Vision SDK 1.3 User Guide

ABSTRACT:
The document describes main components of the Vision SDK. Firstly, the list of prerequisites is discussed along with necessary utility libraries build. In the second part, the examples of use on S32V234 are presented.

KEYWORDS:
Vision SDK, APEX, ISP
APPROVED:

Table of Contents

LIST OF TABLES

1 Introduction

In this document, the S32V234 Vision SDK is described. The SDK supports Standalone (OS less), Linux OS.

The first part of this document covers installation and build of required prerequisites. This includes necessary libraries and any collateral parts of the SW.

The second part then describes the ready to use examples which are part of this SDK. These examples aim to provide an understanding of the basic functionality, such as using OpenCV with the SDK.

1.1 Purpose

The purpose of this document is to present the S32V234 Vision SDK and help users to bring up examples quickly.

1.2 Audience Description

This document is intended to S32V234 Vision SDK users.

1.3 References

Id	***Title***	***Location***
[1]	*OpenCV library*	http://opencv.org/

Table 1 References Table

1.4 Definitions, Acronyms, and Abbreviations

Term/Acronym	***Description***
ACF	*APEX Core Framework*
APEX2	*Programmable massively parallel vision accelerator core.*
APU COMPILER	*Compiler for the APEX core. There are two supported compilers: NXP APU compiler (aka nxp) and APU TOOLS (aka tct).*
ARM	*Family of RISC architectures*
DCU	*Display Controller Unit*

FDMA	*Fast DMA HW block.*
GHS	*Greenhills*
ISP	*Image Signal Processor subsystem of the S32V234 SoC.*
MIPI CSI	*MIPI Alliance standard for Camera Serial Interface.*
OpenCL	*Open Computing Language*
OpenCV	*Open Source Computer Vision*
RTOS	*Real Time Operating System*
SDK	*System Development Kit*
Sequencer	*M0+ core and its firmware controlling the ISP subsystem.*
SRAM	*Static RAM.4MB memory area used for fast data buffering during ISP processing.*

Table 2 Acronyms Table

1.5 Document Location

This document is available in VisionSDK directory structure at the following location:
`s32v234_sdk/docs/vsdk/VisionSDK_UserGuide.pdf`

2 SDK Content

2.1 Content Description

The package was created in order to present the Vision SDK to the user. The vast majority of user applications are demos, which demonstrates a variety of possible use of the software development kit.

This publication is targeted mainly at the vision processing pipeline of S32V234, which is aimed for the programmable massively parallel vision accelerator core (APEX) and Image Signal Processing (ISP).

The publication does not aim to describe either way of programming, but to enable user for using provided demos and compile all necessary prerequisites.

The code in the sdk is provided as an out-of-the-box package. The user doesn't have to build all included components.

2.2 Documentation Description

In the Section 3, Building Applications in VSDK, the build system is presented. This section describes the build environment setup, necessary environment variables as well as the OpenCV build. In the section 4, all the 3rd party libraries are described along with their respective build commands (optional, most of the 3rd party libs are provided in pre-built form). In the section 5 describes the demo applications. The last section 6 describes the known problems and solutions to do.

2.3 Directory Structure

Note: the vSDK installer is targeted to Windows and Linux host development.

After full installation, this is the content of the installation directory:

- **compilers [dir]** – Compilers directory
- **jre [dir] –** Directory containing java software
- **msys32 [dir]** –MinGW + MSYS2 installation
- **s32v234_sdk [dir]** – the SDK itself
    - **3rdparty[dir]** – Directory containing 3rd party libraries
        - **boost_1_62_0 –** boost version 1.62.0 library
        - **eigen [dir]** – Eigen library (source code)
        - **ffmpeg [dir]** – FFmpeg library (build script)
        - **iniparser4 [dir] -** Iniparser 4
        - **ocv [dir]** – OpenCV library (build scripts and pre-compiled binaries)
        - **pthread** – pthread library
        - **tbb_lib [dir]** – Threading Building Blocks library for Windows (build script)
    - **build [dir]** – build toolchains used by the SDK
        - **cmake [dir]**
        - **mvc [dir]**
        - **nbuild [dir]**
    - **demos [dir]** – demo applications (chapter 4)
        - **apex [dir] –** APEX focused demos
        - **apexcv_base [dir] –** APEXCV Base demos
        - **apexcv_pro [dir] –** APEXCV Pro demos
        - **apps [dir] –** More complicated demo applications
        - **avb [dir] –** AVB focused demos
        - **data [dir] -** collects data files required by various demo applications at runtime
            - **apps [dir] –** data inputs to be compiled in to **apps** demos
            - **common [dir] –** data inputs to be compiled in to general demos
            - **outputs [dir]** – placeholder for future use

        - **isp [dir] –** ISP focused demos
        - **neon [dir] –** ARM NEON focused demos
        - **other [dir] –** Other, non-classified apps
    - **docs [dir]** – documentation of the SDK
        - **apex [dir]** – APEX related documentation
        - **doxygen [dir] –** doxygen related documention
        - **drivers [dir]** – Doxygen-generated HTML
        - **nbuild [dir] –** nbuild user guide
        - **vsdk [dir]** – Vision SDK general documentation
    - **include [dir] –** Headers of the SDK
    - **isp** – All ISP related runtime SW components
        - **firmware [dir]** – sequencer SW
        - **graphs [dir]** – ISP processing graphs
        - **image_data [dir]** – test images
        - **inc [dir]** – header files required for ISP sequencer
        - **kernels [dir]** – IPUx kernels
    - **kernels [dir]** – APEX processing kernels which may be used in user applications
        - **apu [dir]** – APEX kernels implemented in “Vector C” language
    - **libs [dir]** – SDK software libraries and drivers
        - **apex [dir]** – APEX libraries and drivers
        - **apexcv_base [dir]** – APEXCV Base library directory
        - **apexcv_pro [dir]** – APEXCV Pro library directory
        - **apexcv_ref [dir] –** apexcv references.
        - **dnn [dir] –** Deep neural network related
        - **io [dir]** – I/O related libs and drivers
        - **isp [dir]** – ISP related libs and drivers
        - **startup [dir]** – Standalone startup routines
        - **utils [dir]** – Other non-classified libs
        - **webserver [dir] –** socket IO related libs
    - **os [dir] -** Environment build directory (OS, u-boot). Includes pre-compiled SD card contents.
        - **debug [dir] –** debugger scripts (mostly Lauterbach)

        - **integrity [dir] –** INTEGRITY OS related MULTI projects (for kernel and modules)
        - **linux [dir] –** Linux os related
        - **uboot [dir] –** uboot related
    - **platform [dir]** – Platform/device specific header files
        - **fpga_vs4 [dir] -** The register and bit field definitions for s32v234 on fpga
        - **s32_v234 [dir] -** The register and bit field definitions for s32v234
    - **tools [dir]** – support tools
        - **emu [dir]** –emulation libraries
            - **acf [dir]** –ACF emulation library
            - **apu [dir]** – APEX emulation library
        - **gdt [dir] –** ISP Graph compiler
        - **isp [dir] –** IPU Kernel compiler
        - **sr2rom [dir] –** S-Records to SAMI ROM format converter

3 Building Applications in VSDK

This chapter describes the build environment of VSDK. Note that everything needed to run and build the demos is installed automatically by the VSDK installer.

Some of the applications need to download external 3rd party libraries. If this is needed, a script is available for automatic download of the specified library in 3rdparty directory.

3.1 Build Environment

Build of all applications is based on NBUILD Makefile system – all buildable parts of the VSDK contain build directories in the following format:

build-[target]-[compiler]-[platform]-[debug/optimized]

Invocation of the Makefile in specific directory will build the application along with all necessary prerequisites.

3.1.1Development tools

The applications for standalone, Linux OS can be cross-compiled from either Linux or Windows OS.

3.1.1.1Windows

Under Windows OS, MSYS2 bash is recommended along with GNU gcc/g++ compilers. The package is part of the Vision SDK and can be installed from the Vision SDK installer. MSYS2 will automatically set up itself into PATH environment variable. To starts MSYS2 bash in Vision SDK, please double click on ***runShell.cmd*** in ..\s32v234_sdk folder installed.

NOTE: it is recommended to use the MSYS2 installation coming with the Vision SDK installer to avoid any incompatibilities.

3.1.1.2Linux

Under Linux the development tools are installed automatically by the Vision SDK installer.

The user needs to add:

- <Base GCC Version>-Earmv7-eabi/i686-linux/bin/
- <Base GCC Version>-Earmv8-eabi/i686-linux/bin; and
- <Base GCC Version>-Xarmv8-linux/i686-linux/bin;

to the PATH variable.

For the specific compiler versions pls refer to the ReleaseNotes.

Setup an environment variables for APU compiler:

- export APU_TOOLS=/path/to/NXP/APU_Compiler_v1.0

NOTE: Make sure correct GNU GCC version on host machine (e.g. Ubuntu). Please see release notes for supported host compiler version. Only 32-bit version is provided for the host.

3.1.2Build for Linux or Standalone

If the application is to be built for Linux or for standalone, following steps are necessary:

1. cd to path/to/the/application/build-v234ce-gnu-[linux/sa]-[d/o]
2. execute “make allsub” (if application doesn’t includes APEX graphs, else use 3 or 4)
3. execute “make APU_COMP=nxp allsub” , plus APU_PRECOMP=1 to use NXP precompiled headers
4. execute “make APU_COMP=tct allsub” , plus APU_PRECOMP=1 to use TCT precompiled headers
5. The APU_COMP can be defined as an environment variable, so it’s not necessary to specify it in the make command.

For both target environment, an .elf file is created, which shall be executed on the target platform. In the case of standalone application, the .bin file is created too and can be loaded onto SD card in order to boot directly from the SD card.

3.2 APEX ACF Build

The ACF Graph is application specific and is **automatically rebuilt with the NBUILD system**. In order to have the ACF Graph built, an APU compiler is necessary, the NXP APU compiler (nxp) comes with Vision SDK installer, or have the APU Tools (tct) installed.

3.2.1Build for all platforms

Some demos contain a `graphs` directory, where ACF graph definitions are located. In this folder, `build-apu-tct-sa-d` and `build-apu-nxp-sa-d` directories are present each with a Makefile, the graphs folders also contain some header files and BUILD.mk file.

If a re-build is required, a simple invocation of "make allsub" in the application build directory, or in the graph build folder (for graph only rebuild) will update the target files invoking automatically the right compilers.

3.3 Kernel modules in Linux OS

To run an application using the S32V234 vision processing pipeline including camera I/F, ISP and APEX2 accelerators a number of kernel (*.ko) modules are required. Those are available pre-built in the vSDK (please refer to the chapter 2.3). In case re-building those kernel modules is required, please follow following instructions.

In Linux OS, the drivers need to be build and installed as a kernel modules into running OS. User can build and install the drivers using following steps:

3.3.1General Preparation

Export necessary environment variables:

- `export ARCH=arm`
  - Target architecture
- `export CROSS_COMPILE=aarch64-linux-gnu-`
  - Compiler prefix specification
- `export LINUX_S32V234_DIR=/linux/sources/directory/for/s32v234/`
  - optional, only for s32v234 Linux build
  - Path to the Linux sources (same kernel as is running on the board)
- `export PATH=/path/to/your/compiler/binaries:$PATH`
  - Path to the compiler binaries

3.3.2Building the Kernel Modules

Following kernel modules can be built:

- apex.ko: s32v234_sdk/libs/apex/drivers/kernel/build-v234ce-gnu-linux-d
- oal_cma.ko: s32v234_sdk/libs/utils/oal/kernel/build-v234ce-gnu-linux-d
- csi.ko: s32v234_sdk/libs/isp/csi/kernel/build-v234ce-gnu-linux-d
- cam.ko: s32v234_sdk/libs/isp/cam_generic/kernel/build-v234ce-gnu-linux-d
- seq.ko: s32v234_sdk/libs/isp/sequencer/kernel/build-v234ce-gnu-linux-d

- fdma.ko: s32v234_sdk/libs/isp/fdma/kernel/build-v234ce-gnu-linux-d
- h264enc.ko: s32v234_sdk/libs/isp/h264denc/kernel/build-v234ce-gnu-linux-d
- h264dcd.ko: s32v234_sdk/libs/isp/h264dec/kernel/build-v234ce-gnu-linux-d
- jpegdcd.ko: s32v234_sdk/libs/isp/jpegdec/kernel/build-v234ce-gnu-linux-d
- viulite.ko: s32v234_sdk/libs/isp/viu/kernel/build-v234ce-gnu-linux-d

1. **Building kernel modules**

```
cd <kernel_build_dir>
make
```

3.3.3Using the Kernel Modules

Prior to running VSDK demos the appropriate kernel modules must be inserted. Please use the following list of commands to insert the modules manually:

```
insmod /s32v234/apex.ko
insmod /s32v234/oal_cma.ko
insmod /s32v234/csi.ko
insmod /s32v234/cam.ko
insmod /s32v234/seq.ko
insmod /s32v234/fdma.ko
insmod /s32v234/h264enc.ko
insmod /s32v234/h264dcd.ko
insmod /s32v234/jpegdcd.ko
insmod /s32v234/viulite.ko
```

NOTE: While inserting the kernel modules it is important to arrange the insmod commands in an order (e.g. given above) which complies to several dependencies between the modules. Explicitly:

- oal_cma.ko has to be inserted prior to fdma.ko. FDMA driver relies on the ability to allocate portion of SRAM through the OAL module.
- csi.ko has to be inserted prior cam.ko (connection to “frame start/end” IRQ handling on CSI)
- If ISP functionality will be used the seq.ko has to be inserted prior to fdma.ko. There is a symbol dependency between the modules.

4 3rd Party Libraries

4.1 Eigen

Eigen is a high-level C++ library of template headers for linear algebra, matrix and vector operations, geometrical transformations, numerical solvers and related algorithms. Some of the demos need this library.

There is no necessary build action – the Eigen library is in the form of included headers.

4.2 FFMPEG

FFmpeg is the multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play vast majority of the multimedia formats. It supports the most obscure ancient formats up to the cutting edge.

The SDK release contains a script for supported platforms, which downloads and builds the FFMPEG library. If a customized build is required please follow the steps in subsection bellow.

4.2.1Build for Linux OS application

1. **Download ffmpeg version 2.8.5**
    a. download archive from https://www.ffmpeg.org/releases/ffmpeg-2.8.5.tar.bz2
    b. extract its contents to working directory (let's say ~/)

2. **Set following system variables:**
    a. `export CROSS_COMPILE=aarch64-linux-gnu-`
    b. `export PATH=/path/to/your/arm/cross/compiler/binaries:$PATH`

3. **Build ffmpeg libraries**
    a. `cd to SDK_ROOT/3rdparty/ffmpeg/build-v234ce-gnu-linux`
    b. `run bash build.sh ~/ffmpeg-2.8.5`
    (usage: first parameter specifies ffmpeg source directory)

4.3 iniParser 4

iniParser is a simple C library offering ini file parsing services. The library is pretty small (less than 1500 lines of C) and robust, and does not depend on any other external library to compile. It is written in ANSI C and should compile on most platforms without difficulty.

The VSDK contains full source code of the iniParser library + implemented NBUILD system for it. The iniParser is then built automatically when needed by the application.

4.4 OpenCV

OpenCV library provides the API for handling image I/O in examples and provides large, community developed library of vision processing algorithms.

The SDK release contains prebuild binaries for supported platforms, so there is no need for re-build of the OpenCV. If a customized build is required please follow the steps in subsection bellow. The OpenCV library build is optimized to be executed on 64bit Linux machine. In MSYS2 environment on Windows the build will fail because of Linux style paths being used with Windows compiler binaries. Please use Linux OS environment on host PC for building the OpenCV (the cross-compiled libraries are useable also for Windows cross-compilation – if the same compiler version is used).

4.4.1 Build for OS Less application

1. **Check the compiler options for OCV**

   Open corresponding `s32v234_sdk/build/cmake/toolchains/aarch64-sa-gnueabi-gcc.cmake` file and check following lines:

   ```
   SET(CMAKE_C_COMPILER   arm-compiler-executable-gcc)
   SET(CMAKE_CXX_COMPILER arm-compiler-executable-g++)
   ```

   The compilers must be visible by the script, so they must be placed in the PATH directory, or the absolute path must be provided in these commands.

2. **Download the OpenCV 3.1.0 sources**

   http://opencv.org/downloads.html

3. **Extract the sources to the s32v234_sdk/../opencv_src**

4. **Apply the Standalone patch**

   a. Switch into `/s32v234_sdk/../opencv_src` directory

   b. Apply available patch by executing:

   ```
   patch -p1 -i s32v234_sdk/3rdparty/ocv/standalone_3.1.0.patch
   ```

5. **Build the OpenCV**
    a. Open corresponding `/s32v234_sdk/3rdparty/ocv/cmake-v234ce-gnu-sa` directory
    b. `bash build.sh`

4.4.2Build for Linux OS application

1. **Check the compiler options for OCV**

    Open corresponding `s32v234_sdk/build/cmake/toolchains/aarch64-linux-gnu-gcc.cmake` file and check following lines:

    ```
    SET(CMAKE_C_COMPILER   arm-compiler-executable-gcc)
    SET(CMAKE_CXX_COMPILER arm-compiler-executable-g++)
    ```

    The compilers must be visible by the script, so they must be placed in the PATH directory, or the absolute path must be provided in these commands.

2. **Set the environment variable**
    a. For this step, the root file system must be available on the host machine. It's possible to do that by mounting the `uramdisk.image.gz` into host machine (see section 3.2.2. – 2.)
    b. For examples which are using OpenCV is recommended to use NFS mounted rootfs due to the size of the OpenCV libraries (ramdisk file is prepared for maximum of 16MB)
    c. `Export LINUX_ROOT_PATH=/path/to/crosscompile/rootfs/`

3. **Download the OpenCV 3.1.0 sources**

    http://opencv.org/downloads.html

4. **Extract the sources to the s32v234_sdk/../opencv_src**

5. **Build the OpenCV**
    a. Open corresponding `/s32v234_sdk/3rdparty/ocv/cmake-v234ce-gnu-linux` directory
    b. `bash build.sh`

4.5 TBB Library

Intel® Threading Building Blocks (Intel® TBB) enables user to easily write parallel C++ programs that take full advantage of multicore performance, that are portable and composable, and that have future-proof scalability.

The TBB library is used for EMU lib only (Windows MS Visual Studio build) and is available in the form of download/install script**.**

5 Examples and demos

The example applications and demos serve to demonstrate the APEX functionality on the S32V234 based boards. In this chapter, each demo is described along with its necessary prerequisites and run commands.

5.1 Demo Overview

Following table gives an overview of available demos and which platform (OS) is currently supported.

Focus	Name	Description	Bare Metal	Linux	Windows
APEX	apex_add	APEX adds two images: simplest APEX usage model which creates a graph calling the addition kernel and computing the addition in parallel on the APEX device. Can be used as a template project for new user defined projects	Yes	Yes	Yes
	apex_basic	APEX most basic application of using a custom graph for the APEX core. Adds two images with random content	Yes	Yes	Yes
	apex_threads	APEX short application showing how to use ACF processes in multi-threaded application	No	Yes	No
	apex_downsample_upsample_cv	APEX Downsample/Upsample:example on how to compute on APEX with image size change from input to output	Yes	Yes	No
	apex_emulation_test	APEX Emulation Test: is a collection of some of the APEX algorithms listed here (gauss filtering, FAST9, histogram computation, integral image, indirect inputs, general filtering), which demonstrates different ways of handling code	Yes	Yes	Yes

		organization for either simplicity or readability. It has also an attached VS project for Windows emulation.			
	apex_face_detection_cv	APEX Face Detection: complex algorithm which uses feature detection and classifiers on APEX to detect human heads in images.	Yes	Yes	Yes
	apex_fast9_cv	APEX Fast9 Corner Detection: FAST9 corner detection algorithm implementation on APEX	Yes	Yes	No
	apex_gauss5x5_cv	APEX Gauss 5x5 Filter: Simple filtering demo which exemplifies how to cope with spatial dependencies on APEX	Yes	Yes	No
	apex_histogram_cv	APEX Histogram Computing: computes in parallel the histogram of an image with demonstration of static data usage on APEX	Yes	Yes	No
	apex_indirect_input_cv	APEX Indirect Inputs: Shows how to randomly access image patches on APEX, with an example of image rotation	Yes	Yes	No
	apex_integral_image_cv	APEX Integral Image: computes the integral image (Summed Area Table – SAT) of an image, which is used an optimization for applying subsequent box filters and others.	Yes	Yes	No
	apex_isp_fast9	The demo shows combined use of ISP input, APEX processing and DCU output of the image.	Yes	Yes	No
	apex_isp_segmentation	The application demonstrates real time image segmentation using SLIC algorithm	No	Yes	No
	apex_roi_cv	Example on how to use the Region Of Interest feature of the APEX, by envisioning a certain region of an input	Yes	Yes	Yes

		image.			
APEXCV Base	apexcv_basic	The example of APEXCV Base functions. The application runs all supported APEXCV routines and shows the result.	Yes	Yes	Yes
	apexcv_camdisp	Application with camera input and display, using APEX-CV functions to invert a color channel depending on the intensity in the green channel.	Yes	Yes	Yes
	apexcv_simple	Simple apexcv demo, demonstrating the histogram computation with the ApexCV-Base library.	Yes	Yes	Yes
	apexcv_threads	short application showing how to use APEX-CV objects in multi-threaded application	No	Yes	No
APEXCV Pro	apexcv_camdisp_histequal	Image pyramid creation in APEXCV Pro.	Yes	Yes	Yes
	apexcv_orb	Demonstrates the ORB algorithm computed with the ApexCV-Pro library	Yes	Yes	No
	apexcv_remap	Apexcv demo, demonstrating remapping with the ApexCV-Pro library.	Yes	Yes	No
Full size applications	apex_isp_face_detection_cv	APEX Face Detection Algorithm: demonstrates the implementation of APEX driven face detection algorithm.	Yes	Yes	No
	apex_isp_ldw_cv	APEX Lane Departure Warning: demonstrates the implementation on APEX of the complex Lane Detection algorithm.	Yes	Yes	No
	cnn_classifier	Demonstrates a convolutional neural network classifier on APEX.	No	Yes	No
	feature_tracking	APEX Feature tracking demo using Lucas-Kanade optical flow.	No	Yes	No

	isp_stereo_apexbm	APEX Stereo disparity computation using two camera ISP input.	No	Yes	No
	isp_stereo_calib	Camera calibration utility for APEX Stereo BM demo.	No	Yes	No
	lane_departure_warning	APEX Lane Departure Warning.	No	Yes	No
	pedestrian_detection	APEX Pedestrian detection application. based on HOG algorithm	No	Yes	No
	pedestrian_detection_aggcf	Demonstrates pedestrian detection with AGGCF features on APEX with camera input.	No	Yes	No
AVB	avb_isp_h264_1stream	Playback of four h.264 AVB video streams with NAL format and I-frames only	No	Yes	No
	avb_isp_h264_4stream	Playback of four h.264 AVB video streams with NAL format and I-frames only	No	Yes	No
	avb_isp_jpeg_4stream	Playback of jpeg AVB video streams	No	Yes	No
ISP	isp_jpeg_4stream	Decodes a JPEG image on all 4 stream inputs of the JPEG HW decoder in conjunction with ISP	No	Yes	No
	isp_h264dec_single_stream	Decodes a H264 static image on using the h264 HW decoder in conjunction with ISP	No	Yes	No
	isp_ov10635_viu_dcu	Demo to show the usage of the VIU (parallel) camera interface	No	Yes	No
	isp_ov10635_quad	Demo to use the MAXIM deserializer with 4 OV10635 cameras	Yes	Yes	no
	isp_ov10640_default	Demo operating the ov10640 MipiCsi camera in HDR mode	Yes	Yes	No
	isp_ov10640_quad	Demo to use the MAXIM deserializer with 4 OV10640 cameras	Yes	Yes	No
	isp_ov9716_mipi_s	Demonstrates MIPI interface and H264 encoding with camera OmniVision_9716.	No	Yes	No

	isp_sonyimx224_csi_dcu	ISP pass-through demo: shows how to grab camera data using ISP and show them on the internal display.	Yes	Yes	No
	isp_sony_imx224_csi_dcu_cam_switch	Camera and graph switching demo: Alternating two graphs execution	No	Yes	No
	isp_sonyimx224_default	Demo operating the Sony camera in dual exposure mode	Yes	Yes	No
	isp_sonyimx224_h264enc	Encodes live video from Sony imx224 using the H264 Encoder HW decoder in conjunction with ISP	No	Yes	No
	isp_sonyimx224_rgb_yuv_gs8	Demo template to demonstrate the generation of RGB, YUV422 and Greyscale8 images from the camera input in parallel using the ISP	Yes	Yes	No
	isp_static_simple	Demo static_simple -> dcu demo	Yes	Yes	No
NEON	neon_eigen_add	Eigen library addition: Presents simple matrix addition which uses Eigen library accelerated by ARM NEON	Yes	Yes	No
	neon_gauss3x3_cv	NEON Gaussian filter: Demonstrates simple ARM NEON SIMD filtering kernel.	Yes	Yes	No
OTHER	h264_encoder_cv	The application demonstrates H264 Encoder driver usage. In the application H264 Encoder HW and SRAM input/output buffers configurations are provided. Input frames are captured from video sequence via OpenCV and encoding is realized. As the output, encoded sequence is saved into the file.	No	Yes	No
	hello	Hello World example how to build a simple ARM only program	Yes	Yes	No
	multi_thread	Simple demo showing OAL Task and OAL Semaphore functionality.	Yes	Yes	No

	web_server	Application which allows communication with EVB using web socket. Demo demonstrate how to show runtime results or how to configure demo immediately using webpage (javascript).	No	Yes	No

5.2 Building the demos

5.2.1ACF Graph Build (optional)

If APEX ACF Graphs need to be rebuilt (all demos have pre-build header files already), following steps will build the ACF graph again. Please note the Synopsys IP Programmer needs to be installed and binaries put into PATH prior to build:

1. **Change to the application directory**
   a. `cd s32v234_sdk/demos/<name_of_demo>/graphs/ build-apu-tct-sa-d`
2. **Build the ACF Graph**
   a. Issue `make allsub` command

5.2.2OS Less Makefile Build

For Standalone application, the VSDK build framework is available to ease up the build process. For any board, following syntax will build the application:

1. **Change to the application directory**
   a. `cd s32v234_sdk/demos/<name_of_demo>/build-v234ce-gnu-sa-d`
2. **Build the ARM application**
   a. Issue `make` command
   b. If `make allsub` command is used, all prerequisites are rebuilt by force.
   c. If APEX is used, please follow 5.2.1 if ACF graph needs to be rebuilt.

5.2.3Linux OS Makefile Build

On Linux OS, the Linaro build framework is available to ease up the build process. For any board, using following syntax will build the application for Linux OS:

1. **Change to the application directory**
   a. `cd s32v234_sdk/demos/<name_of_demo>/ build-v234ce-gnu-linux-d`

2. **Build the ARM application**
    a. Issue `make` command
    b. If `make allsub` command is used, all prerequisites are rebuilt by force.
    c. If APEX is used, please follow 5.2.1 if ACF graph needs to be rebuilt.

5.3 APEX Related Examples

5.3.1APEX Add Two Images

The add example is prepared to demonstrate one-kernel APEX graph in use. The application generates two random grayscale 8-bit images (matrices) and adds them using APEX HW and SW algorithm. The parallel use of both APEX cores is also demonstrated, because the program adds two images on both cores simultaneously creating two outputs. The resulting two 16-bit grayscale images are compared and result is printed to the user.

Name of the application directory: **apex/apex_add**

Input data required: No

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone

5.3.1.1Prerequisites

5.3.1.1.1 OS Less application

There are no specific requirements to run this application.

5.3.1.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

5.3.1.2Running the example

5.3.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_add.cmm script.

OR

Load the apex_add.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_add.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.1.2.2 Linux OS

Application starts as follows:

```
./apex_add.elf
```

5.3.1.2.3 Windows OS

Please load the Visual Studio Project from ./build-deskwin32/mvc/add_project.sln, compile it. Set as Startup project the apex_add project where the main function is to be found. Thereafter run it.

5.3.2APEX basic

This example is prepared to demonstrate one-kernel APEX graph in use. The application generates two random grayscale 8-bit images (matrices) and adds them using APEX HW and SW algorithm.

Name of the application directory: **apex/apex_basic**

Input data required: No

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone

5.3.2.1Prerequisites

5.3.2.1.1 OS Less application

There are no specific requirements to run this application.

5.3.2.1.2 Linux OS

The oal_cma.ko apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

5.3.2.2Running the example

5.3.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_add.cmm script.

OR

Load the apex_add.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image

- On Linux system, use dd:

```
sudo dd if=apex_basic.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.2.2.2 Linux OS

Application starts as follows:

```
./apex_basic.elf
```

5.3.2.2.3 Windows OS

Please load the Visual Studio Project from ./build-deskwin32/mvc/apex_basic_project.sln, compile it. Set as Startup project the apex_add project where the main function is to be found. Thereafter run it.

5.3.3APEX Downsample/Upsample cv

The application demonstrates data rate change when the element k is different for the connected ports. The demo loads an input image and performs a linear downsampling and upsampling of the image. The output consists of two images, one of them is an enlarged version of the input, the other one is a shrinked version displayed on S32V234 EVB display.

Name of the application directory: **apex/apex_downsample_upsample_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.3.1 Prerequisites

5.3.3.1.1 OS Less application

There are no specific requirements to run this application.

Data input required: data/common/headers/in_grey_256x256.h

Copy demos/data folder to C:/vsdk

5.3.3.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Input data required: data/common/ in_grey_256x256.png

Copy demos/data folder to the Linux home directory.

5.3.3.2Running the example

5.3.3.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_downsample_upsample_cv.cmm script.

OR

Load the apex_downsample_upsample_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_downsample_upsample_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.3.2.2 Linux OS

Application starts as follows:

```
./apex_downsample_upsample_cv.elf
```

Example outputs of the downsample/upsample demo

5.3.4APEX Emulation Test

The application demonstrates the use of emulation library. It works as test of several subsequent ACF graphs defined in the application, however if built for host desktop machine, the application uses emulation library. If built for ZC702, an ACF implementation is used. By this approach, it's possible to demonstrate use of the kernels in both emulated and real environment.

Name of the application directory: **apex/apex_emulation_test**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone

5.3.4.1Prerequisites

5.3.4.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required:

data/common/headers/in_grey_256x256.h

data/common/headers/in_color_256x256.h

Copy demos/data folder to C:/vsdk

5.3.4.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Inputs data required:

data/common/in_grey_256x256.png

data/common/in_grey_256x256.png

Copu demos/data folder to the Linux home directory.

5.3.4.2Running the example

5.3.4.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_emulation_test.cmm script.

OR

Load the apex_emulation_test.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_emulation_test.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.4.2.2 Linux OS

Application starts as follows:

```
./apex_emulation_test.elf
```

- Do it by executing: `export LD_LIBRARY_PATH=/mnt/lib`

5.3.4.2.3 Windows OS

Please load into Visual Studio, the project from ./build-deskwin32/mvc/apex_emulation_test.sln, compile it. Set as Startup project the apex_emulation_test project where the main function is to be found. Thereafter run it.

5.3.5APEX Face Detection cv

This application demonstrates the use of APEX LBP classifier. The demo loads an RGB input image, transforms it to grayscale, executes the classifier, marks the faces as green rectangles and outputs the resulting RGB image. The classifier is implemented both in ACF and on the host processor. The applications outputs 3 images: faces found by APEX, faces found by the OpenCV library and faces found by a custom host CPU fixed-point implementation.

Name of the application directory: **apex/apex_face_detection_cv**

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone

5.3.5.1Prerequisites

5.3.5.1.1 OS Less application

There are no specific requirements to run this application.
Inputs data:
data/common/herders/input_faces.h
data/common/header/lbpcascade_frontalface.h
Copy data to C:/vsdk

5.3.5.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Inputs data required:

data/common/input_faces.png

data/common/ lbpcascade_frontalface.xml

data/common/ lbpcascade_frontalface_cv3.1.0.xml

Copy the demos/data folder to the Linux home directory.

5.3.5.2Running the example

5.3.5.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_face_detection_cv_test.cmm script.

OR

Load the apex_face_detection_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_face_detection_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

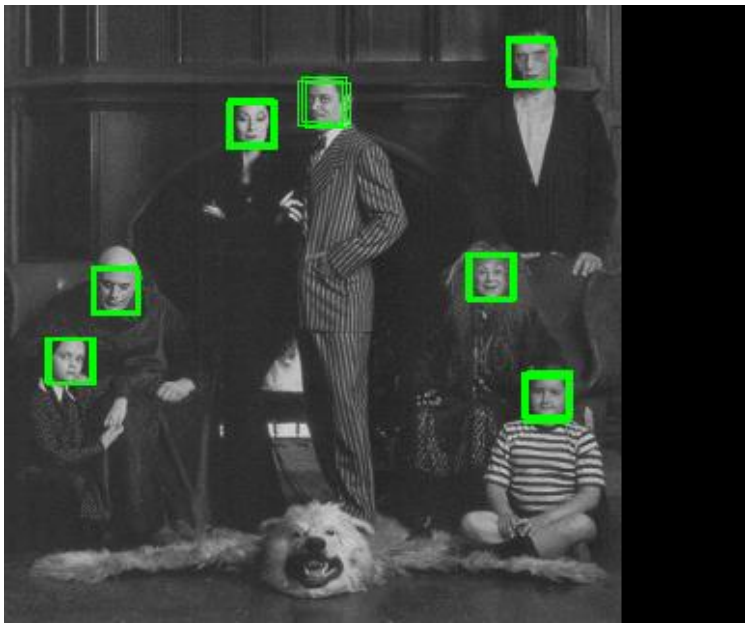
5.3.5.2.2 Linux OS

Application starts as follows:

```
./apex_face_detection_cv.elf
```

5.3.5.2.3 Windows OS

Please load into Visual Studio, the project from ./build-deskwin32/mvc/apex_face_detection.sln, compile it. Set as Startup project the apex_face_detection project where the main function is to be found. Thereafter run it.



Example output of the LBP face detection algorithm

5.3.6 APEX FAST9 cv Corner Detector

This application demonstrates the use of multi-channel ACF input. The demo loads the RGB input image, transforms it to grayscale, executes the FAST9 algorithm, marks the corners in green output channel and outputs the resulting RGB image (EVB Display). All the phases are implemented in ACF.

Name of the application directory: **apex/apex_fast9_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.6.1 Prerequisites

5.3.6.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: data/common/headers/in_color_256x256.h

Copy domes/data to C:/vsdk

5.3.6.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Inputs data require: data/common/in_color_256x256.png

Copy the demos/data folder to the Linux home directory.

5.3.6.2Running the example

5.3.6.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_fast9_cv.cmm script.

OR

Load the apex_fast9_cv.bin to the SD card and run the board:

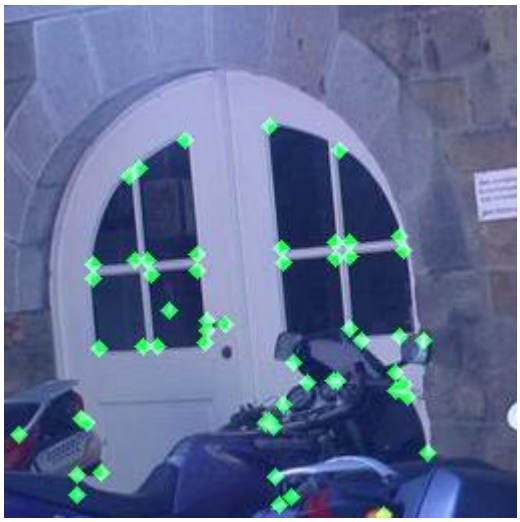
- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_fast9_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.6.2.2 Linux OS

Application starts as follows:

```
./apex_fast9_cv.elf
```



Example output of the FAST9 RGB algorithm

5.3.7APEX Gauss 5x5 Filter

The application demonstrates another simple one-kernel algorithm on the input image. The demo loads an image and executes the 3x3 Gaussian blur APEX kernel on it. The output is the blurred image displayed on the EVB.

Name of the application directory: **apex/apex_gauss5x5_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.7.1Prerequisites

5.3.7.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: data/common/headers/in_grey_256x256.h

Copy demos/data to C:/vsdk

5.3.7.1.2 Linux OS

The oal_cma.ko kernel module must be loaded prior to application launch (already loaded in provided OS). Do it by executing: `insmod oal_cma.ko`

OpenCV shared libraries must be present in the root file system.

Inputs data required: data/common/in_grey_256x256.png

Copy the demos/data folder to the Linux home directory.

5.3.7.2Running the example

5.3.7.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_gauss5x5_cv.cmm script.

OR

Load the apex_gauss5x5_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_gauss5x5_cv.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

5.3.7.2.2 Linux OS

Application starts as follows:

```
./apex_gauss5x5_cv.elf
```

5.3.8APEX Histogram Computation

The application demonstrates a non-image ACF output. The demo loads an input image, computes its histogram and compares it to the reference histogram. It outputs the histogram as an image.

Name of the application directory: **apex/apex_histogram_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.8.1Prerequisites

5.3.8.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: data/common/header/in_grey_256x256.h

Copy demos/data to C:/vsdk

5.3.8.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Inputs data required: data/common/in_grey_256x256.png

Copy the demos/data folder to the Linux home directory.

5.3.8.2Running the example

5.3.8.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_histogram_cv.cmm script.

OR

Load the apex_histogram_cv.bin to the SD card and run the board:

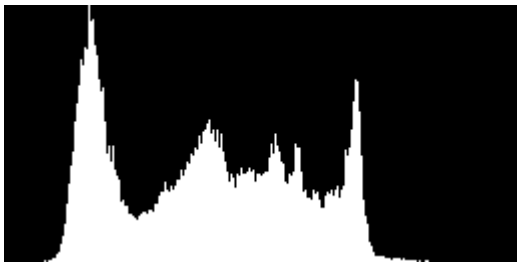
- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_histogram_cv.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

5.3.8.2.2 Linux OS

Application starts as follows:

```
./apex_histogram_cv.elf
```



Example output of the histogram demo

5.3.9 APEX Indirect Inputs Example

The application demonstrates indirect ACF inputs. The demo loads an input image and performs a 180-degree rotation of the image. It outputs the rotated image displayed on EVB.

Name of the application directory: **apex/apex_indirect_input_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.9.1 Prerequisites

5.3.9.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: data/common/header/in_grey_256x256.h

Copy demos/data to C:/vsdk

5.3.9.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Inputs data required: data/common/in_grey_256x256.png

Copy the demos/data folder to the Linux home directory.

5.3.9.2 Running the example

5.3.9.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_indirect_input_cv.cmm script.

OR

Load the apex_indirect_input_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_indirect_input_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.9.2.2 Linux OS

Application starts as follows:

```
./apex_indirect_input_cv.elf
```

Example output of the rotate 180 demo

5.3.10 APEX Integral Image

The application demonstrates static data usage. The demo loads an input image, computes the integral image (i.e. sat = summed area table) and performs box filtering using this table. The output consists of two images, namely the summed area table and the final filtered result image displayed on EVB.

Name of the application directory: **apex/apex_integral_image_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.10.1 Prerequisites

5.3.10.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: data/common/header/in_grey_256x256.h

Copy the demos/data folder to C:/vsdk

5.3.10.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Inputs data required: data/common/in_grey_256x256.png

Copy the demos/data folder to the Linux home directory.

5.3.10.2 Running the example

5.3.10.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_integral_image_cv.cmm script.

OR

Load the apex_integral_image.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_integral_image_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.10.2.2 Linux OS

Application starts as follows:

```
./apex_integral_image_cv.elf
```

Example outputs of the SAT demo

5.3.11 APEX ISP FAST9

The application demonstrates an easy algorithm on APEX – FAST9 corner detector combined with camera input and DCU output. The image is grabbed via SDI library, processed by APEX FAST9 kernel and output on display with highlighted corners.

Name of the application directory: **apex/apex_isp_fast9_cv**

This demo use sony imx224 master camera is connected to CSI0 interface.

Inputs data required: No

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.11.1 Prerequisites

5.3.11.1.1 OS Less application

There are no specific requirements to run this application.

5.3.11.1.2 Linux OS

The apex.ko oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod <moduel_name>.ko
```

OpenCV shared libraries must be present in the root file system.

5.3.11.2 Running the example

5.3.11.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_isp_fast9.cmm script.

OR

Load the apex_isp_fast9.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_isp_fast9.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.11.2.2 Linux OS

Application starts as follows:

```
./apex_isp_fast9.elf
```

5.3.12 APEX Segmentation

The application demonstrates real time image segmentation using SLIC algorithm. Detailed description is available at http://ivrl.epfl.ch/research/superpixels. The algorithm was slightly modified to meet real time requirements and to fit hardware constraints. The input of algorithm use grayscale image instead of CEILAB color space and local k-means algorithm use just one iteration. The algorithm still gives satisfactory segmentation results. The demo uses as an input 720p video from camera and it produce 64 x 36 superpixels. Segmentation and drawing the result is performed by the APEX-2. The result is displayed on EVB.

Name of the application directory: **apex/apex_isp_segmentation**

This demo use sony imx224 master camera is connected to CSI0 interface.

Inputs data required: No

Available builds:

- S32V234 Linux

5.3.12.1 Prerequisites

5.3.12.1.1 Linux OS

The apex.ko, oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.3.12.2 Running the example

5.3.12.2.1 Linux OS

Application starts as follows:

```
./apex_isp_segmentation.elf
```

Example output of the Segmentation algorithm

5.3.13 APEX ROI

The application demonstrates another simple one-kernel algorithm on the input image, but this version is aimed to demonstrate the Region of Interest settings in the image. The demo loads an image and executes in parallel two 5x5 Gaussian blur APEX kernel on it. The parallel implementation is due to demonstration of two ROI use cases. The output is the blurred image displayed on EVB.

If ROI needs to be specified, there are two possibilities to do so:

- Specify the ROI in input images
    - The DataDescriptor::SetROI(…) function sets the ROI in the image. This function does not alter any of image size indicators, just sets the ROI definition. When the image is used in ConnectIO(…) function, only the ROI part is passed into graph.
    - If the user wants to get a local pointer to the ROI region, a DataDescriptor instance can be passed as a parameter for another DataDescriptor constructor. When done so, a new DataDescriptor have pointers, width, height etc. set accordingly.
    - Warning, the SetFreeOnExit() must be used only once for all derived structures, the data are common.
- Specify the ROI when attaching the images into graph
    - It's also possible to attach the image into process by ConnectIO_ROI() function, where user specifies the region only when attaching the image. The DataDescriptor ROI specification is intact.

The demo is thoroughly commented to present both solutions.

Name of the application directory: **apex/apex_roi_cv**

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.3.13.1 Prerequisites

5.3.13.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: data/common/headers/in_grey_256x256.h

Copy the demos/data folder to C:/vsdk

5.3.13.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

Inputs data required: data/common/in_grey_256x256.png

Copy the demos/data folder to the Linux home directory.

5.3.13.2 Running the example

5.3.13.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_roi_cv.cmm script.

OR

Load the apex_roi_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_roi_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.3.13.2.2 Linux OS

Application starts as follows:

```
./apex_roi_cv.elf
```

5.4 APEXCV Base Related Examples

5.4.1ApexCV Basic Demo

The demo sequesntially runs the APEXCV Base functions. Please see the APEXCV Base documentation for the info about started functions.

Name of the application directory: **apexcv_base/apexcv_basic**

Input data required: No

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.4.1.1Prerequisites

5.4.1.1.1 OS Less application

There are no specific requirements to run this application.

5.4.1.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.4.1.2Running the example

5.4.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apexcv.cmm script.

OR

Load the apexcv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apexcv_basic.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.4.1.2.2 Linux OS

Application starts as follows:

```
./apexcv_basic.elf
```

5.4.2APEX Camdisp

The application demonstrates a non-image ACF output. The demo loads an input image, computes its histogram and compares it to the reference histogram. It outputs the histogram as an image.

Name of the application directory: **apexcv_base/apexcv_camdisp**

This demo use sony imx224 master camera is connected to CSI0 interface.

Inputs data required: No

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone

5.4.2.1Prerequisites

5.4.2.1.1 OS Less application

There are no specific requirements to run this application.

5.4.2.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.4.2.2Running the example

5.4.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apexcv_camdisp.cmm script.

OR

Load the apexcv_camdisp.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apexcv_camdisp.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.4.2.2.2 Linux OS

Application starts as follows:

```
./apexcv_camdisp.elf
```

5.4.2.2.3 Windows OS

Please load the Visual Studio Project from build-deskwin32\mvc\ apexcv_camdisp_project.sln, compile it. Set as Startup project the apexcv_camdisp project where the main function is to be found. Therefore run it.

5.4.3APEX Simple

Simple apexcv demo, demonstrating the histogram computation with the ApexCV-Base library.

Name of the application directory: **apexcv_base/apexcv_simple**

Inputs data required: No

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone

5.4.3.1Prerequisites

5.4.3.1.1 OS Less application

There are no specific requirements to run this application.

5.4.3.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS). Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.4.3.2Running the example

5.4.3.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apexcv_simple.cmm script.

OR

Load the apexcv_simple.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apexcv_simple.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.4.3.2.2 Linux OS

Application starts as follows:

```
./apexcv_simple.elf
```

5.4.3.2.3 Windows OS

Please load the Visual Studio Project from build-deskwin32\mvc\ apexcv_simple_project.sln, compile it. Set as Startup project the apexcv_simple project where the main function is to be found. Therefore run it.

5.5 APEXCV Pro Related Examples

5.5.1APEXCV Pro Camdisp histequal

The demo runs the APEXCV Pro functions. Please see the APEXCV Pro documentation for the info about started functions.

Name of the application directory: **apexcv_pro/apexcv_camdisp_histequal**

This demo use sony imx224 master camera is connected to CSI0 interface.

Inputs data required: No

Available builds:

- MS VS Windows build for APEX Emulation
- S32V234 Linux
- S32V234 Standalone

5.5.1.1Prerequisites

5.5.1.1.1 OS Less application

There are no specific requirements to run this application.

5.5.1.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.5.1.2Running the example

5.5.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apexcv_camdisp_histequal.cmm script.

OR

Load the apexcv_camdisp_histequal.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apexcv_camdisp_histequal.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

5.5.1.2.2 Linux OS

Application starts as follows:

```
./ apexcv_camdisp_histequal.elf
```

5.5.1.2.3 Windows OS

Please load the Visual Studio Project from build-deskwin32\mvc\ apexcv_camdisp_histequal_project.sln, compile it. Set as Startup project the apexcv_camdisp_histequal project where the main function is to be found. Therefore run it.

5.5.2APEXCV Pro ORB

The demo runs the APEXCV Pro functions. Please see the APEXCV Pro documentation for the info about started functions.

Name of the application directory: **apexcv_pro/apexcv_orb**

Inputs data required: No

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.5.2.1Prerequisites

5.5.2.1.1 OS Less application

There are no specific requirements to run this application.

5.5.2.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.5.2.2Running the example

5.5.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apexcv_orb.cmm script.

OR

Load the apexcv_orb.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apexcv_orb.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.5.2.2.2 Linux OS

Application starts as follows:

```
./ apexcv_orb.elf
```

5.5.3APEXCV Pro Remap

The demo runs the APEXCV Pro functions. Please see the APEXCV Pro documentation for the info about started functions.

Name of the application directory: **apexcv_pro/apexcv_remap**

Inputs data required: No

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.5.3.1Prerequisites

5.5.3.1.1 OS Less application

There are no specific requirements to run this application.

5.5.3.1.2 Linux OS

The oal_cma.ko and apex.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.5.3.2Running the example

5.5.3.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apexcv_remap.cmm script.

OR

Load the apexcv_remap.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apexcv_remap.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.5.3.2.2 Linux OS

Application starts as follows:

```
./apexcv_remap.elf
```

5.6 Full Size Applications

5.6.1APEX Face Detection Demo

This application implements a face detection demo with camera input and display output used for demonstration purposes. It reads the input from camera and produces the output displayed on EVB with detected faces using LBP search on APEX. The result is visually depicted.

Name of the application directory: **apps/pex_isp_face_detection_cv**

This demo use sony imx224 master camera is connected to CSI0 interface.

Inputs data required: No

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.6.1.1Prerequisites

5.6.1.1.1 OS Less application

- There are no specific requirements to run this application.

5.6.1.1.2 Linux OS

The apex.ko oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod apex.ko
```

OpenCV shared libraries must be present in the root file system.

5.6.1.2Running the example

5.6.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_isp_face_detection_cv.cmm script.

OR

Load the apex_isp_face_detection_cv.bin to the SD card and run the board:

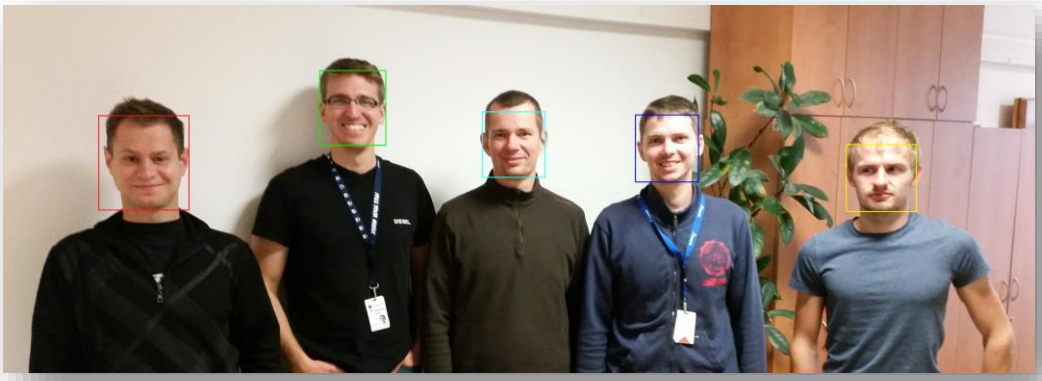
- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=apex_isp_face_detection_cv.bin of=/dev/sdb bs=512
seek=0 conv=fsync
```

5.6.1.2.2 Linux OS

Application starts as follows:

```
./apex_isp_face_detection_cv.elf
```



Example output of the Face Detection algorithm

5.6.2APEX ISP Lane Departure Warning System

This application implements a lane departure warning system. It reads the input from camera and produces the output displayed on EVB with detected lane using inner contour search on APEX. The result is visually depicted.

Name of the application directory: **apps/apex_isp_ldw_cv**

This demo use sony imx224 master camera is connected to CSI0 interface.

Inputs data required: No

Available builds:

- S32V234 Linux
- S32V234 Standalone

5.6.2.1Prerequisites

5.6.2.1.1 OS Less application

There are no specific requirements to run this application.

5.6.2.1.2 Linux OS

The apex.ko, oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod apex.ko
```

5.6.2.2Running the example

5.6.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading apex_isp_ldw_cv.cmm script.

OR

Load the apex_isp_ldw_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:
- `sudo dd if= apex_isp_ldw_cv.bin of=/dev/sdb bs=512 seek=0 conv=fsync`

5.6.2.2.2 Linux OS

Application starts as follows:

```
./apex_isp_ldw_cv.elf
```

Example output of the Lane Departure Warning algorithm

Linux also allows webserver visualization for this demo. For webserver service run demo as follows:

```
./apex_isp_ldw_cv.elf <IP address of EVB> 7777
```

Than just open page stored in demo root folder **ldw.html.** Supported browser is Mozilla Firefox. Visualization looks like following image. More details about webserver service follow **/docs/vsdk/ Webserver_User_Guide.pdf.**

5.6.3CNN Classifier

Demonstrates a convolutional neural network classifer on APEX.

Name of the application directory: **apps/cnn_classifier**

Available builds:

- S32V234 Linux

5.6.3.1Prerequisites

5.6.3.1.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

Inputs data required: data/apps/cnn_classifier folder

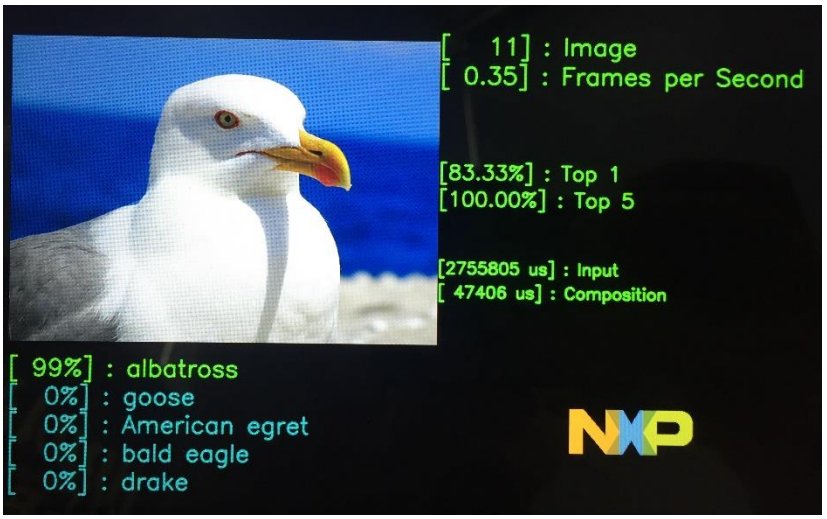
Copy the demos/data folder to Linux home directory.

5.6.3.2Running the example

5.6.3.2.1 Linux OS

Application starts as follows:

```
./gdc_cnn.elf
```



Example output of the CNN Classifier demos

5.6.4APEX Feature Tracking

This application implements a feature tracking system. It reads the input from camera and produces the output displayed on EVB with detected Lucas Kanade optical flow. The result is visually depicted.

Name of the application directory: **apps/feature_tracking**

This demo use sony imx224 master camera is connected to CSI0 interface.

Available builds:

- S32V234 Linux

5.6.4.1Prerequisites

5.6.4.1.1 Linux OS

The apex.ko, oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod apex.ko
```

Inputs data required: data/apps/feature_tracking folder

Copy the demos/data folder to the Linux home directory.

5.6.4.2Running the example

5.6.4.2.1 Linux OS

Application starts as follows:

```
./gdc_ft.elf
```

5.6.5APEX Block Matching Disparity Computation

This application implements a BM disparity computation from stereo image. It reads the input from cameras and produces the output displayed on EVB with color coded disparity. The camera calibration utility is also available to calibrate the image for depth map creation.

Name of the application directory: **apps/isp_stereo_apexbm**

apps/isp_stereo_calib

This demos used a sony imx224 master camera is connected to CSI0 interface and a sony imx224 slave camera is connected to CSI1 interface.

Available builds:

- S32V234 Linux

5.6.5.1Prerequisites

5.6.5.1.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The stereo_calibration.yml file (output of calibration) is needed to successful run of application.

5.6.5.2Calibrating the camera

5.6.5.2.1 Linux OS

Start the calibration application using: `/isp_stereo_apexbm.elf`

Inputs data required: data/apps/isp_stereo_apexbm folder

Copy the demos/data folder to the Linux home directory.

5.6.5.3Running the example

5.6.5.3.1 Linux OS

Application starts as follows:

```
./isp_stereo_apexbm.elf
```

5.6.6APEX Lane Departure Warning

This application implements a lane departure warning system. It reads the input from file and produces the output displayed on EVB with detected lane using APEX. The result is visually depicted

Name of the application directory: **apps/lane_departure_warning**

Available builds:

- S32V234 Linux

5.6.6.1Prerequisites

5.6.6.1.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

Inputs data required: all files under data/apps/lane_departure_warning
Copy the demos/data to the Linux home directory.

5.6.6.2Running the example

5.6.6.2.1 Linux OS

Application starts as follows:

```
./gdc_ldw.elf
```

Example output of the lane departure warning apps

5.6.7 APEX Pedestrian Detection

This application implements a pedestrian detection. It reads the input from camera and produces the output displayed on EVB with detected persons using optical flow. The result is visually depicted.

Name of the application directory: **apps/gdc_pedestrian_detection**

Available builds:

- S32V234 Linux

5.6.7.1 Prerequisites

5.6.7.1.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

Inputs data required: all file under data/apps/pedestrian_detection

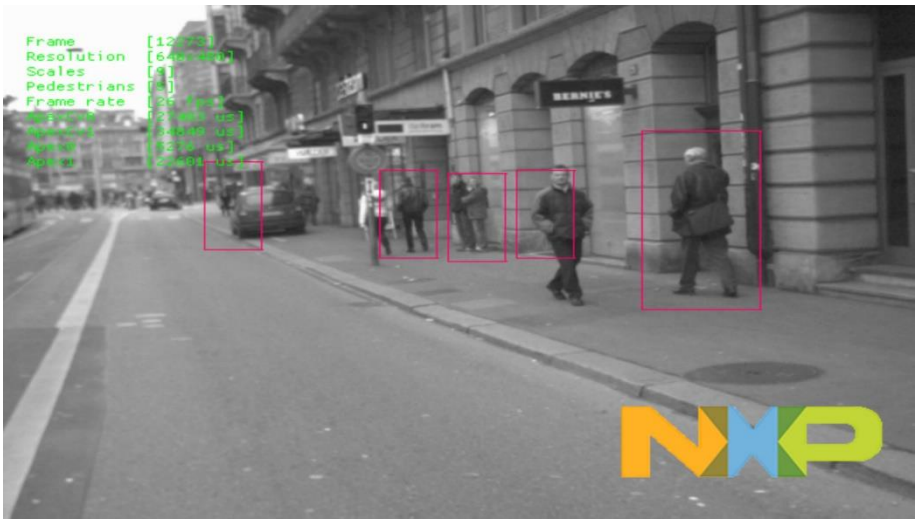
Copy the demos/data to the Linux home directory.

5.6.7.2 Running the example

5.6.7.2.1 Linux OS

Application starts as follows:

```
./gdc_pd.elf
```



Example output of the pedestrian detection apps

5.6.8APEX Pedestrian Detection AGGCF

Demonstrates pedestrian detection with AGGCF features on APEX with camera input.

Name of the application directory: **apps/gdc_pedestrian_detection_aggcf**

Available builds:

- S32V234 Linux

5.6.8.1Prerequisites

5.6.8.1.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:
```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

Inputs data required: all files under data/apps/pedestrian_detection_aggcf

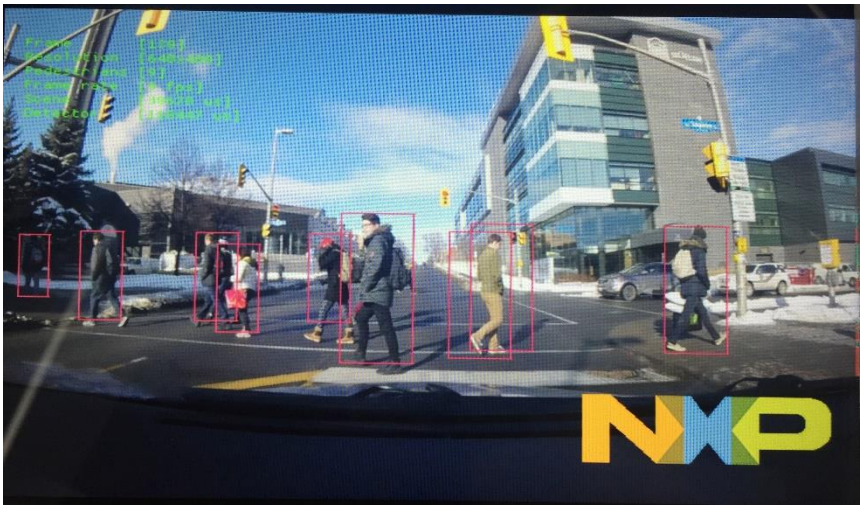
Copy the demos/data to the Linux home directory.

5.6.8.2Running the example

5.6.8.2.1 Linux OS

Application starts as follows:

```
./pd_aggcf.elf
```



Example output of the pedestrian detection aggcf apps

5.7 ISP Related Examples

All ISP related demos expect the s32v234 SoC security fuse to be disabled as described in s32V234-EVB_SetupGuide document. Otherwise the current mechanism for KRAM content setup will not work and execution of ISP demos will fail.

5.7.1ISP H264dec single stream

This demo is decoding an H264 encoded image from a file using the hardware H264 decoder together with ISP.

Name of the application directory: **isp/isp_h264dec_single_stream**

Available builds:

- S32V234 Linux

5.7.1.1Prerequisites

Inputs data required: data/common/img_1280x960.h264

Copy the demos/data to the Linux home directory.

5.7.1.2Running the example

5.7.1.2.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko, fdma.ko and h264dcd.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod h264dcd.ko
```

The application is started by typing `./isp_h264dec_single_stream.elf`

5.7.2ISP JPEG 4stream

This demo is decoding a JPEG image from a file on all 4 streams of the hardware JPEG decoder together with ISP.

Name of the application directory: **isp/isp_jpeg_4stream**

Available builds:

- S32V234 Linux

5.7.2.1Prerequisites

Inputs data required: data/common/jpg_dcd_test.jpg

Copy the demos/data folder to Linux home directory.

5.7.2.2Running the example

5.7.2.2.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko, fdma.ko and jpegdcd.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod jpegdcd.ko
```

The application is started by typing `./isp_jpeg_4stream.elf`

5.7.3ISP ov10635 Quad camera demo

This demo will show how 4 Omnivision 10635 cameras can be used with the ISP. The cameras are connected through a MAXIM deserializer.

Name of the application directory: **isp/isp_ov10635_quad**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.7.3.1Prerequisites

MAXIM deserializer attached to Mipi CSI0 interface. 4 Omnivision 10635 cameras attached to the deserializer.

J9, J10 jumpers removed for SCH-28899 REV D board.

J4, J5, J6, J7 removed for SCH-29288 REV B1.

Inputs data required: No

5.7.3.2Running the example

5.7.3.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_ov10635_quad.cmm script.

OR

Load the isp_ov10635_quad.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_ov10635_quad.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.7.3.2.2 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:
```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_ov10635_quad.elf`

5.7.4ISP VIU DCU with ov10635 camera

This demo demonstrates the use of an Omnivision 10635 camera on the VIU (parallel) camera interface and displays the life image using the DCU.

Name of the application directory: **isp/ isp_ov10635_viu_dcu**

Available builds:

- S32V234 Linux

5.7.4.1Prerequisites

OV10635 camera attached to VIU (camera parallel B) interface.

Inputs data required: No

5.7.4.2Running the example

5.7.4.2.1 Linux OS

The apex.ko, oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:
```
insmod oal_cma.ko
insmod viulite.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

```
insmod apex.ko
```

The application is started by typing `./isp_ov10635_viu_dcu.elf`

5.7.5ISP ov10640 default demo

The demo shows how to operate the ov10640 MipiCsi camera in HDR mode. The exposures are combined into HDR using tone mapping and the result is display using the DCU. AEC and channel gain control (white balancing) is included as well.

Name of the application directory: **isp/isp_ov10640_default**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.7.5.1Prerequisites

Omnivision ov10640 camera attached to Mipi CSI0 interface.

Inputs data required: No

5.7.5.2Building the example

Build of isp_ov10640_default demo requires special steps to be taken to reach expected ISP graph behavior.

Please use the build.sh scripts available in each build flavor directory to reach single-command build experience and expected ISP graph behavior in runtime.

5.7.5.3Running the example

5.7.5.3.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_ov10640_default.cmm script.

OR

Load the isp_sony_dualexp.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_ov10640_default.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.7.5.3.2 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:
```
insmod oal_cma.ko
insmod csi.ko
```

```
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_ov10640_default.elf`

5.7.6ISP ov10640 Quad camera demo

This demo will show how 4 Omnivision 10640 cameras can be used with the ISP. The cameras are connected through a MAXIM deserializer.

Name of the application directory: **isp/isp_ov10640_quad**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.7.6.1Prerequisites

MAXIM deserializer attached to Mipi CSI0 interface. 4 Omnivision 10640 cameras attached to the deserializer.

J9, J10 jumpers removed for SCH-28899 REV D board.

J4, J5, J6, J7 removed for SCH-29288 REV B1.

Inputs data required: No

5.7.6.2Running the example

5.7.6.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_ov10640_quad.cmm script.

OR

Load the isp_ov10640_quad.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_ov10640_quad.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.7.6.2.2 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
```

```
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_ov10640_quad.elf`

5.7.7ISP ov9716 MIPI_S demo

This demo will show how ov9716 camera can be used with the ISP. The cameras are connected through a Mipi CSI0 interface

Name of the application directory: **isp/ isp_ov9716_mipi_s**

Available builds:

- S32V234 Linux

5.7.7.1Prerequisites

Ov9716 camera is connected to Mipi CSI0 interface.

J9, J10 jumpers removed for SCH-28899 REV D board.

J4, J5, J6, J7 removed for SCH-29288 REV B1.

Inputs data required: No

5.7.7.2Running the example

5.7.7.2.1 Linux OS

The apex.ko, oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod apex.ko
```

The application is started by typing `./ isp_ov9716_mipi_s.elf`

5.7.8ISP CSI to DCU with Sony imx224 camera

The application demonstrates simple Sony imx224 camera frame input combined with image display using DCU. The demo invokes functionality of SDI library, OAL, FDMA, Sequencer and DCU drivers. The RAW 12 bit camera data is debayered and resized by the Sequencer and through the user application code provided to the DCU driver to be displayed.

Name of the application directory: **isp/isp_sonyimx224_csi_dcu**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.7.8.1Prerequisites

Sony imx224 camera attached to Mipi CSI0 interface.

Inputs data required: No

5.7.8.2Running the example

5.7.8.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_sonyimx224_csi_dcu.cmm script.

OR

Load the isp_csi_dcu.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_sonyimx224_csi_dcu.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.7.8.2.2 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

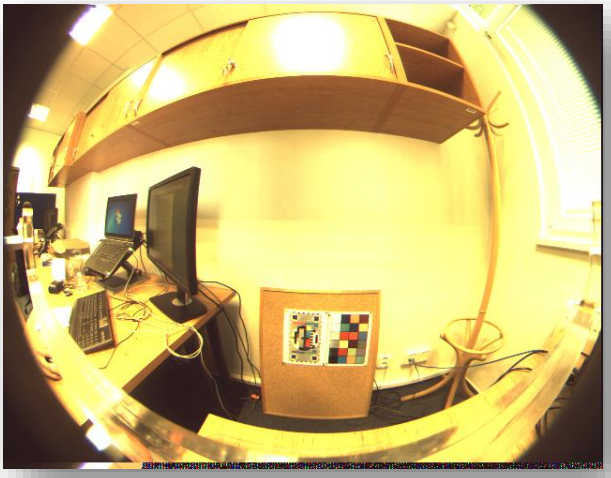
```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_sonyimx224_csi_dcu.elf`

5.7.8.3Expected input and output

Sony imx224 camera attached to MIPI CSI0 interface serves as a source of data. The preprocessed images are being displayed on internal LVDS display.

Example output of the CSI to DCU demo



5.7.9ISP Sony imx224 csi to dcu cam switch

Camera and graph switching demo: Alternating two graphs execution.

Name of the application directory: **isp/isp_sonyimx224_csi_dcu_cam_switch**

Available builds: S32V234 Linux

5.7.9.1Prerequisites

Sony imx224 master cameras are attached to Mipi CSI0 and CSI1 interfaces.

Inputs data required: No

5.7.9.2Running the example

5.7.9.2.1 Linux OS

The apex.ko, oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
insmod apex.ko
```

The application is started by typing `./isp_sonyimx224_csi_dcu_cam_switch.elf`

5.7.10 ISP Sony imx224 default

The demo shows how to operate the Sony camera in dual exposure mode (long and short exposure). The exposures are combined into HDR using tone mapping and the result is display using the DCU. AEC and channel gain control (white balancing) is included as well.

Name of the application directory: **isp/isp_sonyimx224_default**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.7.10.1 Prerequisites

Sony imx224 camera attached to Mipi CSI0 interface.

Inputs data required: No

5.7.10.2 Running the example

5.7.10.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_sonyimx224_default.cmm script.

OR

Load the isp_sony_dualexp.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_sonyimx224_default.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.7.10.2.2 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_sonyimx224_default.elf`

5.7.11 ISP Sony imx224 h264 encoder

Encodes live video from Sony imx224 using the H264 Encoder HW decoder in conjunction with ISP.

Name of the application directory: **isp/isp_sonyimx224_h264enc**

Available builds:

- S32V234 Linux

5.7.11.1 Prerequisites

Sony imx224 camera attached to Mipi CSI0 interface.

Inputs data required: No

5.7.11.2 Running the example

5.7.11.2.1 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_sonyimx224_h264enc.elf`

5.7.12ISP RGB YUV GS8 with Sony imx224 camera

This demo shows how to retrieve 3 different image formats in parallel using the ISP. 3 DDR buffers are getting filled with the camera life image in RGB888, YUV422, and Greyscale8.

Name of the application directory: **isp/isp_sonyimx224_rgb_yuv_gs8**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.7.12.1 Prerequisites

Sony imx224 camera attached to Mipi CSI0 interface.

Inputs data required: No

5.7.12.2 Running the example

5.7.12.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_sonyimx224_rgb_yuv_gs8.cmm script.

OR

Load the isp_rgb_yuv_gs8.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if= isp_sonyimx224_rgb_yuv_gs8.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.7.12.2.2 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:
```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_sonyimx224_rgb_yuv_gs8.elf`

5.7.13ISP Static simple

Demo static_simple -> dcu.

Name of the application directory: **isp/ isp_static_simple**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.7.13.1 Prerequisites

Sony imx224 camera attached to Mipi CSI0 interface.

Inputs data required: data/common/admir_bgr_1920x1080_24bpp.raw

Copy the demos/data folder to Linux home directory.

5.7.13.2 Running the example

5.7.13.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading isp_static_simple.cmm script.

OR

Load the isp_static_simple.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=isp_static_simple.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.7.13.2.2 Linux OS

The oal_cma.ko, csi.ko, cam.ko, seq.ko and fdma.ko kernel modules must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:

```
insmod oal_cma.ko
insmod csi.ko
insmod cam.ko
insmod seq.ko
insmod fdma.ko
```

The application is started by typing `./isp_static_simple.elf`

5.8 ARM NEON Related Examples

5.8.1 ARM NEON Eigen matrix add

The application computes matrix addition using Eigen library accelerated by ARM NEON. Eigen matrixes and operators are used for addition computation. As referent comparison, addition of C arrays with the same size as Eigen matrixes is computed. Both results are printed in the console as the output, showing the same result.

Name of the application directory: **neon_eigen_add**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.8.1.1 Prerequisites

5.8.1.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: No

5.8.1.1.2 Linux OS

The oal_cma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing: `insmod oal_cma.ko`

Input data required: No

5.8.1.2 Running the example

5.8.1.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading neon_eigen_add.cmm script.

OR

Load the neon_eigen_add.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=neon_eigen_add.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.8.1.2.2 Linux OS

Application starts as follows:

```
./neon_eigen_add.elf
```

5.8.2ARM NEON Gauss 3x3 Filter

The application demonstrates simple one-kernel algorithm using ARM NEON SIMD instructions. ARMv8 architecture is used and NEON instructions are realized through inline assembly code. The demo loads an image and executes the Gaussian blur convolution with 3x3 window on it. As the output original and blurred images are displayed on the EVB display.
Name of the application directory: **neon_gauss3x3_cv**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.8.2.1Prerequisites

5.8.2.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: data/common/headers/in_grey_256x256.h

Copy the demos/data to C:/vsdk

5.8.2.1.2 Linux OS

The oal_cma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing: `insmod oal_cma.ko`

Inputs data required: data/common/in_grey_256x256.png

Copy the demos/data folder to Linux home directory.

5.8.2.2Running the example

5.8.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading neon_gauss3x3_cv.cmm script.

OR

Load the neon_gauss3x3_cv.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=neon_gauss3x3_cv.bin of=/dev/sdb bs=512 seek=0
conv=fsync
```

5.8.2.2.2 Linux OS

Application starts as follows:

```
./neon_gauss3x3_cv.elf
```

5.9 Other applications

5.9.1H264 Encoder Application

The application demonstrates H264 Encoder driver usage. In the application H264 Encoder HW and SRAM input/output buffers configurations are provided. Input frames are captured from video sequence via OpenCV and encoding is realized. As the output, encoded sequence is saved into the file.

Name of the application directory: **other/h264_encoder_cv**

Available builds:

- S32V234 Linux

5.9.1.1Prerequisites

5.9.1.1.1 Linux OS

The oal_cma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing:
```
insmod oal_cma.ko
insmod h264enc.ko
```

Inputs data required: data/common/Megamind.avi

Copy the demos/data to Linux home directory.

5.9.1.2Running the example

5.9.1.2.1 Linux OS

Application starts as follows:

```
./h264_encoder_cv.elf
```

5.9.2Hello World Application

Hello world application is the simplest application in the SDK. The run itself consists only of one printf showing a text on console. The used can however use it in order to get familiar with the build and SD card boot.

Name of the application directory: **other/hello**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.9.2.1Prerequisites

5.9.2.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: No

5.9.2.1.2 Linux OS

The oal_cma.ko kernel module must be loaded prior to application launch (already loaded in provided OS). Do it by executing: `insmod oal_cma.ko`

Inputs data required: No

5.9.2.2Running the example

5.9.2.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading hello.cmm script.

OR

Load the hello.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if=hello.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.9.2.2.2 Linux OS

Application starts as follows:

```
./hello.elf
```

5.9.3Multi Thread

Multi thread demo shows the use of OAL Task and OAL Semaphore functionality. It starts a number of threads (OAL Tasks) and reverts the succession of the execution by the OAL Semaphores.

Name of the application directory: **other/multi_thread**

Available builds:

- S32V234 Standalone
- S32V234 Linux

5.9.3.1Prerequisites

5.9.3.1.1 OS Less application

There are no specific requirements to run this application.

Inputs data required: No

5.9.3.1.2 Linux OS

The oal_cma.ko kernel module must be loaded prior to application launch (already loaded in provided OS)

Do it by executing: `insmod oal_cma.ko`

Inputs data required: No

5.9.3.2Running the example

5.9.3.2.1 OS Less application

Application is loaded via Lauterbach Debugger by loading multi_thread.cmm script.

OR

Load the multi_thread.bin to the SD card and run the board:

- On windows, please use for example Win32 Disk Imager application for loading the image
- On Linux system, use dd:

```
sudo dd if= multi_thread.bin of=/dev/sdb bs=512 seek=0 conv=fsync
```

5.9.3.2.2 Linux OS

Application starts as follows:

```
./multi_thread.elf
```

5.9.4Web server Application

Web_server application demonstrate how to communicate with EVB using web socket communication. This demo streams random images which looks like noise and some other example values. Demo also shows how some controlling parameters can be shared.

For more detailed description please follow **/docs/vsdk/ Webserver_User_Guide.pdf.**

Name of the application directory: **other/web_server**

Available builds:

- S32V234 Linux

5.9.4.1Prerequisites

5.9.4.1.1 Linux OS

The oal_cma.ko kernel module must be loaded prior to application launch (already loaded in provided OS). Do it by executing: `insmod oal_cma.ko`

Inputs data required: No

5.9.4.2Running the example

5.9.4.2.1 Linux OS

Application starts as follows:

```
./web_server <IP address of EVB> 7777
```

Than open **webserver.html** in root folder of this demo. (!!! Do not copy it out because this file points some JavaScript files stored in VSDK)

6 Known problems and TODOs

- Three demos sometimes flicker during displaying: pedestrian_detection, pedestrian_detection_aggcf, lane_departure_warning. Will be fixed in upcoming vsdk hotfix release.
- uintptr_t is not defined in isp/inc/typedefs.h for other compilers than gcc, mwks, ghs
- The following examples need 'S32 Design Studio for Vision 2018.R1' (upcoming version) to compile them. In the meanwhile, please use the pre-compiled version provided within this VSDK release.
  - static_simple
  - imx224_exp_ctrl
  - rgb_yuv_gs8
  - yuv_grey_pyramid
  - h264enc
  - ar0231_max9296b
- Demo isp_ov10640_quad rarely failed to initialize cameras if rerun contiguously. Seems MAXIM or ov10640 I2C issue. It's necessary to replug the MAXIM power supply or turn on/off the board by hand.
- Demo isp_sonyimx224_h264enc on SBC board get overflow error. Will be fixed in upcoming vsdk hotfix release.
- Some IPUV engines are using IPUS configuration layout. Cause no real issue, IPUS and IPUV registers share the same layout. Will be fixed with 'S32 Design Studio for Vision 2018.R1' (upcoming version)