User’s Manual

CONFIDENTIAL

# Linux Interface Specification Wayland

## User’s Manual: Software

R-Car H3/M3/M3N/E3 Series

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev.3.1.1 2021.11

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

## Trademark

- Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
- Windows and Windows Media are registered trademarks of Microsoft Corporation in the United States and other countries.
- Other company names and product names mentioned herein are registered trademarks or trademarks of their respective owners.
- Registered trademark and trademark symbols (® and ™) are omitted in this document

# How to Use This Manual

- **[Readers]**

  This manual is intended for engineers who develop products which use the R-Car H3/M3/M3N/E3 processor.

- **[Purpose]**

  This manual is intended to give users an understanding of the functions of the R-Car H3/M3/M3N/E3 processor device driver and to serve as a reference for developing hardware and software for systems that use this Renesas Wayland/Weston system.

- **[How to Read This Manual]**

  It is assumed that the readers of this manual have general knowledge in the fields of electrical

  — engineering, logic circuits, microcontrollers, and Linux.

  → Read this manual in the order of the CONTENTS.

  — To understand the functions of a multimedia processor for R-Car H3/M3/M3N/E3

  → See the R-Car H3/M3/M3N/E3 User’s Manual.

  — To know the electrical specifications of the multimedia processor for R-Car H3/M3/M3N/E3

  → See the R-Car H3/M3/M3N/E3 Data Sheet.

- **[Conventions]**

  The following symbols are used in this manual.

  Data significance: Higher digits on the left and lower digits on the right

  **Note**: Footnote for item marked with Note in the text

  **Caution**: Information requiring particular attention

  **Remark**: Supplementary information

  Numeric representation: Binary ... ××××, 0b××××, or ××××B

  Decimal ... ××××

  Hexadecimal ... 0x×××× or ××××H

  Data type: Double word ... 64 bits

  Word … 32 bits

  Half word ... 16 bits

  Byte ... 8 bits

# Table of Contents

# 1. Overview

## 1.1. Overview

This guide explains overview of Wayland / Weston of Renesas Linux Solution for R-Car Series, 3rd Generation. Renesas Wayland using OpenGL API to control GPU for rendering purpose which is named "gl-renderer".

### 1.1.1. Outline of R-Car 3rd Generation window manager

Wayland / Weston is used as window manager of R-Car Series, 3rd Generation Linux Solution. Wayland / Weston provided by Wayland community, and Renesas customizes them to match the R-Car H3/M3/M3N/E3.

In standard of Wayland solution, GPU is used for both drawing and composing. Please refer to figure 1.

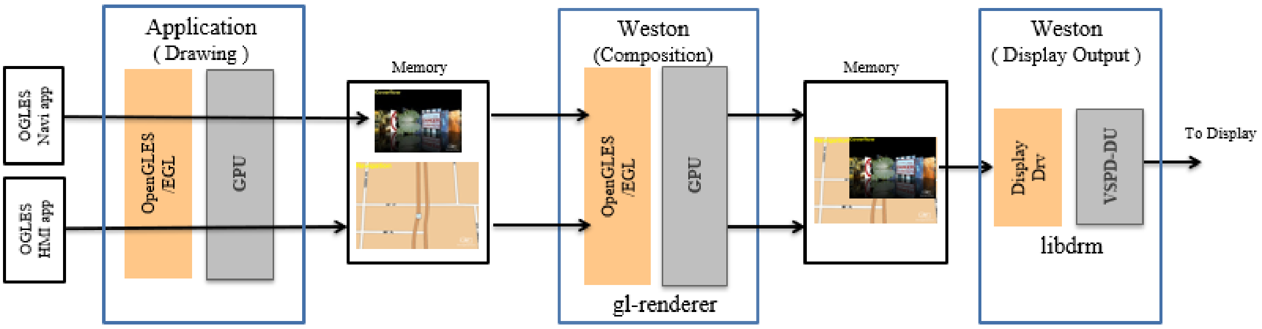


**Figure 1 Wayland/Weston in Renesas R-Car Series**

### 1.1.2. About Wayland/Weston

Wayland is just a protocol to control Wayland Server. Wayland server controls window location (x, y and z order), display image composition and output display. Weston is reference Wayland Server provided by Wayland community. About Standard features of Wayland / Weston, please refer to the official web site for more details. https://wayland.freedesktop.org/

## 1.2. Reference

### 1.2.1. Standard

About the standard Wayland protocol specifications, please refer to xml files in https://wayland.freedesktop.org/releases/wayland-1.18.0.tar.xz and https://wayland.freedesktop.org/releases/weston-8.0.0.tar.xz.

### 1.2.2. Related Documents

The following table shows the documents related to Renesas Wayland solution.

Table 1 Related Documents

| No. | Name | Provider |
|---|---|---|
| 1 | Linux Interface Specification Yocto recipe Start-Up Guide | Renesas Electronics Corporation |
| 2 | R-Car Series, 3rd Generation OpenGL ES Library for Linux User’s Manual | Renesas Electronics Corporation |
| 3 | Linux Interface Specification Device Driver Display | Renesas Electronics Corporation |

## 1.3. Function

The following table shows the supporting functions of gl-renderer.

**Table 2 Function table of gl-renderer**

| Functions/renderers | gl-renderer (GPU) |
|---|---|
| Resolution | Depend on GPU<br>(Max 8192 x 8192) |
| Composition | ✓ |
| Alpha Blend (pixel or global) | ✓ |
| Alpha Blend (pixel and global) | ✓ |
| Rotation | ✓ |
| Scaling | ✓ |
| Clipping | ✓ |
| Screen shot (renderer output image) | ✓ |
| Support Format (RGB) | XRGB8888/ARGB8888 |
| Support Format (YUV) | NV12/NV16/YUV420 |
| Close Animation (Provided by Weston desktop shell) | ✓<br>(Fade/Zoom) |
| Open Animation | ✓<br>(Fade/Zoom) |
| Multi Display | ✓ |

# 2. Terminology

The following table shows the terminology related to Renesas Wayland solution.

**Table 3 Related Documents**

| Terms | Explanation |
|---|---|
| BRU | Blend engine unit |
| DRM | Direct Rendering Manager |
| KMS | Kernel Mode Setting |
| RPF | Read pixel formatter |
| UDS | Up down scaler |
| WPF | Write pixel formatter |

# 3. Operating Environment

This section shows Environment of Hardware / Software of Renesas Wayland / Weston.

## 3.1. Hardware Environment

**Table 4 Hardware**

| No. | Name | Provider |
|---|---|---|
| 1 | LSI:<br>- R-Car H3<br>- R-Car M3<br>- R-Car M3N<br>- R-Car E3 | Renesas Electronics Corporation |
| 2 | Board:<br>- R-Car H3/M3/M3N-SiP System Evaluation Board Salvator-X/XS<br>- R-Car E3-SiP System Evaluation Board Ebisu | Renesas Electronics Corporation |

## 3.2. Software Environment

**Table 5 Software**

| No. | Name | Provider |
|---|---|---|
| 1 | R-Car H3/M3/M3N/E3 Series Yocto recipe package | Renesas Electronics Corporation |
| 2 | Graphics Package<br>- R8A7795 GX6650 OpenGL ES3.1 Library for Linux<br>- R8A7796 GX6250 OpenGL ES3.1 Library for Linux<br>- R8A77965 GE7800 OpenGL ES Library for Linux<br>- R8A77990 GE8300 OpenGL ES Library for Linux | Renesas Electronics Corporation |
| 3 | Wayland/Weston main part<br>- Wayland v1.18.0<br>- Weston v8.0.0<br>Renesas Wayland solution uses Desktop Shell provided by Weston. | http://wayland.freedesktop.org/ |

## 3.3. Configuration of Renesas Wayland Solution

Figure 2 shows the overview configuration of Renesas Wayland solution.

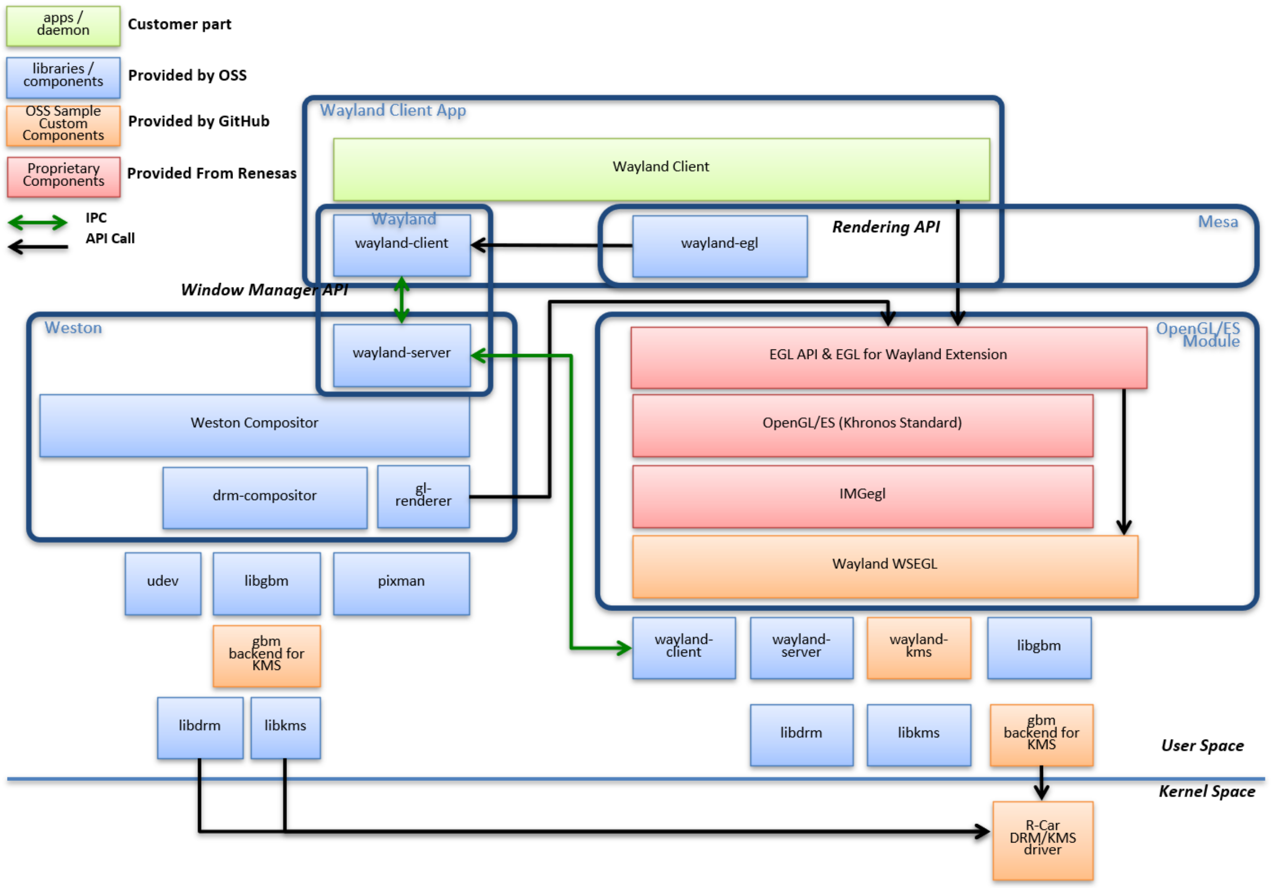


**Figure 2 Overview Configuration of Renesas Wayland Solution**

Figure 3 shows a system configuration and overview of drawing with GSX OpenGL ES Library (OpenGL ES 3.2 Library, EGL 1.4 Library, WSEGL module and GSX user module), it accesses to hardware via GSX kernel module and Linux Display Driver for GSX. GSX kernel module creates various drawing resources for GSX such as a draw command according to user application. GSX performs drawing processing using these drawing resources. The drawing processing in this software is performed by GSX which executes these drawing resources.

WSEGL is a module which depend on window system used by a system, there are two types of them, one is Wayland WSEGL, provided as OSS package.

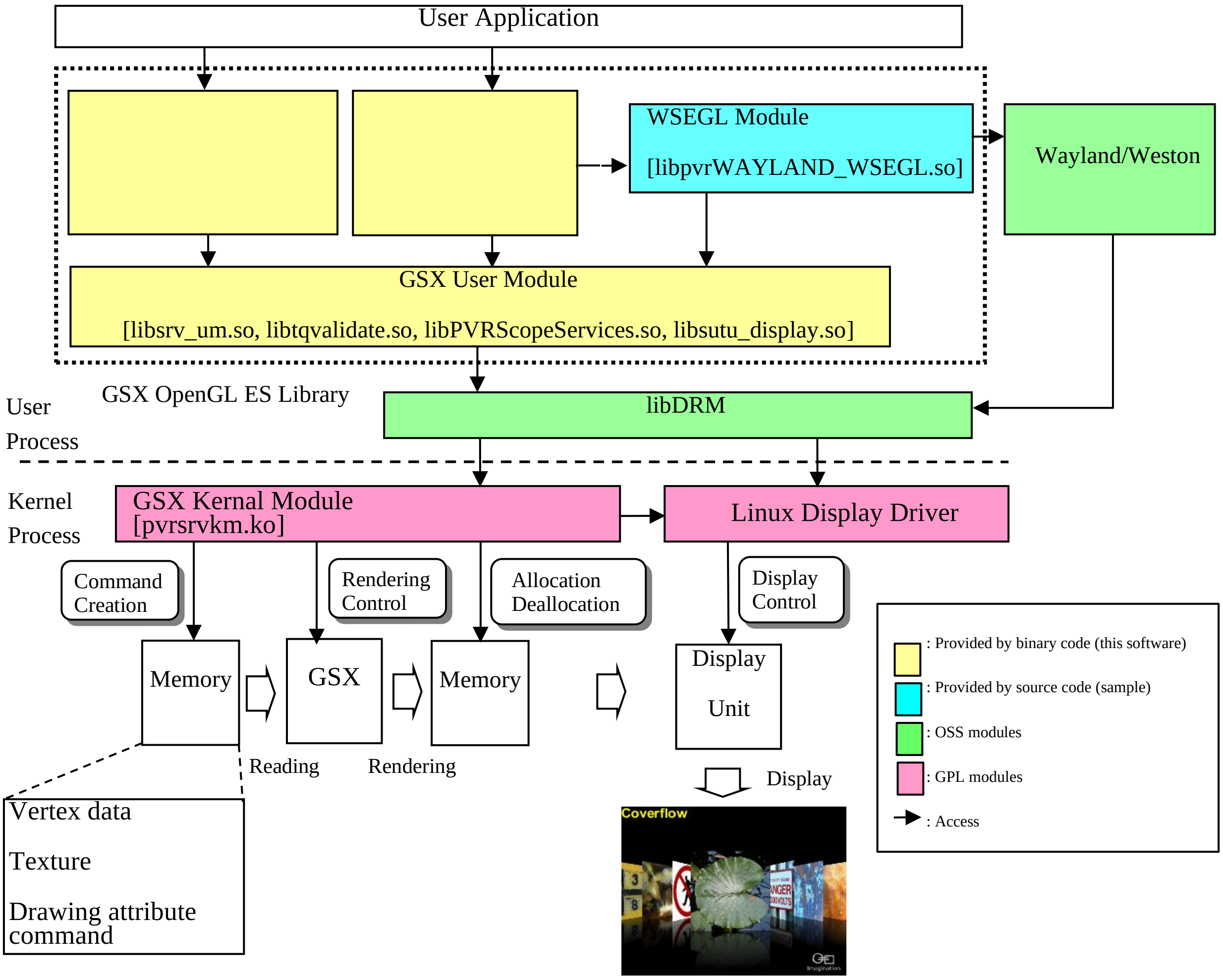


**Figure 3 System Configuration and Overview of Drawing Using This Software**

### 3.3.1. Software component List

Following table shows a list of Renesas Wayland / Weston components.

**Table 6 Software component list**

| No. | Module | Provider | Library | Description |
|---|---|---|---|---|
| 1 | Wayland-client | OSS | libWayland-client.so | Client protocol |
| 2 | Wayland-server | OSS | libWayland-server.so | Display server protocol |
| 3 | Wayland extension protocol (Wayland-kms) | OSS | libWayland-kms.so | The protocol of wl_buffer for KMS BO |
| 4 | Wayland Compositor | OSS | Weston | Wayland reference compositor |
| 5 | drm-compositor | OSS | drm-backend.so | Weston DRM Backend |
| 6 | gl-renderer | OSS | gl-renderer.so | Weston GPU rendering library |
| 7 | udev/evdev | OSS | libudev.so | The device manager for the Linux kernel. Manage event of input device (e.g., USB, keyboard, touchscreen etc.) |
| 8 | Pixman | OSS | libpixman-1.so | Graphics 2D software |
| 9 | Wayland-egl | OSS | libwayland-egl.so | Wayland-egl extracted from Mesa |
| 10 | libgbm | OSS | libgbm.so | Generic Buffer Management |
| 11 | libdrm[user space]<br>libkms[user space] | OSS | libdrm.so<br>libkms.so | User space library for drm |
| 12 | R-Car DRM/KMS driver | OSS | (included in kernel) | Kernel-level support for the Direct Rendering Infrastructure (DRI) for R-Car |
| 13 | EGL<br>[Khronos Standard] | IMG | libEGL.so | The standard (Khronos) for EGL APIs.<br>*Include EGL Wayland Extension |
| 14 | OpenGL/ES<br>[Khronos Standard] | IMG | libGLESv2.so | OpenGL ES Library - The standard (Khronos) for Embedded 3D Graphics |
| 15 | IMGegl | IMG | libIMGegl.so | IMG EGL backend |
| 16 | WSEGL Wayland<br>(wayland-wsegl) | OSS | libpvrWAYLAND_WSEGL.so | WSEGL on Wayland/Weston Implemented Server/Client mode |

### 3.3.2. Renesas provides OSS custom component

Following list shows the components which Renesas provides via Github for Renesas Wayland Solution.

**Table 7 GitHub list for Renesas Wayland Solution**

| No. | Module | Repository | Notes |
|---|---|---|---|
| 1 | wayland-kms | git://github.com/renesas-rcar/wayland-kms.git | wl_kms for KMS BO |
| 2 | libgbm | git://github.com/renesas-rcar/libgbm.git | libgbm with gbm backend for KMS |
| 3 | wayland-wsegl | git://github.com/renesas-rcar/wayland-wsegl.git | RGX WSEGL for Wayland |

**Note:** About drivers of kernel side, please refer to document of Linux Interface Specification Device Drive.

# 4. Functions

This section explains the additional funtions which Renesas customizes standard Wayland / Weston to match the R-Car H3/M3/M3N/E3. About standard features of Weston, please refer to the official web site https://wayland.freedesktop.org/.

## 4.1. Weston plane definitions

Standard Weston has three plane definitions. All surfaces are assigned to one of them. Each surface is handled according to assigned plane.

- ✓ Scanout Plane : One surface assigned to scanout plane is sent to libdrm and displayed it
- ✓ Sprite Plane : The surfaces assigned to sprite plane are sent to libdrm, and then composed by VSPD
- ✓ Primary Plane : The surfaces assigned to primary plane are composed by Weston composition IP(GPU)

■ **Scanout Plane Case**

Weston send the image buffer set "fullscreen" by Wayland I/F to DRM directly

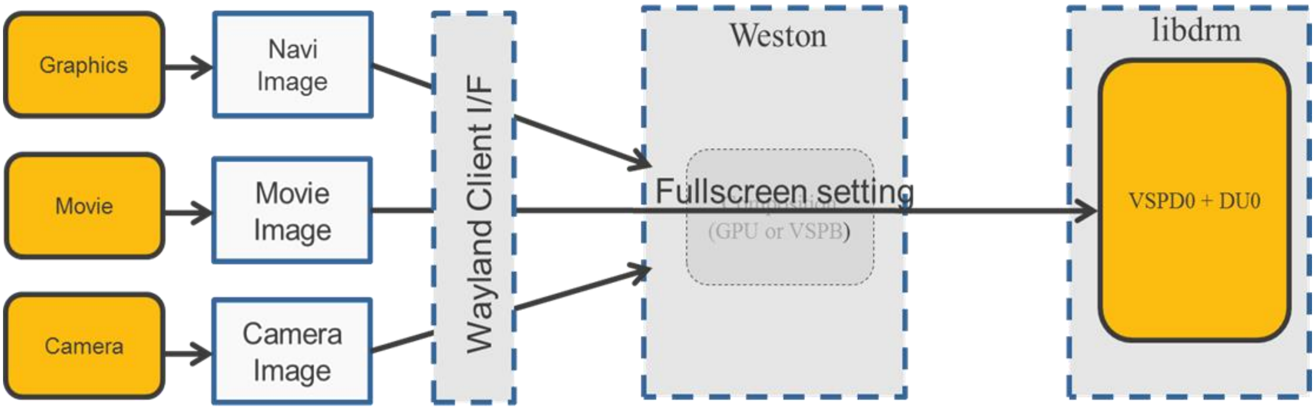


**Figure 4 Scanout plane**

Condition:

Image size = Display Size

No alpha blending

The top of surfaces

Set the "Full screen" via Wayland I/F (e.g.: zxdg_toplevel_v6_set_fullscreen)

- **Sprite Plane Case**

Weston sends the assigned image buffers to DRM directly. And then, they are composed by VSPD.

Default is disable in Weston.

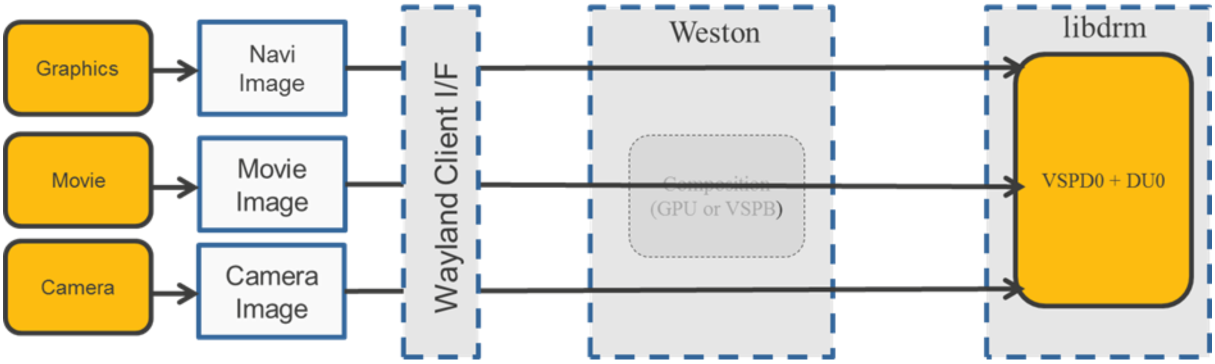


**Figure 5 Sprite plane**

Condition:

- It's required to enable Sprite Plane Flag 'sprites_are_broken' by code modification at user own risk. (In Weston 8.0, users don't need to modify the code)
- Key Control (Push Superkey + Shift + Space + V Key) toggles the enable or disable of Sprite Plane Flag 'sprites_are_broken'.

**Note:** Precautions for use sprite plane

If user want to use the sprite plane, user needs to consider whether or not fit to assume the use case. Because sprite plane access to libdrm directly.

1) The number of overlay planes depends on the VSPD specification. See display driver manuals for details.
2) No support sliding surface between displays.
3) Application needs four buffers. (Typically, 3 buffer)
4) Supported buffer type is zwp_linux_dmabuf_v1 only.

For more detail of other sprite plane conditions, please check the following Weston source code.

- drm_assign_planes() and drm_output_prepare_overlay_view() of " state-propose.c" in the Weston.

- **Primary Plane Case**

The surfaces assigned to Primary Plane are composed by Weston composition IP (GPU).

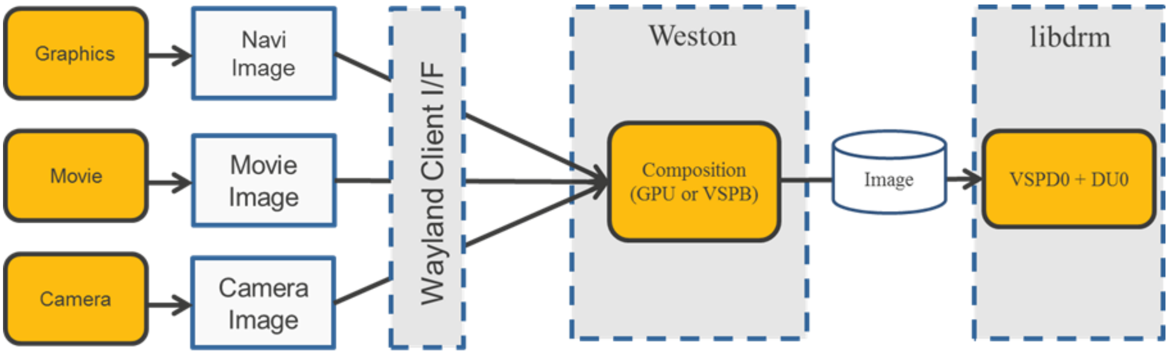


**Figure 6 Primary Plane**

Condition:

Not assigned to Scanout plane / Sprite plane

The surfaces assigned to Primary Plane are composed by Weston composition IP (GPU).

## 4.2. Support Buffer Type

Renesas Wayland / Weston supports three Wayland buffer types ”wl_shm”,”wl_kms” and “zwp_linux_dmabuf_v1”. The wl_kms is the Renesas original Wayland buffer type.

### 4.2.1. wl_shm

- Wayland shared memory buffer type.
- Based on UNIX shared memory.
- Memory copy required on the server side when used with hardware acceleration.

### 4.2.2. wl_kms

- KMS BO buffer type.
- Based on DMABUF and KMS BO, the standard Linux technology.
- No memory copy required.

wl_kms is original protocol of Renesas. If user applications commit image with the wl_kms buffer to Wayland-server, please allocate the CMA buffer by using libkms, and then, get wl_buffer handle using Wayland-kms like below.

- In case of using KMS_BO

Step 1. Create buffer using libkms/kms_bo_create().

Step 2. Get GEM handle and pitch using libkms/kms_bo_get_prop().

Step 3. Convert GEM handle to dmabuf fd using libdrm/drmPrimeHandleToFD().

Step 4. Get wl_buffer handle using Wayland-kms/wl_kms_create_buffer().

**Note:** wl_kms is supported to keep compatibility with R-Car Series, 2nd Generation. It will be stopped supporting in future (Not R-Car Series, 3rd Generation). Therefore, we recommend to use "zwp_linux_dmabuf_v1" protocol provided by Wayland community instead of "wl_kms".

### 4.2.3. zwp_linux_dmabuf_v1

- DMABUF buffer type.
- Based on DMABUF, the standard Linux technology.
- No memory copy required.

The dmabuf is a trend buffer type of the standard Linux technology. The "zwp_linux_dmabuf_v1" is a Wayland buffer type with dmabuf, and it's supported since Wayland version 1.11.0.

**Note:** The buffer allocated by libkms is not supported.

## 4.3. Synchronization

Please refer to Appendix, article 6.6.1.6

# 5. Integration

## 5.1. Directory Configuration

About all Weston Directory Configuration, refer to https://wayland.freedesktop.org/.

## 5.2. Option Setting in weston.ini

### 5.2.1. Weston outputs to Analog RGB and LVDS display even if they aren’t connected

When CONFIG_DRM_FBDEV_EMULATION is enabled in linux kernel, Weston outputs to Analog RGB and LVDS display. If you don’t need their display, we suggest disabling them by using Weston.ini. Following content of weston.ini is sample description for disabling output to them.

Example command:

```
# mkdir /etc/xdg/weston

# vi /etc/xdg/weston/weston.ini
```

Sample of contents for weston.ini:

```
[output]
name=LVDS-1
mode=off
[output]
name=VGA-1
mode=off
```

Then, if you make or modify weston.ini when running Weston, please restart Weston using following command.

```
# systemctl restart weston@root
```

### 5.2.2 repaint-window

Set the approximate length of the repaint window in milliseconds. For more details, please refer to “weston.ini.man” in Weston.

Default value in reference Weston is 7ms, but Renesas set the value to ‘34ms’ by weston.ini provided by Renesas Wayland solution. The value of ‘34ms’ means that “select old algorithm”. This value on the presupposition that 60fps (16.6ms) and 30fps (33.3ms) of display refresh rate.

```
[core]

repaint-window=34
```

When user use the 15fps (66.6ms) of display refresh rate, user need set “repaint-window=67”.

The algorithm of Weston repaint scheduling is explained in the following web site.

https://www.collabora.com/about-us/blog/2015/02/12/Weston-repaint-scheduling/

# 6. Appendix

This section describes some tips and additional information of Wayland / Weston. Please refer to this information as a reference for your system.

## 6.1. Wayland-backend feature

The "Wayland-backend" which uses only GPU (gl-renderer) can connect to drm-backend.

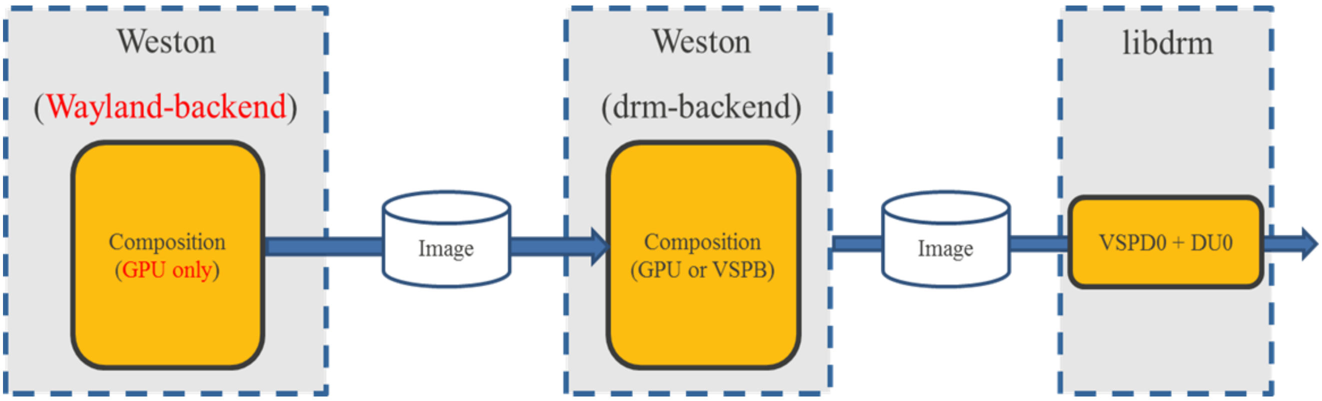


**Figure 7 Use two Westons (Wayland-backend and drm-backend)**

- How to launch the Wayland-backend

Please use following command to launch Wayland-backend.

```
# weston    ( or # weston -Bdrm-backend.so -Swayland-0) (*)

# weston -Bwayland-backend.so -Swayland-1 --display=wayland-0


(*) If you use Yocto Recipe Package from Renesas, The Weston is booted automatically by systemd.
    Therefore, you can skip it.
```

In addition, Wayland backend can be set by the width, height and fullscreen options.

```
# weston -Bwayland-backend.so -Swayland-1 --display=wayland-0 --width=800  ⁻height=480

# weston -Bwayland-backend.so -Swayland-1 --display=wayland-0 --fullscreen
```

- How to select by Application.

To select the backend by application, modify the argument in wl_display_connect ().

struct wl_display *wl_display_connect(const char *name)

[Example]

```
wl_display_connect(“Wayland-0”);        // connect to drm-backend
wl_display_connect(NULL);               // connect to drm-backend
wl_display_connect(“Wayland-1”);        // connect to Wayland-backend
```

As “name” is the file name of IPC Socket that is located in folder set as “XDG_RUNTIME_DIR”.

## 6.2. How to set the composite priority

If the applications which use GPU affect the GPU composition by Weston, the priority of composition can be set to high by modifying gl-renderer.

<File location/file name>

src/gl-renderer.c

<Function name>

gl_renderer_setup()

[Before modification]

```
    static const EGLint context_attribs[] = {
        EGL_CONTEXT_CLIENT_VERSION, 2,
        EGL_NONE
  };
```

[After modification]

```
    static const EGLint context_attribs[] = {
        EGL_CONTEXT_CLIENT_VERSION, 2,
        EGL_CONTEXT_PRIORITY_LEVEL_IMG, EGL_CONTEXT_PRIORITY_HIGH_IMG,
        EGL_NONE
  };
```

**Note:** On Weston 8.0, users don’t need this modification.

## 6.3. Connect the input device to the board

Weston requires the input device for launch in the default setteing of weston.ini. So, you need to connect the input device (ex. usb-mouse or usb-keyboard) to board before booting. If user want to launch without the input device, please set to "require-input=false" in weston.ini. For details, please refer to /man/weston.ini.man in Weston.

**Note:** From kernel 5.4, there is a patch that maps the DIP switches and push buttons on Salvator-X/XS to SoC GPIOs for testing purpose. It makes this log show up when booting

```
input: keys as /devices/platform/keys/input/input0
```

which will treat Weston that input device is available so the launcher can pass without read devices plugged in.

## 6.4. How to set video mode

User can set the video mode in weston.ini for the output. The argument mode can be one of the words "off" to turn the output off, and It can also be a resolution as width x height, or a detailed mode line as below.

**mode=dotclock hdisp hsyncstart hsyncend htotal vdisp vsyncstart vsyncend vtotal hflag vflag**

The below settings are an example of a video mode. For more details, please refer to /man/weston-drm.man in Weston.

```
[output]
name=VGA-1
mode=35.00  800 832 904 1008  600 603 607 623 -hsync +vsync

[output]
name=HDMI-A-1
mode=1920x1080@60

[output]
name=HDMI-A-2
mode=off
```

**Note:** Renesas backported the following patch. This patch tries to preserve existing output routing of display driver.

https://github.com/wayland-project/weston/commit/75487c2560dfca033d8d23649914efb4199356dd

## 6.5. Multi display composition

Multi displays composition are sequentially processed in Weston. Because Weston is single thread process.

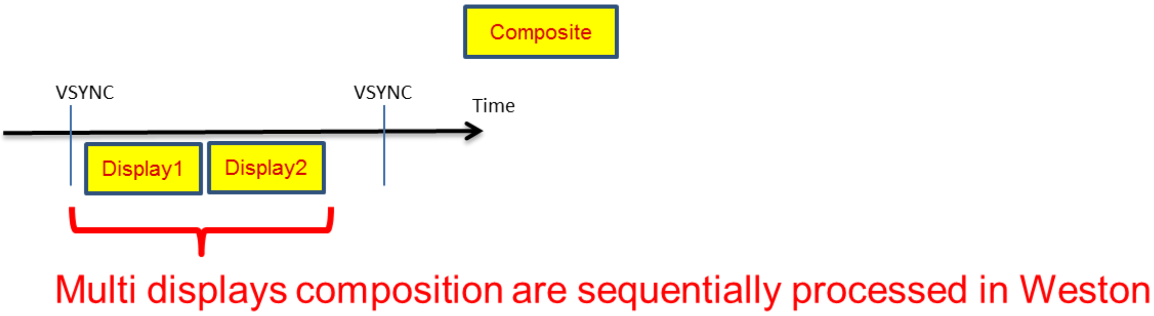

Composite
VSYNC
VSYNC
Time
Display1
Display2
Multi displays composition are sequentially processed in Weston

**Figure 8 Multi display composition**

## 6.6. Using EGL libraries

### 6.6.1 Note

| Section | List of EGL 1.4 Library Notes | H3 | M3 | M3N | E3 |
|---|---|---|---|---|---|
| 6.6.1.1 | Window Resizing | √ | √ | √ | √ |
| 6.6.1.2 | Maximum Number of the Window Surfaces to be created | √ | √ | √ | √ |
| 6.6.1.3 | Specifying a Color Format of a Window Surface | √ | √ | √ | √ |
| 6.6.1.4 | Specifying the number of back buffers for window surface | √ | √ | √ | √ |
| 6.6.1.5 | Specifying Aggressive sync(wl_display_sync) mode | √ | √ | √ | √ |
| 6.6.1.6 | Synchronization | √ | √ | √ | √ |

#### 6.6.1.1 Window Resizing

| H3 | M3 | M3N | E3 |
|---|---|---|---|
| √ | √ | √ | √ |

Call the wl_egl_window_resize when window size is changed. To use this function, it is necessary to include “wayland-egl.h”.

#### 6.6.1.2 Maximum Number of the Window Surfaces to be created

| H3 | M3 | M3N | E3 |
|---|---|---|---|
| √ | √ | √ | √ |

The maximum number of the window surfaces depends on free memory size in CMA area.

#### 6.6.1.3 Specifying a Color Format of a Window Surface

| H3 | M3 | M3N | E3 |
|---|---|---|---|
| √ | √ | √ | √ |

Color format is specified by an argument“EGLConfig” of eglCreateWindowSurface. The EGLConfig is gotten by eglChooseConfig. A supported color format is ARGB8888 and XRGB8888.

```
EGLConfig configs[1];
EGLint cfg_attribs[] = { EGL_BUFFER_SIZE,      32,
                         EGL_SURFACE_TYPE, EGL_WINDOW_BIT,
                         ...
                         EGL_NONE };

eglChooseConfig(dpy, cfg_attribs, configs, 1, &config_count);
eglCreateWindowSurface(dpy, configs[0], NULL, NULL);
```

**Figure 9 Example of Creating a Window Surface of ARGB8888**

```
EGLConfig configs[1];
EGLint cfg_attribs[] = { EGL_BUFFER_SIZE,      24,
                         EGL_SURFACE_TYPE, EGL_WINDOW_BIT,
                         ...
                         EGL_NONE };

eglChooseConfig(dpy, cfg_attribs, configs, 1, &config_count);
eglCreateWindowSurface(dpy, configs[0], NULL, NULL);
```

**Figure 10 Example of Creating a Window Surface of XRGB8888**

#### 6.6.1.4 Specifying the number of back buffers for window surface

| H3 | M3 | M3N | E3 |
|---|---|---|---|
| √ | √ | √ | √ |

In the WSEGL software, user will be able to change to the number of buffer constitution consist of triple buffer constitution of default setting. The user can set the number of buffers to 2~4 in the environment variable as follow.

```
$ export WSEGL_NUM_BUFFERS=2
```

This environment value is possible set for each process. If user want to specify for each process, please run the application and set as follow.

```
$ WSEGL_NUM_BUFFERS=4 weston-simple-egl
```

Don't change this environment variable during the process is running.

#### 6.6.1.5 Specifying Aggressive sync(wl_display_sync) mode

| H3 | M3 | M3N | E3 |
|---|---|---|---|
| √ | √ | √ | √ |

Due to the specification of Weston, the surface may not be updated for event waiting status under certain conditions in the double buffer constitution. As the action in this software, by specifying to 1 in the environment variable as follow, the process of synchronization is requested, and then the event is coming out.

```
$ export WSEGL_ENABLE_AGGRESSIVE_SYNC=1
```

This phenomenon appears using scan-out plane in double buffer constitution at the drm-backend/gl-renderer. When user is not set the environment value, it does not make an aggressive request for display sync. When user set triple buffer constitution, this specifying is not requested.

#### 6.6.1.6 Synchronization

| H3 | M3 | M3N | E3 |
|---|---|---|---|
| √ | √ | √ | √ |

To realize synchronization between GPU-Rendering and Weston Composition, REL wayland solution uses "fence" function. The "fence" function can synchronize between two different processes using same buffer. However, it is deferent fence mechanism that depend by the Weston renderer selected by the user. Please refer below.

<Weston gl-renderer>

This solution to use the "inter-process GPU sync mechanism" of IMG DDK that can synchronize between two different processes using same dmabuf buffer. For example, the process(client-side) flushes any buffered graphics commands and sending dmabuf buffer to the process(server-side) without waiting for the HW to complete. When the rendering of the process(client-side) on the surface has not completed yet, the process(server-side) waits for the DDK's fence signal of buffer from the process(client-side).

By these functions, the following steps are processed on the wayland client and server side.

Step 1. Client-side:

eglSwapBuffers() calls wl_surface_commit() and returns to the caller.

[Notes] eglSwapBuffers() *not wait* to complete the GPU rendering.

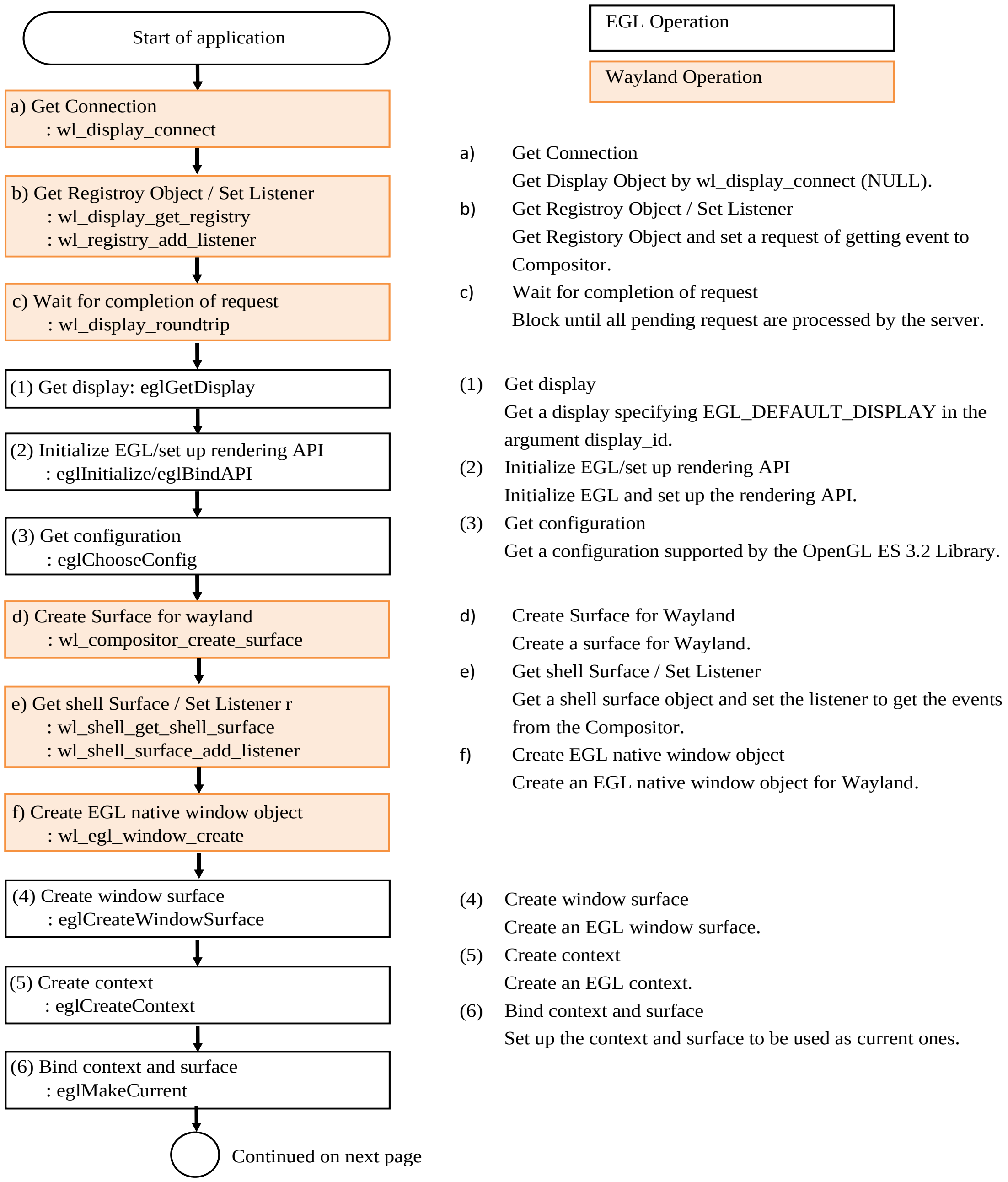
Step 2. Server-side:

server-side *wait* for the GPU rendering of the client side to finish before composite of server side

## 6.6.2 Drawing processing flow

### 6.6.2.1 Window Surface Drawing Processing Flow

Figure 11 shows the processing flow for drawing on a window surface.


Start of application
a) Get Connection
: wl_display_connect
b) Get Registroy Object / Set Listener
: wl_display_get_registry
: wl_registry_add_listener
c) Wait for completion of request
: wl_display_roundtrip
(1) Get display: eglGetDisplay
(2) Initialize EGL/set up rendering API
: eglInitialize/eglBindAPI
(3) Get configuration
: eglChooseConfig
d) Create Surface for wayland
: wl_compositor_create_surface
e) Get shell Surface / Set Listener r
: wl_shell_get_shell_surface
: wl_shell_surface_add_listener
f) Create EGL native window object
: wl_egl_window_create
(4) Create window surface
: eglCreateWindowSurface
(5) Create context
: eglCreateContext
(6) Bind context and surface
: eglMakeCurrent
Continued on next page
EGL Operation
Wayland Operation

a) Get Connection
Get Display Object by wl_display_connect (NULL).

b) Get Registroy Object / Set Listener
Get Registory Object and set a request of getting event to Compositor.

c) Wait for completion of request
Block until all pending request are processed by the server.

(1) Get display
Get a display specifying EGL_DEFAULT_DISPLAY in the argument display_id.

(2) Initialize EGL/set up rendering API
Initialize EGL and set up the rendering API.

(3) Get configuration
Get a configuration supported by the OpenGL ES 3.2 Library.

d) Create Surface for Wayland
Create a surface for Wayland.

e) Get shell Surface / Set Listener
Get a shell surface object and set the listener to get the events from the Compositor.

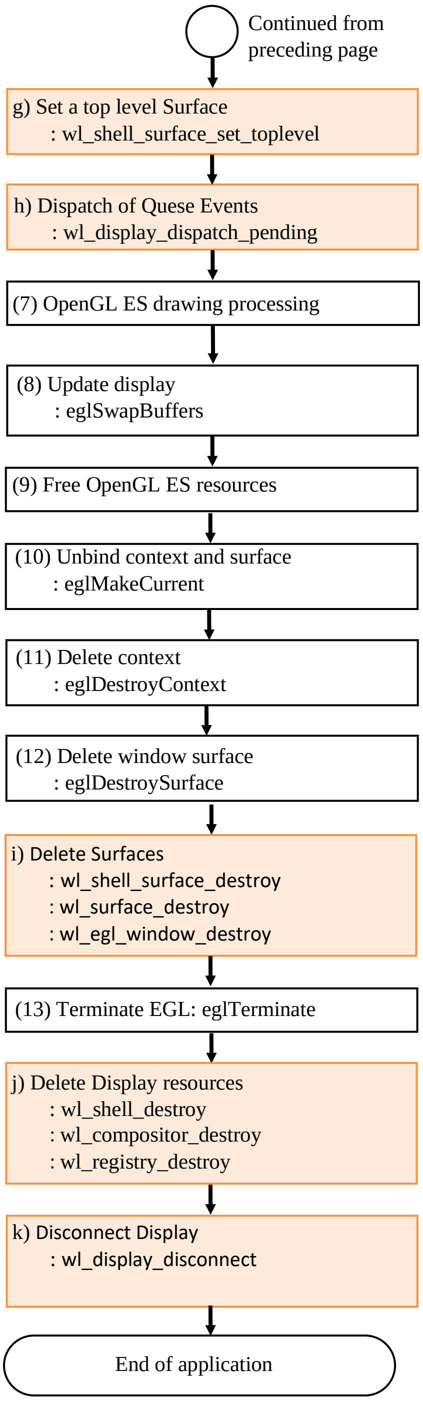
f) Create EGL native window object
Create an EGL native window object for Wayland.

(4) Create window surface
Create an EGL window surface.

(5) Create context
Create an EGL context.

(6) Bind context and surface
Set up the context and surface to be used as current ones.

**Figure 11 Window Surface Drawing Processing Flow (1)**


EGL Operation
Wayland Operation
Continued from preceding page
g) Set a top level Surface
: wl_shell_surface_set_toplevel
h) Dispatch of Quese Events
: wl_display_dispatch_pending
(7) OpenGL ES drawing processing
(8) Update display
: eglSwapBuffers
(9) Free OpenGL ES resources
(10) Unbind context and surface
: eglMakeCurrent
(11) Delete context
: eglDestroyContext
(12) Delete window surface
: eglDestroySurface
i) Delete Surfaces
: wl_shell_surface_destroy
: wl_surface_destroy
: wl_egl_window_destroy
(13) Terminate EGL: eglTerminate
j) Delete Display resources
: wl_shell_destroy
: wl_compositor_destroy
: wl_registry_destroy
k) Disconnect Display
: wl_display_disconnect
End of application

g) Set a top level Surface
Map the surface as a toplevel surface.

h) Dispatch of Quese Events
Dispatches events on the main event queue.

(7) OpenGL ES drawing processing
Create drawing objects, set up drawing parameters, and construct a drawing sequence according to the drawing scenes.

(8) Update display
Render primitives and update the display.

(9) Free OpenGL ES resources
Delete the drawing objects created in step (7) and free the OpenGL ES resources.

(10) Unbind the context and surface
Reset the current settings for the context and surface that are set up in step (6).

(11) Delete context
Delete the EGL context created in step (5).

(12) Delete surface
Delete the EGL window surface created in step (4).

i) Delete Surface
Delete a surface.

(13) Terminate EGL
Terminate EGL.

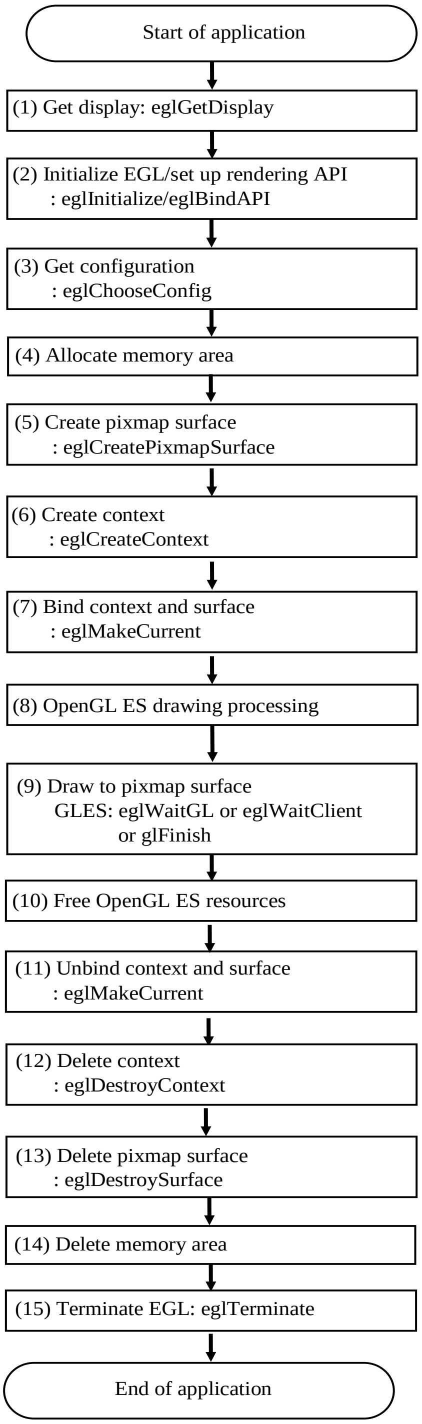
j) Delete Display resources
Delete the display resources created in step (b~c).

k) Disconnect Display
Disconnect Display connected in step (a).

**Figure 12 Window Surface Drawing Processing Flow (2)**

### 6.6.2.2 Pixmap Surface Drawing Processing Flow

Figure 13 shows the processing flow for drawing on a pixmap surface


Start of application
(1) Get display: eglGetDisplay
(2) Initialize EGL/set up rendering API
: eglInitialize/eglBindAPI
(3) Get configuration
: eglChooseConfig
(4) Allocate memory area
(5) Create pixmap surface
: eglCreatePixmapSurface
(6) Create context
: eglCreateContext
(7) Bind context and surface
: eglMakeCurrent
(8) OpenGL ES drawing processing
(9) Draw to pixmap surface
GLES: eglWaitGL or eglWaitClient
or glFinish
(10) Free OpenGL ES resources
(11) Unbind context and surface
: eglMakeCurrent
(12) Delete context
: eglDestroyContext
(13) Delete pixmap surface
: eglDestroySurface
(14) Delete memory area
(15) Terminate EGL: eglTerminate
End of application

(1) Get display
Get a display specifying EGL_DEFAULT_DISPLAY in the argument display_id.

(2) Initialize EGL/set up rendering API
Initialize EGL and set up the rendering API.

(3) Get configuration
Get a configuration supported by the OpenGL ES 3.2 Library.

(4) Allocate memory area
Allocate memory area assign to pixmap surface.

(5) Create pixmap surface
Create an EGL pixmap surface for the memory area created in step (4).

(6) Create context
Create an EGL context.

(7) Bind context and surface
Set up the context and surface to be used as current ones.

(8) OpenGL ES drawing processing
Create drawing objects, set up drawing parameters, and construct a drawing sequence according to the drawing scenes.

(9) Draw to pixmap surface
Draw to the pixmap surface.

(10) Free OpenGL ES resources
Delete the drawing objects created in step (8) and free the OpenGL ES resources.

(11) Unbind the context and surface
Reset the current settings for the context and surface that are set up in step (7).

(12) Delete context
Delete the EGL context created in step (6).

(13) Delete surface
Delete the EGL pixmap surface created in step (5).

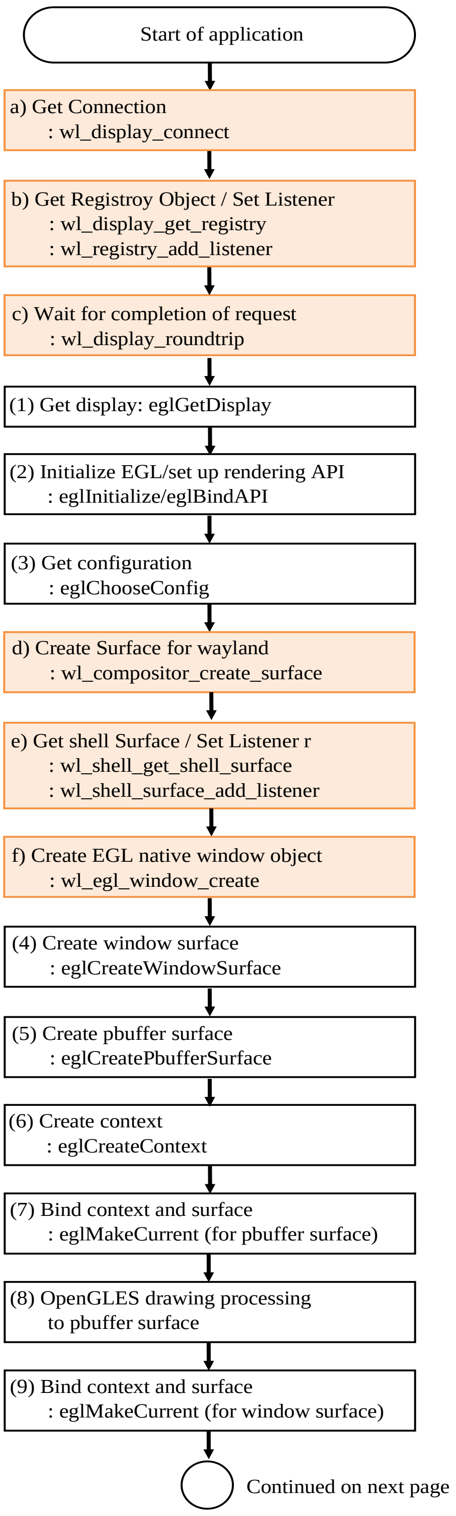
(14) Delete memory area
Delete the memory area created in step (4).

(15) Terminate EGL
Terminate EGL.

**Figure 13 Pixmap Surface Drawing Processing Flow**

### 6.6.2.3 Pbuffer Surface Drawing Processing Flow

Figure 14 shows the processing flow for drawing on a pbuffer surface. The flow is an example of drawing a texture using a drawing result of a pbuffer to a window surface.


EGL Operation
Wayland Operation
Start of application
a) Get Connection
: wl_display_connect
b) Get Registroy Object / Set Listener
: wl_display_get_registry
: wl_registry_add_listener
c) Wait for completion of request
: wl_display_roundtrip
(1) Get display: eglGetDisplay
(2) Initialize EGL/set up rendering API
: eglInitialize/eglBindAPI
(3) Get configuration
: eglChooseConfig
d) Create Surface for wayland
: wl_compositor_create_surface
e) Get shell Surface / Set Listener r
: wl_shell_get_shell_surface
: wl_shell_surface_add_listener
f) Create EGL native window object
: wl_egl_window_create
(4) Create window surface
: eglCreateWindowSurface
(5) Create pbuffer surface
: eglCreatePbufferSurface
(6) Create context
: eglCreateContext
(7) Bind context and surface
: eglMakeCurrent (for pbuffer surface)
(8) OpenGLES drawing processing
to pbuffer surface
(9) Bind context and surface
: eglMakeCurrent (for window surface)
Continued on next page

a) Get Connection
Get Display Object by wl_display_connect (NULL).

b) Get Registroy Object / Set Listener
Get Registory Object and set a request of getting event to Compositor.

c) Wait for completion of request
Block until all pending request are processed by the server.

(1) Get display
Get a display specifying EGL_DEFAULT_DISPLAY in the argument display_id.

(2) Initialize EGL/set up rendering API
Initialize EGL and set up the rendering API.

(3) Get configuration
Get a configuration supported by the OpenGL ES 3.2 Library.

d) Create Surface for Wayland
Create a surface for Wayland.

e) Get shell Surface / Set Listener
Get a shell surface object and set the listener to get the events from the Compositor.

f) Create EGL native window object
Create an EGL native window object for Wayland.

(4) Create window surface
Create an EGL window surface.

(5) Create pbuffer surface
Create an EGL pbuffer surface.

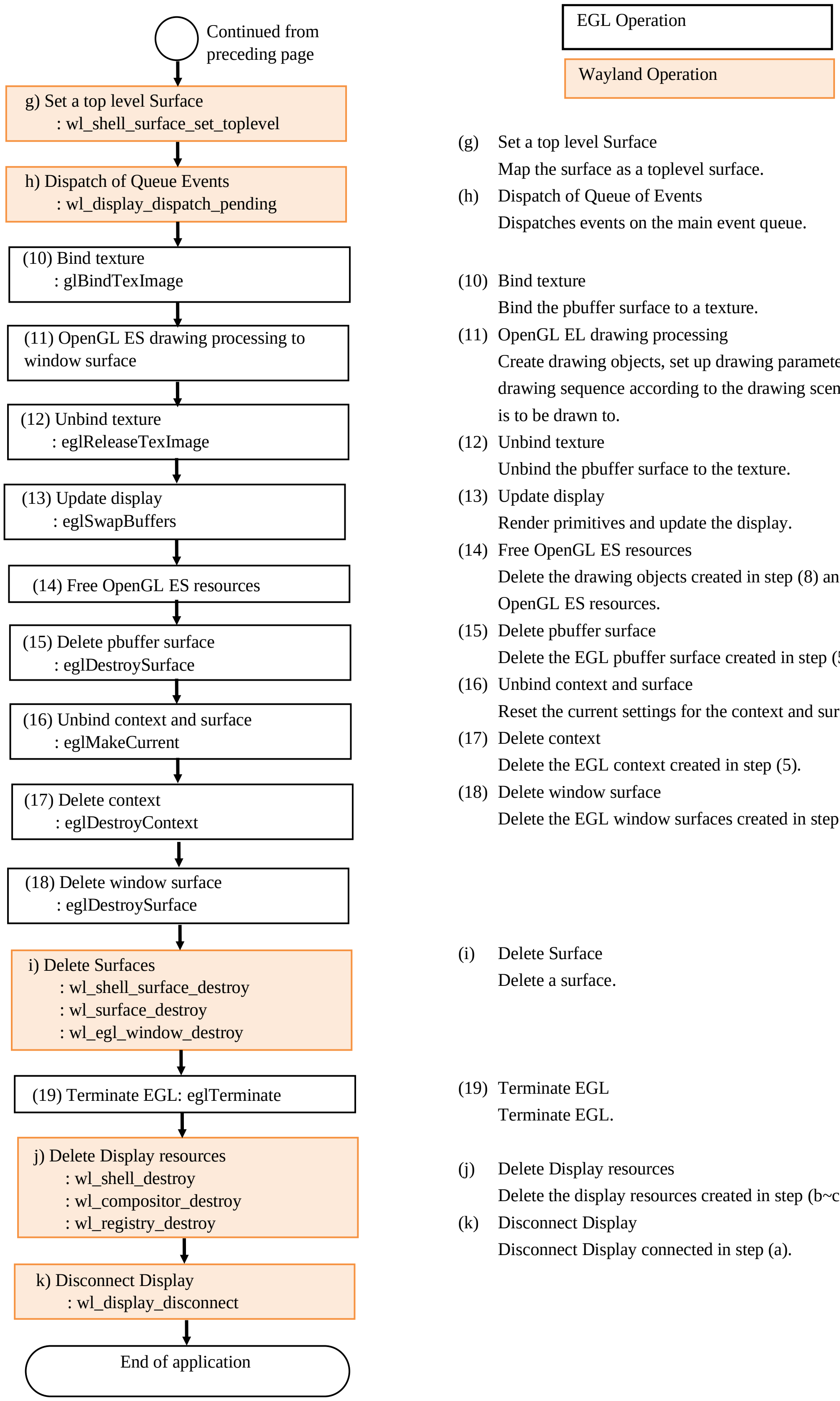
(6) Create context
Create an EGL context.

(7) Bind context and surface
Set up the context and surface to be used as current ones. Set the pbuffer surface to the current surface.

(8) OpenGL ES drawing processing
Create drawing objects, set up drawing parameters, and construct a drawing sequence according to the drawing scenes. The pbuffer surface is to be drawn to.

(9) Bind context and surface
Set up the context and surface to be used as current ones. Set the window surface to the current surface.

**Figure 14 Pbuffer Surface Drawing Processing Flow (1)**


EGL Operation
Wayland Operation
Continued from preceding page
g) Set a top level Surface : wl_shell_surface_set_toplevel
h) Dispatch of Queue Events : wl_display_dispatch_pending
(10) Bind texture : glBindTexImage
(11) OpenGL ES drawing processing to window surface
(12) Unbind texture : eglReleaseTexImage
(13) Update display : eglSwapBuffers
(14) Free OpenGL ES resources
(15) Delete pbuffer surface : eglDestroySurface
(16) Unbind context and surface : eglMakeCurrent
(17) Delete context : eglDestroyContext
(18) Delete window surface : eglDestroySurface
i) Delete Surfaces : wl_shell_surface_destroy : wl_surface_destroy : wl_egl_window_destroy
(19) Terminate EGL: eglTerminate
j) Delete Display resources : wl_shell_destroy : wl_compositor_destroy : wl_registry_destroy
k) Disconnect Display : wl_display_disconnect
End of application

(g) Set a top level Surface
Map the surface as a toplevel surface.

(h) Dispatch of Queue of Events
Dispatches events on the main event queue.

(10) Bind texture
Bind the pbuffer surface to a texture.

(11) OpenGL EL drawing processing
Create drawing objects, set up drawing parameters, and constructs a drawing sequence according to the drawing scenes. The window surface is to be drawn to.

(12) Unbind texture
Unbind the pbuffer surface to the texture.

(13) Update display
Render primitives and update the display.

(14) Free OpenGL ES resources
Delete the drawing objects created in step (8) and (11) and free the OpenGL ES resources.

(15) Delete pbuffer surface
Delete the EGL pbuffer surface created in step (5).

(16) Unbind context and surface
Reset the current settings for the context and surface.

(17) Delete context
Delete the EGL context created in step (5).

(18) Delete window surface
Delete the EGL window surfaces created in step (4).

(i) Delete Surface
Delete a surface.

(19) Terminate EGL
Terminate EGL.

(j) Delete Display resources
Delete the display resources created in step (b~c).

(k) Disconnect Display
Disconnect Display connected in step (a).

**Figure 15 Pbuffer Surface Drawing Processing Flow (2)**

| REVISION HISTORY | Linux Interface Specification Wayland<br>User’s Manual: Software | | |
|---|---|---|---|
| **Rev.** | **Date** | **Description** | |
| | | **Page** | **Summary** |
| 0.1 | Oct. 23, 2015 | — | New creation. |
| 0.2 | Mar. 25, 2016 | 2 | 1. Environment<br>Update the version of Wayland from 1.6.0 to 1.9.0.<br>Update the version of Weston from 1.6.0 to 1.9.0. |
| | | 6 | 4.1 wl_shm needs double memory area and memory copy<br>Change section title and description.<br>4.2 How to commit the buffer created by user application<br>New added for reference. |
| | | 7 | 4.3 Weston outputs to Analog RGB and LVDS display even if they aren’t connected<br>Change the sample of contents for weston.ini. |
| 0.3 | Apr. 15, 2016 | All | Add R-Car M3 support. |
| | | 4 | 2.2 Overview of new composite mode<br>New added for reference. |
| | | 5 | 2.3 Support Buffer Type<br>New added for reference. |
| | | 8 | 4.1 The memory copy is run in v4l2-renderer when the client application uses wl_shm.<br>Merged into 2.3.1<br>4.2. How to commit the buffer allocated by user application.<br>Merged into 2.3.2 |
| | | 9 | 4.6 How to set the composite priority.<br>New added for reference. |
| 0.4 | May. 20.2016 | 7~8 | 2.4 Wayland Feature<br>New added for description about wayland features. |
| | | 9 | 3.1 How to launch Wayland backend<br>Merged into 2.4.3. |
| 0.5 | Aug. 24,2016 | All | Add v4l2-renderer |
| 0.6 | Nov. 11.2016 | All | Wayland version up (v1.9.0 -> v1.11.0) |
| 0.65 | Nov. 25.2016 | 6 | 3.3 Delete libmediactl-v4l2 (v4l2-renderer controls V4L2 driver directly). |
| | | 11 | 4.1.2 Add gl-fallback mode. |
| | | 18 | 6.2.1 Configuration of Weston.ini<br>Change the value of repaint-window from "17" to "34" to support 30Hz/60Hz display. |
| 0.7 | Dec. 22.2016 | 9 | 4.1.2 Add scaling condition to gl-fallback |
| | | 17 | 6.2.1 [media-ctl] section of weston configuration is integrated into [v4l2-renderer] section. |
| 0.8 | Mar. 15.2017 | 19 | 6.2.2 Correction of Weston launch commands |
| | | 22 | 7.4 Add “How to set video mode” |
| | | 22 | 7.5 Add “Multi display composition” |
| | | All | Correction of Typo |
| 0.9 | Mar. 30.2017 | 16 | 4.4 Add dma-fence of GPU-rendering |
| | | All | Correction of Typo |
| 1.00 | Jul. 12.2017 | All | Add "CONFIDENTIAL" marks |
| | | 13 | 4.2 Add available buffer type in sprite plane |
| | | All | Correction of Typo |
| 1.01 | Sep. 13.2017 | All | Correction of Typo |
| 1.05 | Nov 7.2017 | All | Add R-Car M3N support. |
| 1.05 | Nov 7.2017 | All | Wayland version up (v1.11.0 -> v1.13.0), Weston version up(v1.11.0 -> 2.0.0) |
| | | All | Correction of Typo |
| 1.10 | Dec 22.2017 | All | Correction of Typo |

| | | | |
|---|---|---|---|
| 1.10 | Dec 22.2017 | 3 | 1.3 Function<br>Add Restriction of HDMI hot plug. |
| | | 22 | Delete “7.2 Connect the input device to the board” |
| | | 22 | 7.3 How to set video mode<br>Add Renesas backport patch information. |
| 1.2.0 | Mar 14.2018 | All | Correction of Typo |
| | | All | Add R-Car E3 support |
| | | 2 | 1.2.2. Related Documents<br>Modify graphics document name. |
| | | 4 | 1.3. Function<br>Add Restriction of scaling, gl-fallback, and surface size |
| | | 10,12 | 4.1.2. VSPB composition<br>Add “Optimization VSPB composition”<br>Add Restriction of gl-fallback with pixel alpha + global alpha case |
| | | 13 | 4.1.3. Support Pixel Format<br>Add “premultiplied alpha blending”<br>Add Restriction of DRM_FORMAT_ABGR8888 and DRM_FORMAT_RGBA8888 pixel formats.<br>Add Restriction of pixel format on gl-fallback and gl-renderer. |
| | | 15,17 | 4.2. Weston plane definitions<br>Add informations of Atomic mode setting and sprite plane<br>Delete description about mmnger |
| | | 25 | 7.3 Add “Connect the input device to the board” |
| 1.3.0 | Jun 11.2018 | 4 | 1.3. Function<br>The v4l2-renderer can use the close animation of application. |
| 1.4.0 | Oct 12.2018 | All | Correction of Typo |
| | | 4 | 1.3. Function<br>Updated minimum size of V4L2 driver. |
| 2.0.0 | Oct 30.2018 | All | Correction of Typo |
| 3.0.0 | Apr 06.2021 | All | Remove information of v4l2 renderer |
| | | 1, 6 | Wayland version up (v1.13.0 -> v1.18.0), Weston version up (v2.0.0 -> 8.0.0) |
| 3.1 | Aug 16.2021 | 6, 7, 8 | 3.3 Configuration of Renesas Wayland Solution<br>Change Wayland WSEGL to OSS package on Github.<br>Add system configuration and overview of drawing with GSX OpenGL ES Library. |
| | | 14 | 4.3 Synchronization<br>Move this to article 6. Appendix. |
| | | 19 | 6.3 Connect the input device to the board<br>Add note for the patch of DIP switch on kernel 5.4. |
| | | 20 - 27 | 6.6 Using EGL libraries<br>Add new article for using EGL libraries. |
| 3.1.1 | Nov 02.2021 | 3 | 1.3. Function<br>Delete limitation notes |
| | | 27-28 | 6.6.2.3 Pbuffer Surface Drawing Processing Flow<br>Add the missing flow part |

Linux Interface Specification Wayland
User’s Manual: Software

Publication Date: Rev.3.1.1 Nov. 02, 2021

Published by: Renesas Electronics Corporation

**SALES OFFICES**

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

**Renesas Electronics Corporation**
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

**Renesas Electronics America Inc.  Milpitas Campus**
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

**Renesas Electronics America Inc.  San Jose Campus**
6024 Silver Creek Valley Road, San Jose, CA 95138, USA
Tel: +1-408-284-8200, Fax: +1-408-284-2775

**Renesas Electronics Canada Limited**
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
Room 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai 200333, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, #06-02 Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

**Renesas Electronics India Pvt. Ltd.**
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700

**Renesas Electronics Korea Co., Ltd.**
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒135-0061 東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/

Colophon 3.1

Linux Interface Specification
Wayland User's Manual

Renesas Electronics Corporation