SAMSUNG

ARTIK™ Modules

GPU Guide

TABLE OF CONTENTS

LIST OF FIGURES

VERSION HISTORY

Revision	Date	Description	Maturity
V1.0	March 13.2017	Software GPU Guide	Release

PLATFORM SUPPORT

This document will describe how to work with the GPU. Currently the following Samsung ARTIK™ Hardware Platforms are supported in this document:

Platform	Version
ARTIK 530 Module	No Limitation, holds for all released versions
ARTIK 710 Module	No Limitation, holds for all released versions

EXPLORING THE GPU

INTRODUCTION

This guide describes the use of the GPU graphics stack that can be deployed on Samsung ARTIK™ Modules.

The ARTIK 520 and ARTIK 530 Module has a Mali400™-MP2 (2x cores) hardware accelerator supported by a vendor independent Java OPC client named 'utgard'.

The ARTIK 710 Module has a Mali400™-MP4 (4x cores) hardware accelerator supported by a vendor independent Java OPC client named 'utgard'.

The Mali™-utgard driver consists of a kernel driver and a Driver Developers Kit (DDK) that operates in user mode. The kernel driver is fully opened as a GPLv2 License. The DDK is an ARM® license with proprietary source code.

The 'utgard' driver supports OpenGL® ES 1.1 and 2.0. The Mali™ 'utgard' DDK supports an 'fbdev' X Window System using a Wayland backend. Each backend has an associated binary package that must be downloaded separately.

All software drivers necessary for using the GPU environment that are referenced during the course of this document will be provided by Samsung in a 'GPU Library Pack'. For access to the Mali™ DDK please contact your Samsung sales representative.

ARCHITECTURE

The architecture of the Mali environment with Wayland DDK integration can be seen in *Figure 1*.

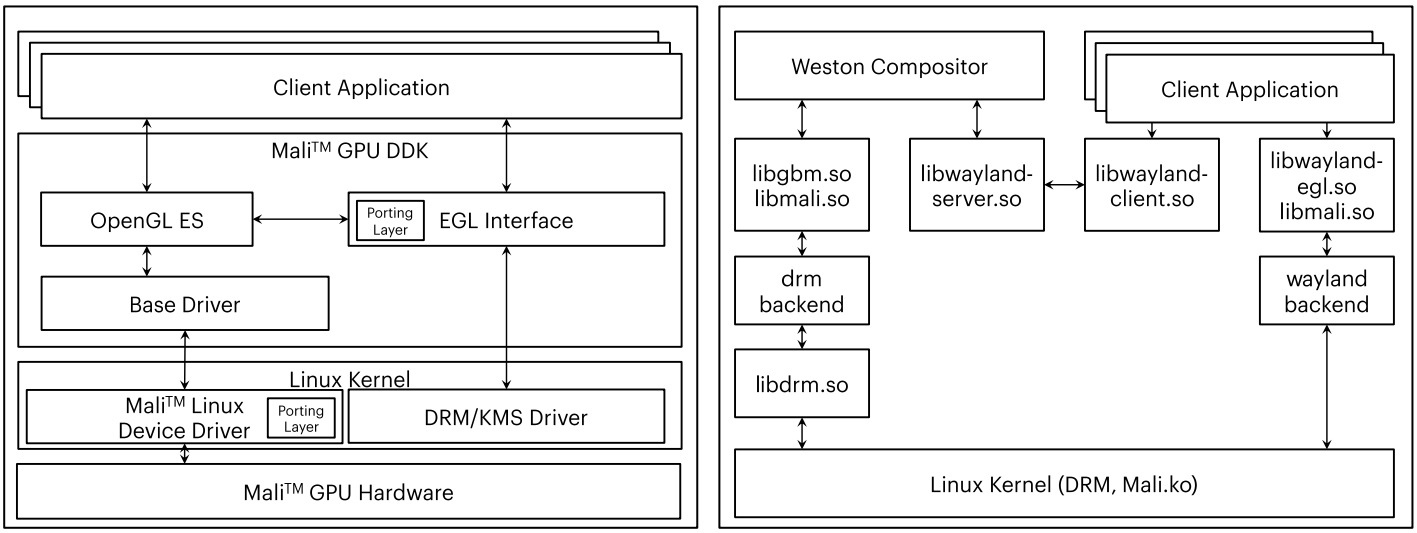

Figure 1. Mali™ GPU Architecture using Wayland DDK

DRIVER FUNCTIONALITY

The following provides a list on what graphical features are supported with the GPU driver package:

- Pixel formats
  - 32 bpp ARGB8888
  - 16 bpp ARGB4444
  - 16 bpp ARGB1555
  - 16 bpp RGB565
- OpenGL Support
  - OpenGL ES 1.1: This API is primarily for 2D and 3D graphics, using a fixed-function pipeline architecture.
  - OpenGL ES 2.0: This API is primarily for 3D graphics. Unlike OpenGL ES 1.1, it is based on a programmable pipeline. Transformation, lighting, texturing and alpha blending are fully programmable.
- EGL interface

    - The EGL interface is a standard interface that connects the OpenGL ES drivers to the platform-specific windowing system to:
        - Create rendering surfaces where the API drivers can draw
        - Create graphics contexts for the API driver
        - Synchronize native platform rendering and Mali GPU rendering
        - Use the Base driver to manage the Mali GPU and memory allocation
    - Supported Native window system
        - Wayland-EGL
        - DRM/KMS backend
        - Framebuffer (fbdev)
- Wayland integration with Mali DDK
    - Base Wayland version of ARTIK Mali ddk : Wayland 1.10
    - Mali Wayland support for backend that supports Wayland-drm backend only.

PREREQUISITES

In order to use this graphical user interface and before you install the DDK packages, make certain that the following components are in place for the Samsung ARTIK™ Module:

- Install the latest ARTIK Module firmware binary.
- Recommend Ethernet connection for downloading rpm files.
- A display supporting MIPI-DSI, LVDS or HDMI.
- Connect a USB Keyboard and mouse.

INSTALL GPU DDK PACKAGE FOR WAYLAND-DRM

- Fedora
    - Download the 'GPU Library Pack' provided to you by Samsung, select 'op opengl-es-mali-utgard-7.0-0.armv7hl.rpm' for the Wayland-drm backend
    - Send the package to your board and place the file in your selected directory.
    - Install the package use the following command:

```
$ rpm –ivh opengl-es-mali-utgard-7.0-0.armv7hl.rpm
```

- Ubuntu:
    - Download the 'GPU Library Pack' provided to you by Samsung
    - Send the following packages to your boad and place them in your selected directory
        - libwayland-client0_1.10.0-1_armhf.deb
        - libwayland-server0_1.10.0-1_armhf.deb
        - opengl-es-mali-utgard-wayland_7.0-1_armhf.deb
    - Install the packages using the following commands

```
$ dpkg -i libwayland-client0_1.10.0-1_armhf.deb libwayland-server0_1.10.0-1_armhf.deb opengl-es-mali-utgard-wayland_7.0-1_armhf.deb

$apt-get install -f
```

- Install GPU ddk library on other system
    - The 'opengl-es-mali-utgard-7.0-0.armv7hl.rpm' installs the library files and generates symlinks for opengl libraries. If you want to install it to another system like, yocto, you have to generate the appropriate symlinks.
    - There are two options to extract the libMali.so
        - Extract the rpm using:

```
$ rpm2cpio opengl-es-mali-utgard-7.0-0.armv7hl.rpm | cpio -ivd
```

- Extract the deb using

```
$ dpkg -x opengl-es-mali-utgard-wayland_7.0-1_armhf.deb .
```

- You have to install 'libMali.so' into the proper location as depicted below. To avoid mesa conflicts, the library will be installed under '/usr/lib/driver' and 'ldconfig' will indicate the location prior to installing the mesa library.

```
$ mkdir /usr/lib/driver
$ cp libMali.so /usr/lib/driver
$ cd /usr/lib/driver
$ ln -sf libMali.so           libEGL.so.1.4
$ ln -sf libEGL.so.1.4        libEGL.so.1
$ ln -sf libEGL.so.1          libEGL.so
$ ln -sf libMali.so           libGLESv1_CM.so.1.1
$ ln -sf libGLESv1_CM.so.1.1  libGLESv1_CM.so.1
$ ln -sf libGLESv1_CM.so.1    libGLESv1_CM.so
$ ln -sf libMali.so           libGLESv2.so.2.0
$ ln -sf libGLESv2.so.2.0     libGLESv2.so.2
$ ln -sf libGLESv2.so.2       libGLESv2.so

# You have to make libgbm symlink except fbdev backend
$ ln -sf libMali.so       libgbm.so.1
$ ln -sf libgbm.so.1      libgbm.so

# If you want to use wayland-backend, you have to make libwayland-egl symlink.
$ ln -sf libMali.so       libwayland-egl.so.1
$ ln -sf libwayland-egl.so.1  libwayland-egl.so
```

- After installation,, you have to make mali.conf for ldconfig and run ldconfig

```
$ echo "/usr/lib/driver" > /etc/ld.so.conf.d/mali.conf
$ /sbin/ldconfig
```

INSTALL WESTON

The next step is to install the reference compositor named Weston. Weston is the reference compositor of Wayland.

- Fedora

  Please use:

```
$ dnf install weston
```

- Ubuntu
  - Download weston_1.10.0-1_armhf.deb, libwayland-cursor0_1.10.0-1_armhf.deb
  - Install the packages

```
$ dpkg -i weston_1.10.0-1_armhf.deb libwayland-cursor0_1.10.0-1_armhf.deb
$ apt-get install -f
```

LAUNCH WESTON SHELL

Before executing the Weston shell first create a script called 'start_wayland.sh':

```
#/bin/sh

export XDG_RUNTIME_DIR="/run/shm/wayland"
mkdir -p "$XDG_RUNTIME_DIR"
chmod 0700 "$XDG_RUNTIME_DIR"

/usr/bin/weston --backend=drm-backend.so --tty=1 --log=/var/log/weston.log
```

Change the script's permission to executable using:

```
$ chmod a+x start_wayland.sh
```

Configure your display creating 'weston.ini'. An example of such a file is given below:

```
[core]
modules=xwayland.so

[shell]
background-color=0xff002244
panel-color=0x90ff0000
locking=true
animation=zoom

[input-method]
path=/usr/libexec/weston-keyboard

[output]
name=DSI-1
mode=1080x1920
#transform=90

[output]
name=HDMI-A-1
mode=1920x1080

[output]
name=LVDS-1
```

```
mode=1024x600
```

Place 'weston.ini' into '~/.config/' like:

```
$ mkdir -p ~/.config/
$ cp weston.ini ~/.config/
```

And run the 'start_wayland.sh' file using:

```
$ ./start_wayland.sh
```

BENCHMARK THE SYSTEM

A benchmark that is used regularly in the OpenGL community is the 'glmark2'. This benchmark supports both 'drm/kms' and the Wayland backend. This implies that you can run this benchmark system without the use of any windows environment.

- Fedora
  - First access the 'glmark2' software that is part of the GPU Library Pack provided by Samsung

```
$ tar xf glmark2.tar.gz
$ cd glmark2
$ dnf install *.rpm
```

- Ubuntu
  - Install the following packages
    - glmark2_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb
    - glmark2-es2_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb
    - glmark2-es2-drm_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb
    - glmark2-es2-wayland_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb

```
$ dpkg -i glmark2_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb glmark2-
es2_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb glmark2-es2-
drm_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb glmark2-es2-
wayland_2014.03+git20150611.fa71af2d-0ubuntu2_armhf.deb


$ apt-get install -f
```

Then, run 'glmark2' on the terminal (not the Wayland terminal) using:

```
$ glmark2-es-drm
```

Now open a Weston terminal and run 'glmark2':

```
$ glmark2-es-wayland
```

QT5

QT is a cross-platform application development framework.

INSTALL QT5 LIBRARIES

- Fedora

  Access 'qt5-rpms.tar.gz' from the 'GPU Library Pack' and place it in your selected directory. Then install the qt5 libraries using the following command:

```
$ tar xf qt5-rpms.tar.gz
$ cd qt5-rpms
$ dnf install *.rpm
```

- Ubuntu

  Access the following packages from the 'GPU Library Pack' and place it in your selected directory. The install the libraries.
  - libdrm-dev_2.4.70-1.1_armhf.deb
  - libwayland-bin_1.10.0-1_armhf.deb
  - libwayland-dev_1.10.0-1_armhf.deb

```
$ dpkg -i libdrm-dev_2.4.70-1.1_armhf.deb libwayland-bin_1.10.0-1_armhf.deb libwayland-dev_1.10.0-1_armhf.deb

$ apt-get install qt5-default qtwayland5
```

INSTALL QT5 EXAMPLES

- Fedora

  Access the example files 'qt5-examples-rpms.tar.gz' from the 'GPU Library Pack' and use:

```
$ tar xf qt5-examples-rpms.tar.gz
$ cd qt5-examples-rpms
$ dnf install *.rpm
```

- Ubuntu

  Install the examples

```
$ apt-get install qtbase5-examples qtdeclarative5-examples qtquick1-5-examples
```

INSTALL QT5 DEVEL

- Fedora

  Access the development environment 'qt5-devel-rpms.tar.gz' from the 'GPU Library Pack' and use:

```
$ tar xf qt5-devel-rpms.tar.gz
$ cd qt5-devel-rpms
$ dnf install *.rpm
```

- Ubuntu

```
$ apt-get install qmlscene qml-module-qtgraphicaleffects
```

You are now in a position to run some of the examples.

RUN A QT5 EXAMPLE ON EGLFS

The Qt5 does support EGLFS for an embedded system. EGL is an interface between OpenGL and the native windowing system. Qt5 will use EGL for context and surface management, without using platform specific calls. EGLFS is a platform plugin for running Qt5 applications on top of EGL and OpenGL ES 2.0 without an actual windowing system.

More documentation on the Qt5 framework can be found under 'http://doc.qt.io/qt-5/embedded-linux.html'.

- Qt5 does support several backend solutions such as EGLFS, Wayland-egl, Wayland, Xcb. The specific backend solution used can be enabled using the '-platform' option as below:

```
$ ./<your_qt5-sample> -platform eglfs
$ ./<your_qt5-sample> -platform wayland-egl
```

  Alternatively you can set all 'qt apps' using the same backend through a generic environment variable using:

```
$ export QT_QPA_PLATFORM=eglfs
```

- Cube
  - Fedora
    When the right backend has been enabled you can run an example using:

```
$ cd /usr/lib/qt5/examples/opengl/cube
$ ./cube
```

  - Ubuntu
    Depends on architecture, for example, ARTIK 530 armhf will be located /usr/lib/arm-linux-gnueabihf/qt5

```
$ cd /usr/lib/arm-linux-gnueabihf/qt5/examples/opengl/cube
$ ./cube
```

- KMS Output Configuration
  - DRM/KMS setting can be referenced using the 'QT_QPA_EGLFS_KMS_CONFIG' environment variable.
  - Create a 'kms.json' like:
    - The 'UNKNOWN1' means your MIPI_DSI display.

```
{
        "device": "/dev/dri/card0",
        "hwcursor": false,
        "outputs": [
                {
                        "name": "LVDS1",
                        "mode": "1024x600"
                },
                {
                        "name": "UNKNOWN1",
                        "mode": "1080x1920"
                },
                {
                        "name": "HDMI1",
                        "mode": "1920x1080"
                }
        ]
}
```

  - Now export the 'QT_QPA_EGLFS_KMS_CONFIG':

```
$ export QT_QPA_EGLFS_KMS_CONFIG=~/kms.json
```

- Multi-screen support
  - The “hellowindow” example supports multiscreen on EGLFS. Before launching it, you have to export your KMS display configuration.
    - Fedora

```
$ cd /usr/lib/qt5/examples/opengl/hellowindow
$ ./hellowindow --multiscreen
```

    - Ubuntu

```
$ cd /usr/lib/arm-linux-gnueabihf/qt5/examples/opengl/hellowindow
$ ./hellowindow --multiscreen
```

- The QT5 Conematic Experience is awesome benchmark based on qt5 graphic effects. Unfortunately, current mali driver doesn't support device frequency. So, you have to boost the frequency manually.
  - Send the Qt5_CinematicExperience_rpi_1.0.tar.gz into your board and extract it. You can run the program through qmlscene.

```
$ tar xf Qt5_CinematicExperience_rpi_1.0.tar.gz
$ cd Qt5_CinematicExperience_rpi_1.0
$ qmlscene -platform eglfs Qt5_CinematicExperience.qml
```

RUN A QT5 EXAMPLE ON WAYLAND

Launch the Weston shell:

```
$ ./start_wayland.sh
```

Export the ‘QT_QPA_EGLFS_KMS_CONFIG’ and the ‘QT_QPA_PLATFORM’ environment variables for Wayland using:

```
$ export QT_QPA_PLATFORM=wayland-egl
$ export QT_QPA_EGLFS_KMS_CONFIG=~/kms.json
```

Run an example from the Weston terminal:

- Fedora

```
$ cd /usr/lib/qt5/examples/opengl/cube
$ ./cube
```

- Ubuntu

```
$ cd /usr/lib/arm-linux-gnueabihf/qt5/examples/opengl/cube
$ ./cube
```

LEGAL INFORMATION