Porting Tizen-IVI 3.0 to an ARM based SoC Platform

Damian Hobson-Garcia, IGEL Co., Ltd.
Current State of Affairs

- Intel architecture (x86) system
  - Tizen IVI 2.0alpha, Tizen IVI 3.0

- ARM architecture based system
  - Tizen IVI 2.0alpha (ivi-panda)
Test Platform

- **Renesas R-Car Gen2 series platform**
- **R-Car M2**
  - ARM Cortex A15 x2
- **R-Car H2**
  - ARM Cortex A15 x4, + ARM Cortex A7 x4 (option)
- **Graphics System**
  - 3D - Imagination Technologies PowerVR series
  - Display Unit
- **On board IP**
  - H/W video decode/encode
  - image processing
Agenda

• Objective
• Methodology
• Porting Tasks
  • Weston/Wayland Integration
  • WebKit Integration
  • GStreamer Integration
Objective

• Tizen IVI 3.0 on R-Car M2/H2

1. Standard Native Applications
   • Terminal program
   • Open GL/ES applications

2. Web
   • Browser and web applications

3. Multimedia
   • Video playback (1080p @ 30fps)
Methodology

• **Tizen IVI 3.0 milestone releases we used:**
  • M2-Sep (released Oct 11, 2013)
  • M2-EOY (released Jan 15, 2014)
  • M2-March2014 (released April 11, 2014)

• **Non-hardware dependant packages**
  • Rebuild for ARM instruction set

• **Hardware dependant packages**
  • Replace with R-Car M2/H2 versions
Getting Source Code and Workflow
Using Upstream Repos

- review.tizen.org
  - git source code repo

- download.tizen.org
  - rpm binary package repo

- gbs
  - Locally modified source code
  - Local binary package

- mic

Package Compilation

Image creation

File system image

Flash onto target system
Full local build

https://source.tizen.org
search: building tizen from scratch

review.tizen.org
git source code repo

Locally modified source code

Package Compilation

gbs
mic

Local binary package

Image creation

File system image

Flash onto target system

Package Compilation
Source Code and Build Preparation

• Get source code
  
  ```
  $ repo init -u review.tizen.org:scm/manifest -b tizen -m ivi.xml
  ```

• Overwrite `projects.xml` with milestone manifest file
  
  [URL](http://download.tizen.org/$(RELEASE_PATH)/builddata/manifest/)

[Diagram showing structure of the `.repo/` directory with links to `manifests/`, `ivi/`, `metadata.xml`, `prebuilt.xml`, and `projects.xml`.]
Build Preparation (cont.)

• Use prebuilt ARM toolchain from Tizen mobile branch

```
-<project name="pre-built/toolchain-arm" ... revision="tizen-ivi"/>
+<project name="pre-built/toolchain-arm" ... revision="tizen"/>
```

$ repo sync
Porting Tasks

- Recompile packages (roughly 800)
- Wayland/Weston (windows system) backend
  - Use PowerVR driver instead of Mesa
- Web Applications
  - Implement WaylandBufferManager
- Multimedia Acceleration Video Playback
  - 0 – copy video stream processing
Replacing the Mesa driver for Wayland/Weston
Wayland/Weston Overview

client/server based windowing system

**client process**

Client Application

**server process**

Weston Compositor

---

**client**: draws application content

**server**: composites one or more client windows to create output screen
Wayland/Weston with Mesa

Wayland EGL extension lets Mesa use Wayland buffers
Wayland/Weston on R-Car M2/H2

Client process

server process

- Client Application
- Weston Compositor
- wl_kms
- PowerVR OpenGL driver
- libgbm
- libkms
- drm/kms driver
- GPU driver

almost same as wl_drm
but with libkms back end
uses generic dumb buffer backend

user space
kernel

Wayland protocol
buffer sharing Wayland prot.
ioctl
GPU API

generic library
proprietary library
other functional unit
Replacing Mesa

Replacement libraries must

- Implement EGL_WL_bind_wayland_display EGL extension
  http://cgit.freedesktop.org/mesa/mesa/tree/docs/specs/WL_bind_wayland_display.spec
- Provide
  - libgbm – Access to dri/drm device
  - libdrm/libkms – for access to memory buffers (we use DUMB buffers)
  - buffer sharing interface – (similar to Mesa wl_drm)
  - libgbm backend should match buffer sharing interface
Replacing Mesa on Tizen

1. remove mesa library
   
   $ rm -r platform/upstream/mesa

2. edit build.conf (build settings file)
   
   -%define with_mesa=1
   ... 
   +Substitute: pkgconfig(gl)
   +Substitute: mesa-devel pkgconfig(gles20)
   ...
   Macros
   -%with_mesa=1

3. build the system
   
   $ gbs build -A armv7l

   (for full build command line see http://source.tizen.org “building Tizen from scratch”)
Objective

- **Tizen IVI 3.0 on R-Car M2/H2**
  1. **Standard Applications**
     - Terminal program
     - Open GLES applications
  2. **Web**
     - Browser and web applications
  3. **Multimedia**
     - Video playback (1080p @ 30fps)
WebKit2 and WaylandBufferManager
Simple client-server configuration

- **client process**
  - WebKit UI Process

- **server process**
  - Weston Compositor
    - wl_kms
    - libgbm
    - libkms
  - PowerVR OpenGL driver
  - Other functional unit
  - GPU driver

**Wayland protocol**
- buffer sharing Wayland prot.
- ioctl
- GPU API

**Kernel**
- drm/kms driver

**User space**
- generic library
- proprietary library
- other functional unit
WebKit2 client-client/server-server configuration

**client process**

WebKit Web Process

**client/server process**

WebKit UI Process

- `wl_kms`
- `libkms`

**server process**

Weston Compositor

- `wl_kms`
- `libgbm`

GPU driver

Wayland protocol

buffer sharing Wayland protocol

`ioctl`

GPU API

user space

kernel

generic library

proprietary library

other functional unit

TIZEN

**DEVELOPER CONFERENCE 2016 SAN FRANCISCO**

22
Webkit2 Buffer Allocation

**WaylandDisplay (class):**
Update to use `wl_kms` instead of `wl_drm`

**WaylandKmsBufferManager (class):**
Implementation of `WaylandBufferManager` interface

Wayland protocol
buffer sharing Wayland prot.
ioctl

generic library  other functional unit

drm/kms driver
WaylandBufferManager Interface

- **Interface for allocating/locking shareable buffers (e.g., kms_bo)**
  - `allocateBO` returns `handleId`.
  - `*handle` is pointer to shareable fd (i.e., flinked fd, or DMABuf handle)
  - `query` to get buffer virtual address

```cpp
class WaylandBufferManager {
    allocateBO(w, h, stride, size, align, *handle);
    lockSurface(handleId);
    unlockSurface(handleId);
    freeBO(handleId);
    query(handleId, **addr);
}
```
Objective

- **Tizen IVI 3.0 on R-Car M2/H2**
  1. **Standard Applications**
     - Terminal program
     - Open GLES applications
  2. **Web**
     - Browser and web applications
  3. **Multimedia**
     - Video playback (1080p @ 30fps)
Using GStreamer with Tizen IVI 3.0
GStreamer

- Encode, decode, capture and display multimedia data
- Make a pipeline of components to do what you want

Example GStreamer pipeline:

- video demuxer
- audio decoder
- sample player
- video decoder
- frame renderer

 naar to speaker
t naar to screen
Video Decode on R-Car M2/H2 on Tizen IVI 3.0

- **Audio pipeline**
  - Software decode for now

- **Video decode**
  - Use gst-omx to bridge GStreamer to OpenMAX IL component

- **Color conversion/scaling**
  - Use hardware accelerated color conversion/scaling module

- **Display**
  - Use waylandsink to display via Weston compositor
GStreamer H/W accelerated video decode

client process

GStreamer Application

GStreamer Plugins

gst-omx

vspfilter (color conv./scaler)

waylandsink

OpenMAX IL Video decoder

H/W video decoder

H/W color conv./scaling

server process

Weston Compositor

GPU hardware

Wayland protocol  API call

full custom  as-is upstream component  customized component  Reneas proprietary library
Waylandsink customization

- H/W color conversion requires physically contiguous buffers
  - Waylandsink allocates non-contiguous shared memory buffers
  - Need to add extra memcpy()s into pipeline.

- Buffers allocated from kms bo are physically contiguous (on our system)
  - Use the same method as with WebKit to allocate and share graphics buffers
Waylandsink customized for libkms usage

Allocated kms dumb buffers used for H/W color conversion.

No memcpy()s required between video decode and screen display.
Objective

• Tizen IVI 3.0 on R-Car M2/H2

1. Standard Applications
   • Terminal program
   • Open GLES applications

2. Web
   • Browser and web applications

3. Multimedia
   • Video playback (1080p @ 30fps)
What we learned - review

- **Building**
  - Use mainifest.xml from milestone release on download.tizen.org
  - Use mobile toolchain for ARM
- **Weston/Wayland**
  - Need support for EGL_WL_bind_wayland_display in Open GL/ES driver
  - Can use libkms dumb buffers
- **WebKit**
  - Implement WaylandBufferManager; update WaylandDisplay
- **Multimedia playback**
  - Use libkms and Wayland buffer sharing to implement 0-copy processing with physically contiguous memory buffers
Thank you.

Questions?
Links

- **Building Tizen from scratch**

- **EGL_WL_bind_wayland_display EGL extension**

- **Renesas R-Car series platforms**
  - http://am.renesas.com/applications/automotive/cis/cis_highend/