This 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 the board, an ACF implementation is used. By this approach,

it's possible to demonstrate use of the kernels in both emulated and real environment

The example does not show any images on the screen.

It outputs statistics: use COM-port terminal with settings: 115200 b/s 8 bits, no parity, 1 stop bit.

The project contains the configuration to run the application on Linux (A53).

APEX debugging:

The provided breakpoint "apu\_hal\_Enable" is disabled by default,

it should be enabled to debug APEX code via S32 Debugger.

Due to a known issue with the version of Eclipse CDT with which S32 Design Studio is integrated,

it will require some extra steps to properly enable the provided breakpoint "apu\_hal\_Enable".

Click to check the box, click again to uncheck the box, then click a 3rd time to again check the box.

Execute "Resume" command twice.

Start APEX debugging after the breakpoint is reached at the second time.

Breakpoints at <graph\_name> functions are set by default.

The execution is in "runnning" state on APEX debugging thread.

Execute "Resume" command on A53 application.

The execution stops on <graph\_name> breakpoint on APEX debugging thread.

Start APEX debugging.

Requirements:

S32V234 board with Linux.

Prerequisite to connect s32v234 board

**Integrated Development Environment (IDE)**: It is a software application that provides facilities to software programmer to develop software.

**The graphical user interface(GUI):**it is a form of [user interface](https://en.wikipedia.org/wiki/User_interface) that allows [users](https://en.wikipedia.org/wiki/User_(computing)) to [interact with electronic devices](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) through graphical [icons](https://en.wikipedia.org/wiki/Computer_icon) and visual indicators such as secondary notation, instead of [text-based user interfaces](https://en.wikipedia.org/wiki/Text-based_user_interface), typed command labels or text navigation.

**Graphics Processing Unit (GPU)**: is a specialized [electronic circuit](https://en.wikipedia.org/wiki/Electronic_circuit) designed to rapidly manipulate and alter [memory](https://en.wikipedia.org/wiki/Memory_(computing)) to accelerate the creation of [images](https://en.wikipedia.org/wiki/Image) in a [frame buffer](https://en.wikipedia.org/wiki/Frame_buffer) intended for output to a [display device](https://en.wikipedia.org/wiki/Display_device). GPUs are used in [embedded systems](https://en.wikipedia.org/wiki/Embedded_system), [mobile phones](https://en.wikipedia.org/wiki/Mobile_phone), [personal computers](https://en.wikipedia.org/wiki/Personal_computer), [workstations](https://en.wikipedia.org/wiki/Workstation), and [game consoles](https://en.wikipedia.org/wiki/Game_console). Modern GPUs are very efficient at manipulating [computer graphics](https://en.wikipedia.org/wiki/Computer_graphics) and [image processing](https://en.wikipedia.org/wiki/Image_processing). Their highly parallel structure makes them more efficient than general-purpose [CPUs](https://en.wikipedia.org/wiki/Central_processing_unit) for [algorithms](https://en.wikipedia.org/wiki/Algorithm) that process large blocks of data in parallel. In a personal computer, a GPU can be present on a [video card](https://en.wikipedia.org/wiki/Video_card) or embedded on the [motherboard](https://en.wikipedia.org/wiki/Motherboard). In certain CPUs, they are embedded on the CPU.

**APEX (Algorithm Prototyper and EXperimentor for Cognitive Systems):**

It is highly object-oriented programming language.

**Short Overview how to make an apex graph and from that how to make an apex program:**

* At first need to create an apex 2 graph project (named as APEX\_VGT\_test\_Graph)
* From the Palette window apu\_upsample and apu\_downsample(resizing\_Kernels) were added from the Add Kernels.
* Then the input and output graph blocks were connected by the connector.
* The names can be changed from the properties.
* Then the graph is validated by right clicking anywhere in the white part of the graph.
* Then Apex program is created (named Apex\_VGT\_test\_program)
* Apex\_VGT\_test\_Graph is selected from the process from graph block.
* Image Inlet and image Outlet is created.
* After that from S32 DS Application project new APEX\_VGT\_test\_application project is created.
* The picture “in\_grey256x256.png”is copied from s32ds installation\_directory\S32DS\s32v234  
  \_sdk\demos\data\common and pasted to APEX\_VGT\_test\_application project folder.
* The inlet and outlet image are configured.
* The up sample and down sample blocks are configured from their properties.
* Then the validation is done.
* The destination of the auto generated source code is selected.
* Then the source code is emited.
* Then the linux application was build.

**Required action to prepare SD card for Linux boot:**

**The reason of using BSP Image:**

To support a real time operating system (RTOS) we must create a BSP (Board Support pakage) that allows the RTOS to run on their platform. In most cases the RTOS image and license, the BSP containing it, and the hardware are bundled together by the hardware vendor.

Therefore, we need to download the BSP image and write it to our SD card for Linux Boot.

We can download the BSP image by following steps:

login to nxp.com Software Licensing and Support Automotive SW - Linux' Next select 'SW32XX-LINUXBSP01-RTM-V20 Download Precompiled binaries with VSDK binaries\_auto\_linux\_bsp20.0\_vsdk.tgz

VSDK binaries\_auto\_linux\_bsp20.0\_vsdk.tgz is our required BSP image.

After that we need to write this BSP image to the SD card by using Win32DiskImager software.

**ACF(Apex Core Framework):**The main work of APEX core framework is combining multiple processing task into a single process makes the data movement efficient.

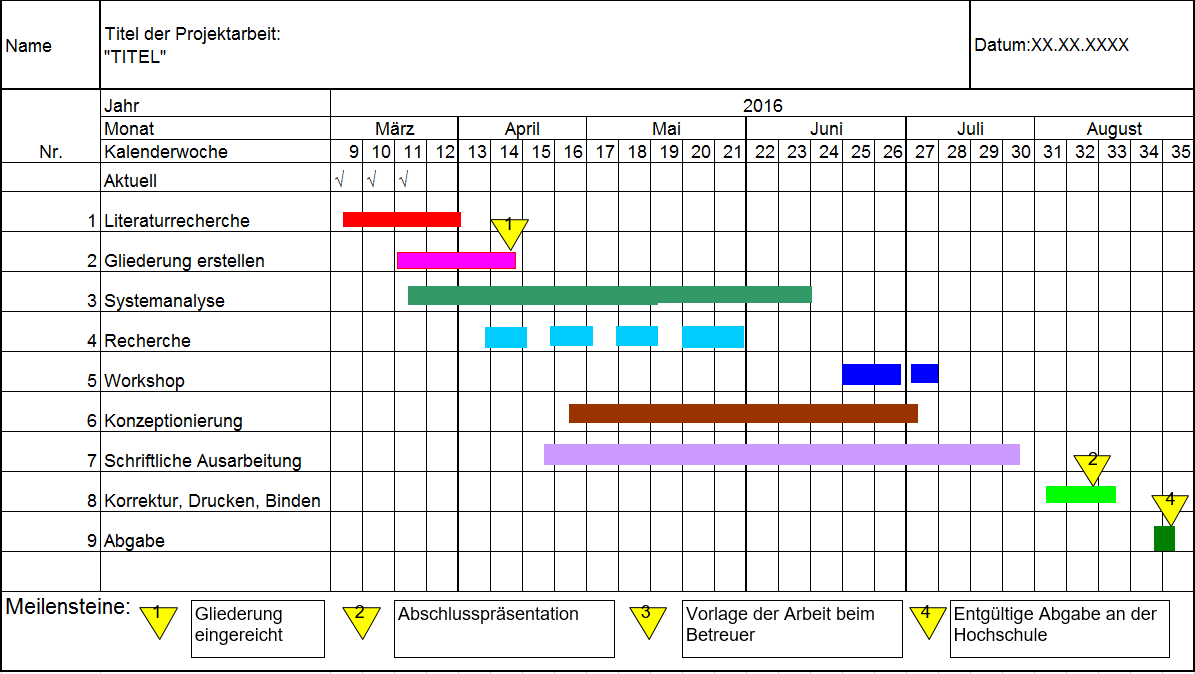
**Kernel Definition:**A kernel is a well-defined unit of processing that executes on a specific processor. It takes well-defined inputs, processes them, and produces well-defined outputs. Exactly what goes on inside a kernel is generally unknown to the framework (i.e. it is more or less a black box), however, interface and meta-data requirements must be adhered to by all kernels.

**Vectorization:**

In the ACF context, vectorization refers to the subdivision of input data into smaller pieces (i.e. chunks) for the purpose of distribution across multiple processors to be processed in parallel (i.e. data level parallelism).

**Tiling:**

A tile is a grouping of one or more chunks in a row. In the ACF context, tiling refers to the subdivision of input data into ‘tiles’ for sequential or iterative processing. The need for tiling is in part a consequence of limited local APEX memory. For example, the APU has relatively small amounts of local memory. In typical use cases, input data sizes are much too large to fit entirely into CMEM (e.g. a megapixel image), so input data must be subdivided into tiles and moved into APU memory, processed, and moved out of APU memory in a producer/consumer fashion.#

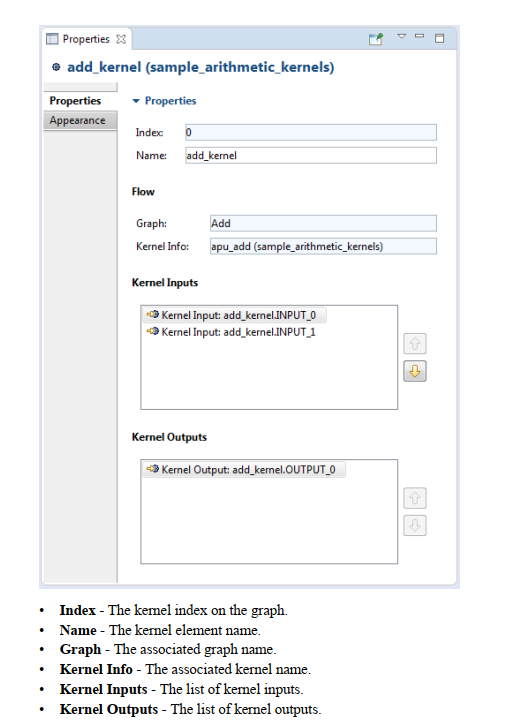


|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Project Name:Setup a Lane keeping assistance system evaluation demonstration using NXP SBC-S32V234 Evaluation Board | | | | | | | | | | | | | | | | | | | | | Date:24.05.19 | | | | |
| Nr. | Year | 2019 | | | | | | | | | | | | | | | | | | | | | | | | |
| Month | April | | | | May | | | | June | | | | July | | | | August | | | | | September | | | |
| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | | 21 | 22 | 23 | 24 |
|  | S32V234 Board Evaluation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| S32DS Software Installation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| Running Demo project on Emulator |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| Preparation of SD Card |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
|  | Creation of own Demo Project |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
|  | Camera setup |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
|  | NXP Setup Environment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
|  | Analyzing Vision SDK architecture and change in required programming |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| Image processing and calculation of required parameters on NXP Board |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| Sending Processed Image and required parameters to Simulator via CAN bus and Ethernet Bus |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
| Comparison of calculated data with ground truth Simulated Environment |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
|  | Probable errors fine tuning |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |
|  | Documentation and writing thesis report |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |

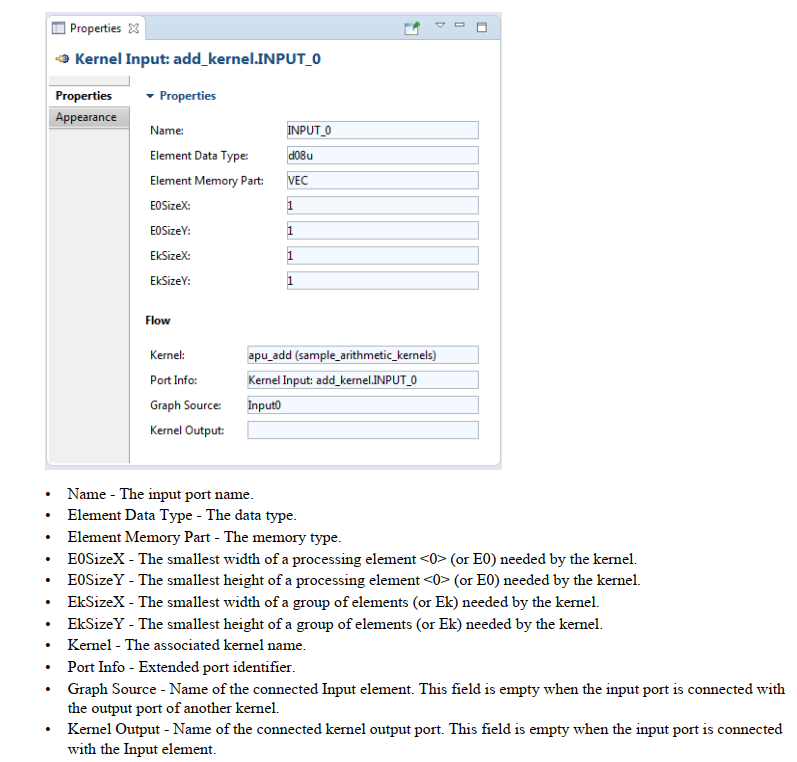
**APEX2 Kernel Properties:**

The Add Kernel allows us to choose a kernel from the custom or library kernels.

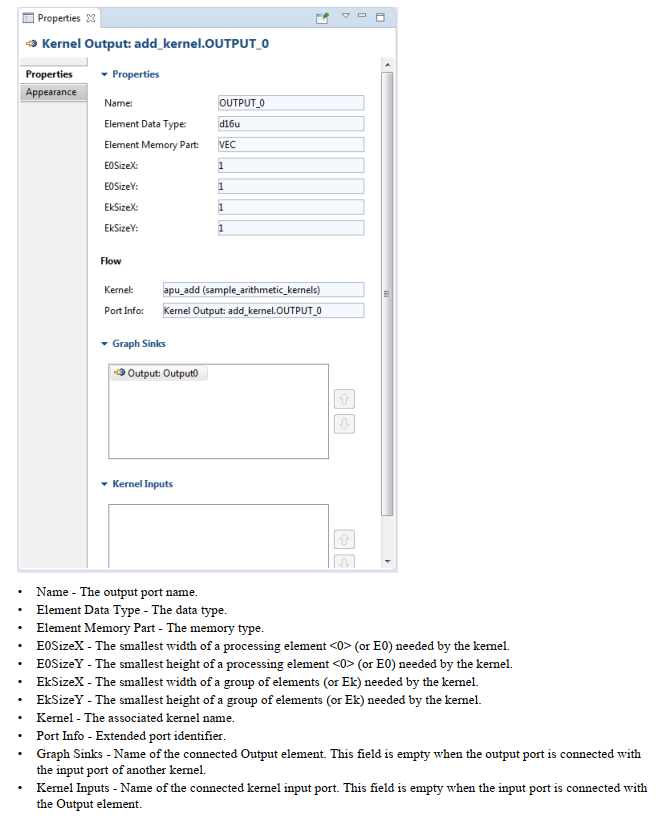
The Add kernels has the following properties:



**The Kernel inputs has the following properties:**



**The kernel output has the following properties:**

**Image Inlet and Image Outlet:**

The image inlet and image outlet allow us to create input and output for the input and output images. The parameters of image inlet and image outlet is given below:

**ROI Element**:The ROI element allows us to define a Region of Interest (ROI) - subset of data fully contained within a larger  
source region. You can define ROI for input and output data.  
For example, you have 640x480 input image and want to process a 320x240 ROI that corresponds to the top left  
quadrant of the 640x480 source region. In your workspace it will look like this:

**The procedure to create an ISP dataflow project and ISP Application project:**

* The ISP dataflow graph is created at first.
* Then camera parameters and other parameters are added for different components on the ISP dataflow graph.
* Then the ISP dataflow graph is validated
* To generate the source code from the ISP data flow graph the ISP Application project is created
* Then the ISP dataflow graph is emitted.
* Then main.cpp generated in A53\_src is modified to define the DDR Buffers.
* The isp\_user\_define.h file is edited to define the image type.
* Then the ISP Application project is build for the A53 Core.