
$>export INSTALL_DIR=<installation_directory_absolute_path>
$>cd $INSTALL_DIR/ti_components/os_tools/linux/targetfs
$>tar cvf tisdk-rootfs-image-dra7xx-evm.tar.xz ./*
$>mv ./tisdk-rootfs-image-dra7xx-evm.tar.xz
$INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot
$>exit
```

3.1.1 NFS + SD boot

Important Note: You need to keep file system in two places (idiosyncrasy of 4.4 kernel). The zImage and dra7-evm-infoadas.dtb/dra72-evm-infoadas.dtb/ dra71-evm-infoadas.dtb is picked up from /boot folder on the sd card always while firmware binaries are picked up from $ $INSTALL_DIR/ti_components/os_tools/linux/targetfs /lib/firmware

$INSTALL_DIR/ti_components/os_tools/linux/targetfs is used as NFS

For NFS to work, $ INSTALL_DIR/ti_components/os_tools/linux/targetfs needs to be exported from /etc/exports of installation machine.

Refer section 4.2 to prepare SD card for NFS.

Note: NFS Boot is failing in case of TDA2ex 17x17 for this release

3.1.2 SD only boot

The tarred file system needs to be flashed on sd card along with uboot and uenv.txt refer section 4.2

- Change the dtb picked in the uenv.txt depending upon platform

  By default fdtfile=dra7-evm-infoadas.dtb for tda2x

  fdtfile=dra7-evm-infoadas.dtb → fdtfile= dra72-evm-infoada.dtb for tda2ex

  fdtfile=dra7-evm-infoadas.dtb → fdtfile= dra76-evm-infoada.dtb for tda2px

  fdtfile=dra7-evm-infoadas.dtb → fdtfile= dra72-evm-infoada.dtb for tda2ex_17x17

3.2 Build Linux Vision SDK for AVB Capture, Decode and Display UCs

Note: if you are trying this after 3.1, ensure you do a 'make clean' and manually delete $INSTALL_DIR/vision_sdk/binaries/$(MAKECONFIG) folder and then proceed

NOTE: AVB is not supported on tda2px in this release

- You must have followed all the steps in software installation and setup section 2.4
- **Ensure that** A15_TARGET_OS := Linux and NDK_PROC_TO_USE=ipu2 in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk

- **IMP:** Patch the dra7-evm.dts (or dra72-evm-common.dtsi for TDA2ex/TDA2ex 17x17) from kernel (ti_components/os_tools/linux/kernel/omap/arch/arm/boot/dts)as shown below

```
Change below
&mac {
        status = "okay";
to
&mac {
        status = "disabled";
```

  Basically you need to disable mac from Kernel so VSDK can use it exclusively.

- Build the linux dependencies, this will build kernel, u-boot and sgx drivers

  NOTE: This step is needed only first time you build or if you make any changes to u-boot/kernel/sgx drivers, otherwise this can be skipped.

```
$> cd  $INSTALL_DIR/vision_sdk/build
$> make linux -j
$> make linux_install -j
$> make -s -j depend
$> make -s -j
```

- Disable all cores in \vision_sdk\$(MAKEAPPNAME)\configs\$(MAKECONFIG)\cfg.mk except IPU1_0, IPU2 and A15 using PROC_<CORE_NAME>_INCLUDE
- Refer steps 4 to 7 in section 3.1

3.3 Build Linux Vision SDK for LCD display (M4 Display)

Note: Only TDA2xx and TDA2ex support 10.1" LG LCD.

- You must follow all the steps in software installation and build.
- Change the dtb picked in the uenv.txt from

  fdtfile=dra7-evm-infoadas.dtb → fdtfile= dra7-evm-infoadas-lcd-lg.dtb for tda2xx

  fdtfile=dra72-evm-infoadas.dtb → fdtfile=dra72-evm-infoadas-lcd-lg.dtb for tda2ex

3.4 Build Linux Vision SDK for fast boot (Early boot and late attach of remote cores)

Apply the earlyboot-lateattach-patches for kernel and u-boot

```
cd $INSTALL_DIR/ti_components/os_tools/linux/kernel/omap
git am ../linux-kernel-addon/ earlyboot-lateattach-patches/kernel/*

cd $INSTALL_DIR/ti_components/os_tools/linux/u-boot/u-boot
git am ../linux-kernel-addon/ earlyboot-lateattach-patches/u-boot/*
```

1. You must have followed all the steps in software installation and setup section 2.4 before you proceed
2. Select make config in Rules.make

   TDA2XX

   ```
   MAKECONFIG=tda2xx_evm_linux_all
   ```

3. Select the Lens Module used for the camera:

   For TDA2xx, copy the binary files CALMAT.bin and CHARTPOS_RUBICON.BIN to SD Card from vision_sdk/apps/tools/surround_vision_tools/Srv_LUTs/TDA2X/

   For TDA2PX, copy vision_sdk/apps/tools/Lens_params/LENS_imi.BIN, vision_sdk/apps/tools/surround_vision_tools/Srv_LUTs/TDA2X/CALMAT.bin and vision_sdk/apps/tools/surround_vision_tools/Srv_LUTs/TDA2X/ CHARTPOS_RUBICON.BIN to SD Card.

4. Build the Linux dependencies, this will build kernel, u-boot and sgx drivers & memcache.ko

   NOTE: This step is needed only first time you build or if you make any changes to u-boot/kernel/sgx drivers, otherwise this can be skipped.

   ```
   $> cd $INSTALL_DIR/vision_sdk/build
   $> make linux
   $> make linux_install
   ```

5. Build the sdk

   ```
   $>make –s –j depend
   $>make –s –j
   ```

   Executing "**make –s –j depend** "will build all the necessary components (PDK drivers, EDMA drivers) and "**make –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

6. Step 5 will ensure your firmware binaries and linux application's .out are copied into your targetfs i.e. $INSTALL_DIR/ti_components/os_tools/linux/targetfs/lib/firmware and $INSTALL_DIR/ti_components/os_tools/linux/targetfs/opt/vision_sdk respectively.

   It will also execute make linux_install to copy all binaries needed to boot into $INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot and $INSTALL_DIR/ti_components/os_tools/linux/targetfs/boot

7. The file system under $INSTALL_DIR/ti_components/os_tools/linux/targetfs now can be used as either NFS or rootfs on sd card.

   a. Tar the file system and keep it in $INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot folder.

      Important Note: With 4.19 kernel filesystem 4GB and above card is required.

```
$>export INSTALL_DIR=<installation_directory_absolute_path>
$>cd $INSTALL_DIR/ti_components/os_tools/linux/targetfs
$>tar cvf tisdk-rootfs-image-dra7xx-evm.tar.xz ./*
$>mv ./tisdk-rootfs-image-dra7xx-evm.tar.xz
$INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot
$>exit
```

Prepare SD card Ref (4.3) Preparing SD card & Boot

Only when the above steps are completed run the early-boot-update.sh utility, or else the system boot will fail due to missing dependencies. **If you intend to boot from QSPI, skip running the early-boot-update script and jump to Section 3.4.1 of this file.**

This command will delete existing uImage, zImage, uenv.txt, dtb and remote-core firmware present in the boot partition and update it with the necessary files.

early-boot-update.sh <makeconfig><path to target FS><path to boot-partition>

Argument 1 is mandatory and indicates the current configuration being built. The list of valid arguments inlcude "tda2xx_evm_linux_all"
Argument 2 is mandatory and indicates path where the Linux target file-system is present
Argument 3 is mandatory and indicates the path to the target boot partition. Usually this is /media/<username>/boot

Early-boot + Late-attach is supported on TDA2xx, TDA2Ex and TDA2Px

Important Note: By default, LPAE is enabled for this release. This facilitates running VSDK-Linux with and without early-boot + late attach on boards with more than 2GB RAM. For graphics intensive use-cases (which need more than 128MB of graphics memory), issues may be observed.

3.4.1 QSPI boot

From Vision-SDK 3.07 support is enabled for loading of remote-core firmware from QSPI. Below are the steps to be followed to flash binaries to QSPI.

Note: With Vision-SDK 3.08, support for loading the remote-core firmwares from QSPI is not available.

Software pre-requisites:

We will use the mkimage binary from u-boot-tools package for wrapping binaries in uImage header. This package can be installed by:

sudo apt-get install u-boot-tools

Follow instructions mentioned in Section 3.4, before running the below steps.

i. Connect a USB cable from P2/USB1 to host PC. This is used for flashing the EVM using fastboot

ii. Connect a USB cable from the USB-UART adapter on the EVM to the host PC

iii. Change the switch settings on the EVM to as below

SW2[7:0] 0000 0111

SW3[7:0] 1000 0001

SW5[9:0] 00 0001 0100

iv. Reboot the board with the SD-card. Halt at u-boot and run the below commands

=> env default –fa

=> saveenv

v. Reboot the board with the SD-card, and halt again at u-boot

=> fastboot 0

vi. Run the below command from the PC, which will flash the contents to the QSPI

sudo flash-qspi.sh <makeconfig> <path_to_linux_targetfs> <path to MLO>

vii. After this is complete, the contents are copied to QSPI. Now change the switch settings to below. The SD-card can be removed from the board

SW2[7:0] 0011 0111

viii. Reboot the board, you'll notice the early boot from QSPI

3.5 Early M4 based chains for VSDK-Linux builds

From Vision-SDK 3.5, support is provided for launching chains from M4, to facilitate early use-case lauch. A sample application which executes the below sequence of steps is provided as part of the release:

1. U-boot loads Primary IPU, DSP firmware
2. Primary IPU has a single camera capture chain which is enabled -- OV-10635 capture begins and displays on screen **(early M4 chain)**
3. Linux Initialization occurs in parallel, video continues to be displayed on the screen
4. Once Linux initialization is complete, and vision-SDK initialization is complete (running apps.out) Weston is displayed in the background **(verification of late-attach and late-init of chain using VDRM based Weston)**

To enable this feature, set EARLY_USECASE_ENABLE=yes in the apps/configs/<evm_id>_linux_all/cfg.mk file, and follow the instructions in Sec 3.4 of this document.

To ensure the above use-case works, the setup needs to have a vision-daughter card, and a de-serializer board to which OV-10635 cameras are connected. It is recommended that before trying the early use-case, the multi-camera capture use-case using OV-10635 is validated along with Weston background use-case using the traditional VSDK-Linux boot-flow. Details on getting VDRM based Weston can be found in section 4.3.4 of this document and in the below link:

http://processors.wiki.ti.com/index.php/Virtual_DRM_:_An_User_Guide_for_Developing_Usecases)

Once the **"EARLY_USECASE_ENABLE"** flag is set, the application is limited to only launching the Weston window in the background, and no additional chains / UART operations are feasible.

3.6 Build Linux Vision SDK for Autosar

1. Apply Early-boot patches on u-boot from /linux-kernel-addon/earlyboot-lateattach-patches/u-boot/* and kernel patches /linux-kernel-addon/earlyboot-lateattach-patches/kernel/* on linux kernel
2. Open ti_components/os_tools/linux/kernel/omap/arch/arm/boot/dts/dra7-evm-infoadas.dts
3. Change below

```
&ipu1 {
      /delete-property/ watchdog-timers;
};

&ipu2 {
      /delete-property/ watchdog-timers;
 timers= <&timer9> , <&timer11>;
};
To
&ipu1 {
```

```
        status= "okay";
        /delete-property/ watchdog-timers;
    timers= <&timer9> , <&timer11>;
};
DISABLE_COMPLETE(ipu2);
```

4. Open vision_sdk\apps\configs\tda2xx_evm_linux_all\cfg.mk and set

   PROC_IPU1_0_INCLUDE = yes

   PROC_IPU2_INCLUDE = no

   AUTOSAR_APP = yes

   IPU_PRIMARY_CORE = ipu1_0

   IPU_SECONDARY_CORE = ipu2

5. 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
```

6. Follow step 1 to 5 in section 3.4

   It will build all cores except IPU2.

7. Build only IPU2 by disabling all cores except IPU2.
8. Install Autosar.

```
$> cd $INSTALL_DIR/vision_sdk/build
$> make autosar_install
```

9. Follow step 7 in section 3.4
10. Run early-boot-update.sh utility with as described in section 3.4 with makeconfig as *tda2xx_evm_linux_autosar.*

3.7 Build Linux Vision SDK file-system

From Vision-SDK 3.4, there are changes in the file-system build to enable a smaller file-system. While the file-system provided as part of the Processor-SDK Linux automotive is ~1GB, the file-system as part of Vision-SDK release is ~100MB. This reduction in size is achieved by removing components not required for traditional ADAS use-cases.

To rebuild the Vision-SDK file-system follow the below instructions:

1. Build the Yocto file-system by following instructions as part of the Processor SDK Linux Automotive SW development guide

2. Apply the patches present in the linux-kernel-addon/fs-patches/yocto/meta-ti folder of Vision-SDK to the tisdk/sources/meta-ti folder in the yocto repository.

3. Apply the patches present in the linux-kernel-addon/fs-patches/yocto/meta-arago folder of Vision-SDK to the tisdk/sources/meta-arago folder in the yocto repository.

4. Rebuild the file-system by running the bitbake command as documented in the Processor SDK Linux Automotive SW development guide

The changes in meta-arago are for the reduction in file-system size and IPUMM based gstreamer decoding while the changes in meta-ti are for the VDRM based update to SGX libraries.

4 Run

4.1 Board Setup

Refer corresponding setup documents from $INSTALL_DIR/vision_sdk/docs/VisionSDK_UserGuide_TDAxxx.pdf

Note: The setup is different for 1ch / 4ch LVDS capture + SGX display and AVB->Decode->SgxDisplay usecase.

4.1.1 Capture switch setting

Video Config pins needs to set for different capture inputs

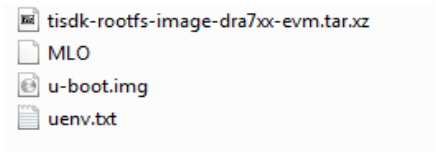
VIDEO CONFIG switch settings (SW3 on TDA2xx Vision Application Board (set for Ov10635 in Original version of CPLD))

Capture Type	**Hardware controlled pin settings Vision Application Board (Rev C CPLD) (default cpld image)**							
	1	2	3	4	5	6	7	8
OV10635	**OFF**	**ON**	**OFF**	**ON**	**OFF**	**ON**	**OFF**	**ON**
LVDS	**OFF**	**OFF**	**ON**	**OFF**	**OFF**	**ON**	**OFF**	**ON**
HDMI	**OFF**	**OFF**	**ON**	**ON**	**OFF**	**ON**	**OFF**	**ON**

4.2 Preparing SD card & Boot

1. Connect micro SD card to installation machine
2. uenv.txt changes for boot
    a. *NFS + SD boot*
        i. Copy uenv_nfs.txt at $INSTALL_DIR/vision_sdk/build/hlos/scripts/linux to $INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot and rename it to uenv.txt
        ii. Update uenv.txt in folder to have right nfs path & server path
    b. *SD only boot – Nothing to be done*
        i. Copy and renaming is taken care by the build system.
3. Ensure $INSTALL_DIR/vision_sdk/binaries/$(MAKEAPPNAME)/<MAKECONFIG>/hlos/linux/boot folder appears as shown in picture after build

   Ensure the built filesystem is copied from the location (section) $INSTALL_DIR/ti_components/os_tools/linux/targetfs



4. Format SD card and create two partitions (boot (FAT32) and rootfs(ext4)) using script below, <parent_device_name> can be found using "mount" command

```
$> bash
$> cd <INSTALL_DIR>/build
$> sudo ./hlos/scripts/linux/mksdboot.sh --device /dev/<parent_device_name> --appname <MAKEAPPNAME>
--makeconfig <MAKECONFIG>
$> exit
```

(MAKEAPPNAME and MAKECONFIG should be same as set in Rules.make)

5. Disconnect SD card from installation machine and insert it into EVM micro SD slot.
6. Set 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) [0:15] to the below shown position

SYSBOOT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	SW2								SW3							
SW Pin	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
SW Position	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	ON

IMPORTANT NOTE: Pin setting for J6 Rev G 2.0 Silicon is as mentioned below

SYSBOOT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	SW2								SW3							
SW Pin	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	**8**
SW Position	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	**OFF**

7. Power on the EVM, press power reset

4.3 Run demos

1. Log in as root on the UART console

$> dra7xx-evm login: root

root@dra7xx-evm:~#

Following errors may be seen on display and the message can be ignored.

Trying to boot from MMC1

reading dra7-ipu2-fw.lzop

spl_load_file_fat: error reading file dra7-ipu2-fw.lzop, err - -1

spl: error reading image dra7-ipu2-fw.lzop, err - -1

Failed to load module: /usr/lib/gbm/gbm_dri.so: cannot open shared object file: No such file or directory
failed to load module: /usr/lib/gbm/gbm_gallium_drm.so: cannot open shared object file: No such file or directory

2. Proceed on UART console & execute

(TDA2XX/TDA2EX/TDA2EX 17x17)

```
$> cd /opt/vision_sdk
$> ./load_ocl_kos.sh (needed only for opencl usecases)
$> source ./opencl_env.sh (needed only for opencl usecases)
$> source ./vision_sdk_load.sh
$> ./$(MAKEAPPNAME).out
Select the demo to run
```

(TDA2XX – TIDA00455/OV490 multi-channel use-case)

```
$> cd /opt/vision_sdk
$> ./load_ocl_kos.sh (needed only for opencl usecases)
$> source ./opencl_env.sh (needed only for opencl usecases)
$> source ./vision_sdk_load.sh
$>./vision_sdk_ov490_pinmux.sh
$> ./$(MAKEAPPNAME).out
Select the demo to run
```

(TDA2PX – ISS single/multi-channel use-case)

```
$> cd /opt/vision_sdk
$> source ./vision_sdk_load.sh
$> ./$(MAKEAPPNAME).out
Select the demo to run
```

(TDA2XX/TDA2EX – OpenVX use-case)

```
$> cd /opt/vision_sdk
$> source ./vision_sdk_load.sh
$>Copy test_data folder from the ti_components/open_compute/tiovx_01_00_01_00/conformance_tests/ to /opt/openvx/test_data
$> export VX_TEST_DATA_PATH=/opt/openvx/test_data
$> ./$(MAKEAPPNAME).out
Select the OPenVX demo to run
```

IMPORTANT NOTE: After copying all required files, please make sure all folder and files should have same user and group id. If it is root then please change it using chown command.

SGX Load bars and prints

For SGX loads run the following

Option 1:
Run the demo as normal
Option another terminal (telnet)
cd /opt/vision_sdk
./pvrscope -f 0

Option 2:
Follow below commands to run the demo
cd /opt/vision_sdk
source ./vision_sdk_load.sh
./$(MAKEAPPNAME).out &
sleep 10
./pvrscope -f 0 &
fg
Enter demo option

IMPORTANT NOTE: Some kernel logs and errors failure message will appear while running "source ./vision_sdk_load.sh", this is caused when DSS interrupts are disabled on A15 using omapconf write, and these errors need be ignored as it would not harm the execution

4.3.1 AVB usecases

As we do not have AVB cameras, encoded video streams being streamed from Linux PC is used as AVB source.

IMPORTANT NOTE: Ensure Ethernet cable is connected to port Ethernet0 (Ethernet port close to HDMI), and run the avb talker application after you choose the demo option

4.3.1.1 Running avb talker binary on host PC

1. Copy $INSTALL_DIR/ti_components/networking/avbtp_talker/avbtp_talker.sh and 1.AVI (input file) to host PC
2. On Host machine (connected to EVM by Ethernet cable), run following command

   $> cd <avb_talker_install_directory>

   $> ./avbtalker.sh 1.AVI 1.AVI 1.AVI 1.AVI

4.3.2 3D surround view demos

Refer VisionSDK_UserGuide_TDA2xx.pdf (section Hardware Requirements) & VisionSDK_SurroundView_DemoSetUpGuide.pdf to set-up the surround view demos and run the use-case.

Refer VisionSDK_3D_SurroundView_Calibration_UserGuide_TDA2xx.pdf for 3D surround view calibration use-case.

Erase the calibration table, when run the demo first time on any new set-up after boot-up by

rm –rf /home/root/.calibtable

NOTE: In case of green artifacts on display, try changing toggling polarity of capture using the script ./VipClockInversion.sh

4.3.3 TIDA00455/OV490 based capture, 3D surround view and SGX display

Refer to section 4.3 for steps to run the use-case. In case of green artifacts on display, try changing toggling polarity of capture using the command "omapconf write 0x4A002534 0x0" or "omapconf write 0x4A002534 0x5". 0x5 is the recommended setting – it might vary based on different OV490 firmwares/board versions/etc. This command can be run before running $(MAKEAPPNAME).out or while the use-case is running from another terminal by connecting via telnet.

4.3.4 Linux DRM usecase

Weston is the default Linux DRM application running on the device at boot up. The weston screen can be seen on the display when the DispDistSrc based usecase is run by selecting "1: Single Camera Usecases" -> "8: DispDistSrc (weston) + Display (1920x1080 HDMI)".

For developing usecases based on Linux DRM applications and vision SDK applications, refer to the Virtual DRM usecase development guide here: http://processors.wiki.ti.com/index.php/Virtual_DRM_:_An_User_Guide_for_Developing_Usecases

4.3.5 With UB964 EVM (CSI2 based capture only)

Refer UB964 EVM (CSI2 based capture only) 2.3.1 section on bios user guide for CSI2 hardware setting for surround view settings.

NOTE: For Tda2px refer $INSTALL_DIR/vision_sdk/docs/VisionSDK_UserGuide_TDA2px.pdf

4.3.6 Linux Autosar Usecase

1. Select option 1: Single Camera Usecases
2. Select option 9: 1CH VIP capture + SGX Copy + Display + Autosar
3. Press 'm' to send a message to Autosar and receive the acknowledgement.

4.4 IPUMM based decode use-case using Gstreamer

4.4.1 IPUMM and VSDK-Linux on IPU-2

From Vision-SDK 3.5, support is provided for IPUMM based decode using GStreamer plugin. Follow section 2.4.2.2 of this document to include IPUMM as part of the primary IPU (IPU-2) build. Please note when IPUMM is included, **IVAHD needs to be set to no** in the corresponding apps/configs/<evm_id>_linux_all/cfg.mk file.

Gstreamer playback requires vpe plugin to be available, so A15 accesses vpe.

Set VPE_INCLUDE=no in apps/configs/<evm_id>_linux_all/cfg.mk

Comment DISABLE_COMPLETE(vpe); statement in the device tree file to make vpe available for A15.

Once the firmware is built, a script 'decode_ipumm.sh' is provided in the /home/root folder of the file-system which is capable of decoding a video and displaying it to the connected. For this script to work, below dependencies must be satisfied:

1. A mp4 file must be copied to the /home/root folder, and must be renamed test.mp4
2. Weston must be running on the screen (refer section 4.3.4 of this document)

If the above two dependencies are met, run decode_ipumm.sh and the video will be displayed on the screen

4.4.2 IPUMM on VSDK-Linux and IPU-1 as primary IPU for Vision-SDK

If the primary IPU is set as IPU-1, and the IPUMM firmware which is built as part of the processor-SDK Linux is included as the IPU-2 firmware, then the following changes are needed.

1. Remove the IPU-2 entries from the kernel device-tree and enable IPU-1

```
- &ipu2_cma_pool {
-  reg = <0x0 0x99000000 0x0 0x5000000>;
- };
- &ipu2 {
-  /delete-property/ watchdog-timers;
-  timers= <&timer9> , <&timer11>;
- };
&ipu1 {
-   status= "disabled";
  /delete-property/ watchdog-timers;
};
```

2. Remove late-attach properties (if any) from the device tree for the the IPU-2 core / MMU

3. Make the following changes in the apps/configs/<evm_id>_linux_all/cfg.mk file

PROC_IPU1_0_INCLUDE=yes

PROC_IPU1_1_INCLUDE=no

PROC_IPU2_INCLUDE=no

IPU_PRIMARY_CORE=ipu1_0

IPU_SECONDARY_CORE=ipu2

IPUMM_INCLUDE=no

IVAHD_INCLUDE=no

Once these changes are made, perform a clean build. Ensure that the file /lib/firmware/dra7-ipu2-fw.xem4 is the IPUMM firmware which is built as part of the Processor-SDK Linux release. With this configuration, IPU-1 runs the Vision-SDK firmware while IPU-2 runs IPUMM which performs decode operations. Refer to section 4.4.1 on how to run decode use-cases.

NOTE: Early boot-strap of stand-alone IPUMM firmware included as part of the Processor-SDK Linux release is not supported. However, if IPUMM is built as part of the Vision-SDK firmware, Early-boot + late-attach of the Vision-SDK + IPUMM firmware is supported.

NOTE: Gstreamer has been modified to support buffer allocation using OMAPDRM even when the display is running on a different DRM card, which in the VSDK-Linux scenario is VDRM. Only Wayland sink is supported in this configuration, KMS sink isn't supported. Changes made to the GSTDucati plugin is available in linux-kernel-addon/fs-patches/0001-gstducati-Enable-encode-decode-for-vDRM-setup.patch

If other video playback applications need support with VDRM, make sure buffers targeted for IPUMM are allocated from OMAPDRM and imported to VDRM before display.

5 Revision History

Version	Date	Revision History
1.0	30th June 2014	Initial Version
1.1	30th July 2014	Updated for vision sdk 2.03
1.2	17 th Dec 2014	Updated for 2.05
1.3	28th Feb 2015	Updated for 2.06
1.4	12th Oct 2015	Updated for 2.08
1.5	17th March 2016	Updated for 2.09
1.6	28th June 2016	Updated for 2.10
1.7	5th December 2016	Clean up and Minor updates
1.8	11th Jan 2017	Update for 4.4 kernel support
1.9	2nd February 2017	Updated for 2.12
2.0	30th March 2017	Updated for PSDK 3.2 migration
2.1	14th April 2017	Updated for 2.12.02
2.2	29th June 2017	Updates for 3.00.00
2.3	19th July 2017	Updates as per e2e comments For bug id ADASVISION-1608
2.4	12th Oct 2017	Updates for 3.01.00
2.5	13th Oct 2017	Updates for TDA2px For requirement id ADASVISION-1080
2.6	27th Nov 2017	Updates to fix ADASVISION-1080
2.7	20th Dec 2017	Updates for release 3.02.00
2.8	5th April 2018	Updates for release 3.03.00
2.9	21st June 2018	Updated with vDRM usecase details
3.0	25th September	Updated for early use-case and IPUMM decode ADASVISION-1955, ADASVISION-1971, ADASVISION-1957 & ADASVISION-1958
3.1	21st December 2018	Updated for Autosar Usecase

3.2	6th June 2019	Updates for release 3.07.00
3.3	17th July 2019	Updates for release 3.07.01
3.4	10th December 2019	Updates for release 3.08.00

«««§»»»