Rockchip Linux Qt WebEngine Instruction

ID: RK-SM-YF-324

Release Version: V1.0.0

Release Date: 2020-02-06

Security Level: □Top-Secret □Secret □Internal ■Public

Fuzhou Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document mainly introduces the usage of Rockchip Linux Qt WebEngine, aiming to help engineers get started with Qt WebEngine development and debugging methods faster.

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers
Software development engineers

Chipset Support

Chipset	Buildroot	Debian	Yocto
RK3288	Y	Y	N
RK3326/PX30	Y	Y	N
RK3328	Y	N	N
RK3399	Y	Y	N
RK3399Pro	Y	Y	N

Revision History

Date	Version	Author	Revision History
2020-02-06	V1.0.0	Caesar Wang	Initial version

Contents

1 Qt WebEngine

1.1 Overview

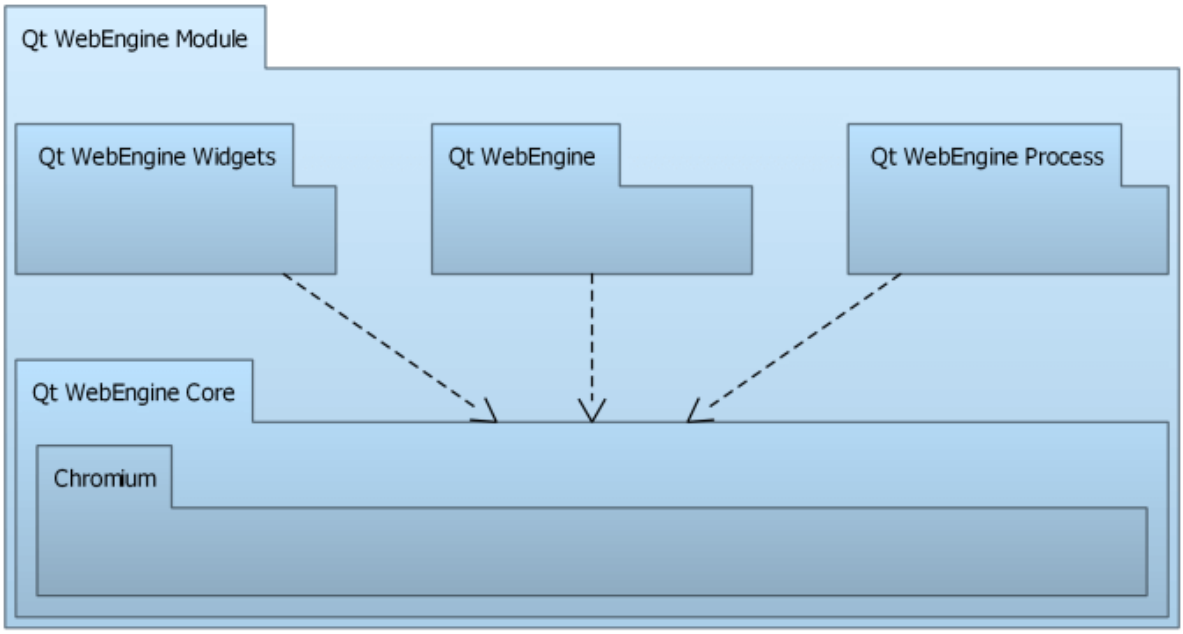
The Qt WebEngine module provides a web browser engine built for embedding web content into applications without using a local browser. The engine features QML types and C++ classes for rendering HTML, XHTML, and SVG documents using CSS and programmed with JavaScript.

This document mainly introduces the embedded usage of Qt WebEngine in Buildroot and Debian systems, and how to call the multivideo hard decode process from ffmpeg/mpp/vpu.

1.2 Architecture

The functions of Qt WebEngine can be divided into the following modules:

Qt WebEngine Widgets Module: for creating web application modules based on widget
Qt WebEngine Module: for creating web application modules based on Qt Quick.
Qt WebEngine Core Module: it is the core module of Qt WebEngine, used for interacting with Chromium
Please refer to the following figure for details:



For more details, please refer to Qt official documents.

2 Different Systems Support

2.1 Buildroot

Buildroot supported by Rockchip Linux is developed based on 2018.02-rc3, and the Qt WebEngine is developed based on version 5.12.2.
If the Qt WebEngine should be supported in Buildroot, the config (BR2_PACKAGE_QT5WEBENGINE) and related configurations should be enabled. Currently the latest released SDKs all support this function, but the related configuration is disabled. So you need to open the following configuration:

```
#include "chromium.config"
```

If the video hard decode is implemented by ffmpeg, the following config should be enabled:

```
#include "video_ffmpeg.config"
```

For example, if Qt WebEngine function should be supported in RK3399, the following configuration should be enabled:

```
diff --git a/configs/rockchip_rk3399_defconfig
b/configs/rockchip_rk3399_defconfig
index dc84293..db6e177 100644
--- a/configs/rockchip_rk3399_defconfig
+++ b/configs/rockchip_rk3399_defconfig
@@ -1,8 +1,11 @@
 #include "rk3399_arm64.config"
 #include "base.config"
 #include "base_extra.config"
+#include "chromium.config"
 #include "gpu.config"
 #include "display.config"
+#include "video_ffmpeg.config"
 #include "video_mpp.config"
```

After compiling, the source code of Qt WebEngine is in the buildroot$ vi output/rockchip_rk3399/build/qt5webengine-5.12.2/ directory.
Or refer to the official QT source code.

About Qt WebEngine configuration settings of Buildroot, please refer to: buildroot$ vi package/qt5/qt5webengine/ directory,
Test Demo is in the package/rockchip/rockchip_test/src/rockchip_test/chromium/ directory

```
#cat test_simplebrowser.sh
...
cd /usr/lib/qt/examples/webenginewidgets/simplebrowser
./simplebrowser --no-sandbox --disable-es3-gl-context
#./simplebrowser --no-sandbox --disable-es3-gl-context https://www.baidu.com
#./simplebrowser --no-sandbox --disable-es3-gl-context
"file:///oem/SampleVideo_1280x720_5mb.mp4"
#./simplebrowser --no-sandbox --disable-es3-gl-context --enable-logging --v=5
"file:///oem/SampleVideo_1280x720_5mb.mp4"
..
```

The Opengles has been specified during Buildroot compilation, so you only need to fix the previous context issues, add the parameter --disable-es3-gl-context to allow chromium to use es2 during booting. Because the hard video decode function of chromium needs to access some device nodes, --no-sandbox parameter is needed when starting simplebrowser .

2.2 Debian

The official Qt of Rockchip Linux Debian 9 (stretch) is 5.7 which does not support WebEngine. The Qt packages used in the SDK are updated based on the buster source with the version of 5.11, so you need to modify the source during manual installation.

```
export DISPLAY=:0
su linaro -c "xhost +"
echo "deb http://ftp.cn.debian.org/debian buster main" >> /etc/apt/sources.list
apt-get update
apt-get install qtwebengine5-examples
/usr/lib/aarch64-linux-gnu/qt5/examples/webengine/minimal/minnimal --no-sandbox
```

After testing, recover the modification of /etc/apt/sources.list.

If you are porting official Qt WebEngine compilation, like 5.12.2, please note the following items:

- The official Qt WebEngine of Debian is compiled with xcb glx (Rockchip platform is implemented by mesa software) and xcb egl (which is implemented by mali gpu), and glx is preferred. Which needs to use egl through environment variable setting, otherwise it should be rendered by software (the same as turning off RGA effect);

```
export QT_XCB_GL_INTEGRATION=xcb_egl
```

- When "Cannot find EGLConfig, returning null config" appears after choosing egl, for the xcb of Qt is implementing, the default renderable type is set to opengl (it is not supported by mali library). You can refer to the following method to solve this problem:
  Change the default setting to QSurfaceFormat::OpenGLES on the side of application through QSurfaceFormat's setRenderableType (please google the keyword for details),or refer to the official demo: qt5base-5.12.2# vi examples/opengl/computegles31/main.cpp, or remove the opengl (only opengles) in the configuration and recompile Qt. Note that the opengles is enabled by default.

```
qt5base-5.12.2# git diff src/platformsupport/eglconvenience/qeglconvenience.cpp
diff --git a/src/platformsupport/eglconvenience/qeglconvenience.cpp
b/src/platformsupport/eglconvenience/qeglconvenience.cpp
index 020d035..a4156cb 100644
--- a/src/platformsupport/eglconvenience/qeglconvenience.cpp
+++ b/src/platformsupport/eglconvenience/qeglconvenience.cpp
@@ -252,7 +252,7 @@ EGLConfig QEglConfigChooser::chooseConfig()
         break;
 #ifdef EGL_VERSION_1_4
     case QSurfaceFormat::DefaultRenderableType:
-#ifndef QT_NO_OPENGL
+#if 0//ndef QT_NO_OPENGL
         if (QOpenGLContext::openGLModuleType() == QOpenGLContext::LibGL)
             configureAttributes.append(EGL_OPENGL_BIT);
         else
```

- After the modification, you will see "eglCreateContext failed with error EGL_BAD_CONTEXT". The reason is that Qt has created the context of es2 by default, but it is packaged into es3 in the chromium. You can refer to the following method to solve:
  Add the parameter --disable-es3-gl-context when booting to let chromium use es2 or change the default setting to 3 through QSurfaceFormat setVersion on the side of application(please google the keyword for details) or refer to the official demo:
  qt5base-5.12.2 # vi examples / opengl / computegles31 / main.cpp or modify the qt xcb plugin to let Qt create an es3 context:

```
qt5base-5.12.2# git diff
src/plugins/platforms/xcb/gl_integrations/xcb_egl/qxcbeglintegration.cpp
diff --git
a/src/plugins/platforms/xcb/gl_integrations/xcb_egl/qxcbeglintegration.cpp
b/src/plugins/platforms/xcb/gl_integrations/xcb_egl/qxcbeglintegration.cpp
index fe18bc2..bb8c72c 100644
--- a/src/plugins/platforms/xcb/gl_integrations/xcb_egl/qxcbeglintegration.cpp
+++ b/src/plugins/platforms/xcb/gl_integrations/xcb_egl/qxcbeglintegration.cpp
@@ QXcbWindow *QXcbEglIntegration::createWindow(QWindow *window) const
 QPlatformOpenGLContext *QXcbEglIntegration::createPlatformOpenGLContext
 (QOpenGLContext *context) const
 {
```

```
     QXcbScreen *screen = static_cast<QXcbScreen *>(context->screen()-
>handle());
-    QXcbEglContext *platformContext = new QXcbEglContext(screen-
>surfaceFormatFor(context->format()),
+
+    QSurfaceFormat format = screen->surfaceFormatFor(context->format());
+    format.setMajorVersion(3);
+
+    QXcbEglContext *platformContext = new QXcbEglContext(format,
                                                          context-
>shareHandle(),
                                                          eglDisplay(),
```

- Finally "Failed to initialize extensions" appears.
  It is because webengine links to some symbol tables in opengl (mesa) and opengles (mali) libraries at the same time, causing some to use mesa and some to use mali. This problem can be solved by adding libGLESv2.so library dependency when compiling the application, which will bind its symbol table first. When testing, you can directly modify the application and add dependencies with tools, such as: patchel --add-needed libGLESv2.so minimal.