Enabling Vc4Graphics and v4l2 on RPI

- **This Page is under Development**

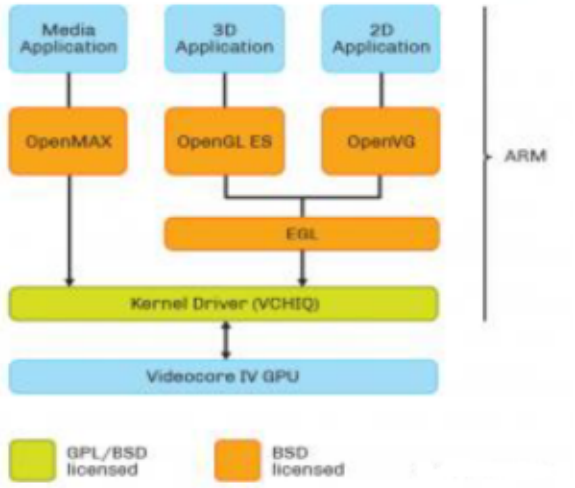
1.1. Introduction

The Broadcom VideoCore 4 (present in the Raspberry Pi) contains a OpenGL ES 2.0-compatible 3D engine called V3D, and a highly configurable display output pipeline that supports HDMI, DSI, DPI, and Composite TV output. The 3D engine also has an interface for submitting arbitrary compute shader-style jobs using the same shader processor as is used for vertex and fragment shaders in GLES 2.0. Closed source graphics stack runs on VC4 GPU and talks to V3D and display component. Vc4 uses mesa instead of userland for graphics. Mesa is an open source software implementation of OpenGl, Vulkan and other graphics API specification. Mesa translates these specifications to vendor specific graphics hardware drivers.

1.2. RPI Structure

1.2.1. Current RDK CMF

Current RPI structure uses userland to send OpenGL commands to GPU/VPU through VCHIQ. One of the major drawback of current approach is OpenMax IL for 64 bit is not officially supported. VCHIQ also needs changes for supporting 64 bit architecture.



1.2.2. New Structure

An alternative for userland and dispmanx is vc4graphics and v4l2. Vc4 uses mesa instead of userland for graphics and video. Mesa implements a translation layer between a graphics API such as OpenGL and the graphics hardware drivers in the operating system kernel. The supported version of the different graphic APIs depends on the driver, because each hardware driver has its own implementation.

blocked URL

1.3. Efforts Involved to enable vc4graphics

In order to enable vc4graphics on RPI, below changes are made in configuration.

meta-raspberrypi layer:

With new changes, vc4graphics is added to Machine features and headers from userland is used for device-hal compilation. Mesa library is enabled for vc4 supported architecture. Graphics libraries like Egl and Gles2 from userland is suppressed and is taken from Mesa. OpenGL (Open graphics Library), API for interacting with GPU (graphical Processing Unit) is enabled with vc4.

meta-wpe layer :

Userland dependency of wpeframework and snapshot feature is removed with vc4 changes. wayland opengl is enabled for gstreamer plugins. Gstreamer packages are enabled with GLES2 and EGL libraries.

Westeros Changes :

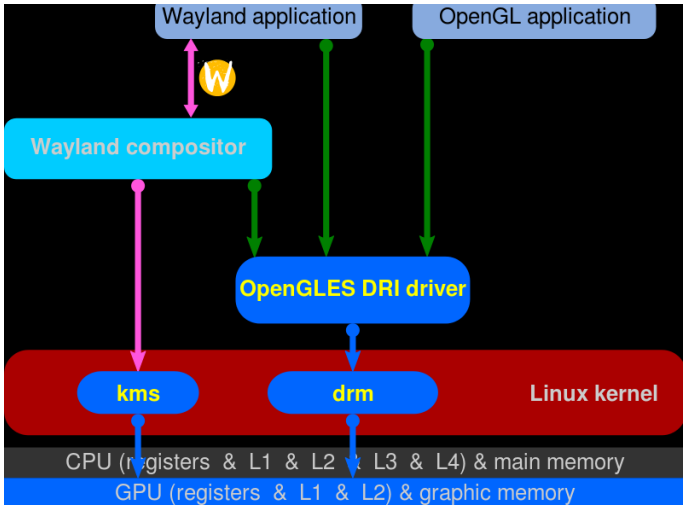
Userland uses dispmanx for rendering graphics and video. With vc4graphics instead of dispmanx, DRM (Direct Rendering Manager) module is used. DRM is a subsystem of the Linux kernel responsible for interfacing with GPUs of modern video cards. DRM is a kernel module that gives direct hardware access to DRI clients. Westeros-dispmanx is suppressed and westeros-drm is used for rendering. Westeros-sink is enabled to use v4l2 by modifying soc-path to v4l2. Westeros was updated to latest revision for supporting video codecs that are already used.

1.3.1. List of changes done

- https://code.rdkcentral.com/r/c/components/generic/rdk-oe/meta-cmf-raspberrypi/+/39272
- https://github.com/WebPlatformForEmbedded/meta-wpe/pull/450
- https://github.com/agherzan/meta-raspberrypi/pull/647, this pull was modified for CMF as https://code.rdkcentral.com/r/c/rdk/components/opensource/oe/meta-raspberrypi/+/39347
- https://code.rdkcentral.com/r/c/components/generic/rdk-oe/meta-cmf-raspberrypi/+/39279
- https://code.rdkcentral.com/r/c/components/generic/rdk-oe/meta-cmf-raspberrypi/+/39287
- https://code.rdkcentral.com/r/c/components/generic/rdk-oe/meta-cmf-raspberrypi/+/39285
- https://code.rdkcentral.com/r/c/components/generic/rdk-oe/meta-cmf-raspberrypi/+/39284

1.4. System Work Flow

Direct Rendering Infrastructure (DRI) provide interface to interface Mesa, OpenGL and other 3D rendering API libraries with the device drivers and hardware. Mesa also contains an implementation of software rendering called swrast that allows shaders to run on the CPU as a fallback when no graphics hardware accelerators are present. The Gallium software rasterizer is known as *softpipe* or when built with support for LLVM *llvmpipe*, which generates CPU code at runtime



1.5. Environment Setup

The build procedure is as follows:

- repo init -u https://code.rdkcentral.com/r/manifests -m rdkv-asp-nosrc.xml -b thunder-next
- repo sync -j4 --no-clone-bundle
- source meta-cmf-raspberrypi/setup-environment (select option raspberrypi-rdk-hybrid-thunder.conf)
- bitbake rdk-generic-hybrid-thunder-image

- Cloning the code before login once to code.rdkcentral.com, user would get the Authentication error, even though the account is in good standing and has all the required access.
- Please login to code.rdkcentral.com before attempting to clone.

- ### 1.5.1. Flashing the image

  Image is flashed to micro SD card before inserting to RPI board.

  Command to flash the image

  Generated image has to be flashed to an micro SD card using this command in local PC:

$ sudo dd if=<path to ImageName.rpi-sdimg> of=<path to micro SD card space> bs=4M

Ex:

$ sudo dd if=rdk-generic-hybrid-thunder-image_default_20200302130659.rootfs.rpi-sdimg of=/dev/sdc bs=4M
317+0 records in
317+0 records out
1329594368 bytes (1.3 GB, 1.2 GiB) copied, 104.88 s, 12.7 MB/s
$ sync

The micro SD card is then inserted to the Raspberry Pi board and booted to check for containers created. The Raspberry Pi board is connected to the PC via a USB to serial converter and the logs can be checked in console or can be connected via HDMI cable to a TV and logs will be shown in the terminal

1.6. **References**

- http://heim.ifi.uio.no/~knuto/kernel/4.14/gpu/vc4.html?highlight=vc4
- https://en.wikipedia.org/wiki/Free_and_open-source_graphics_device_driver
- https://en.wikipedia.org/wiki/Mesa_(computer_graphics)
- https://www.raspberrypi.org/blog/open-source-arm-userspace/
- https://en.wikipedia.org/wiki/EGL_(API)
- https://www.khronos.org/opengl/wiki/Platform_specifics:_Linux
- https://en.wikipedia.org/wiki/OpenGL
- https://elinux.org/Raspberry_Pi_VideoCore_APIs