User's Manual

CONFIDENTIAL

RH850/D1x Device Family

Renesas Graphics Library Porting Layer Guide

Renesas Electronics
www.renesas.com

Rev. 2.0 May 2020

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding the porting layer functions of RGL (Renesas Graphics Library). This manual is written for engineers who use RGL.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

Please refer to documents of drivers and hardware for a target system implementing RGL as necessary.

The following documents are related documents. Make sure to refer to the latest versions of these documents.

Document Type	Description	Document Title	Document No.
User's manual for Hardware	Hardware specifications (pin assignments, memory maps, peripheral function specifications, electrical characteristics, timing charts) and operation description	RH850/D1L/D1M Group User's Manual: Hardware	R01UH0451EJ0220
User's manual for Software	Description of RGL overview	Renesas Graphics Library User's Manual: Software	R01US0181ED0400
	Description of WM	Renesas Graphics Library Window Manager (WM) Driver User's Manual: Software	LLWEB-10035990
	Description of SPEA	Renesas Graphics Library Sprite Engine A (SPEA) Driver User's Manual: Software	LLWEB-10035991
	Description of VDCE	Renesas Graphics Library Video Data Controller E (VDCE) Driver User's Manual: Software	LLWEB-10035992
	Description of VOWE	Renesas Graphics Library Video Output Warping Engine (VOWE) Driver User's Manual: Software	LLWEB-10035993
	Description of JCUA	Renesas Graphics Library JPEG Codec Unit A (JCUA) Driver User's Manual: Software	LLWEB-10035994
	Description of SFMA	Renesas Graphics Library Serial Flash Memory Interface A (SFMA) Driver User's Manual: Software	LLWEB-10064753
	Description of HYPB	Renesas Graphics Library HyperBus Controller (HYPB) Driver User's Manual: Software	LLWEB-10064754
	Description of OCTA	Renesas Graphics Library OctaBus Controller (OCTA) Driver User's Manual: Software	LLWEB-10064755
	Description of VOCA	Renesas Graphics Library Video Output Checker (VOCA) Driver User's Manual: Software	LLWEB-10063801

	Description of DISCOM	Renesas Graphics Library Display Output Comparator (DISCOM) Driver User's Manual: Software	LLWEB-10063802
	Description of DRW2D	Renesas Graphics Library 2D Graphics (DRW2D) Driver User's Manual: Software	LLWEB-10059472
Porting Layer Guide	Description of porting layer of RGL	Renesas Graphics Library Porting Layer Guide	LLWEB-10035995 (This manual)

2. Notation of Numbers and Symbols

This manual uses the following notation.

Binary	0bXXXXXXXX	(X=0 or 1)
Decimal	XXX	(X=0-9)
Hex	0xXXXXXXXX	(X=0-9,A-F)

3. List of Abbreviations and Acronyms

Abbreviation	Full Form
API	Application Programming Interface
CLUT	Color Look Up Table
Context	An internal state machine of the single framework
CPU	Central Processing Unit. The microprocessor core of the LSI.
DDB	Display Database
DISCOM	Display Output Comparator. This is H/W, which calculates CRC.
ECM	Error Control Module. This is H/W, which handles error interrupt.
Frame buffer	A region in the memory attached to a window that can be shown on the screen; A region in the memory holding the bitmap as the result of GPU/JCUA rendering activities
GPU	Graphics Processing Unit. This is H/W which controls 2D graphics rendering.
Hsync	Horizontal Sync Pulse that synchronize the start of the horizontal picture scan
H/W	Hardware
INT_STAx	Interrupt Status Bit
JCUA	JPEG Codec Unit A. This is H/W which controls JPEG decoder.
JDTI	Image Data Transfer Interrupt
JEDI	Compression/De compression process interrupt
Layer	H/W concept of the stackable visual area on the display
LDI	High Bandwidth digital video interface standard for connecting graphics/video processors to flat panel monitors
LVTTL	Low Voltage Transistor Transistor Logic
OIR	Output Image Rendering. This is H/W block in VDCE.
Pitch	(a.k.a. stride) Distance in pixels between two adjacent pixel rows of the framebuffer in the memory
RLE	Run Length Encoding. TARGA run-length encoded image standard, for easy image compression, supported by the SPEA.
Screen	A physical display surface; SW abstraction of the attached physical display
SPEA	Sprite Engine A. This is H/W, which controls Sprite data and RLE decoding.
Sprite	Graphical entity which can be moved on screen independently
Surface	A concrete (i.e. physical) implementation of the window’s area
S/W	Software
Texture	A binary image registered with the GPU driver that can be transformed and drawn to the framebuffer in a HW accelerated way
VBLANK	Vertical Blanking
VDCE	Video Data Controller E. This is H/W, which controls video input, image synthesis and video output.
VOCA	Video Output Checker A. This is H/W, which monitors video output signal.
VODDR	Video Output DDR
VOWE	Video Output Warping Engine. This is H/W, which controls video warping.
Vsync	Vertical Sync
Window	A SW abstraction of the rectangular visual area that can be shown on the display
WM	Window Manager. This is a driver stack, which enables an abstract access to VDCE driver and SPEA driver.

Table of Contents

1. Overview

1.1 Scope

In order to work in a user defined hardware platform with RH850/D1x, a level of software porting is required. This guide describes the changes required to port to a customer's platform.

1.2 Component Structure

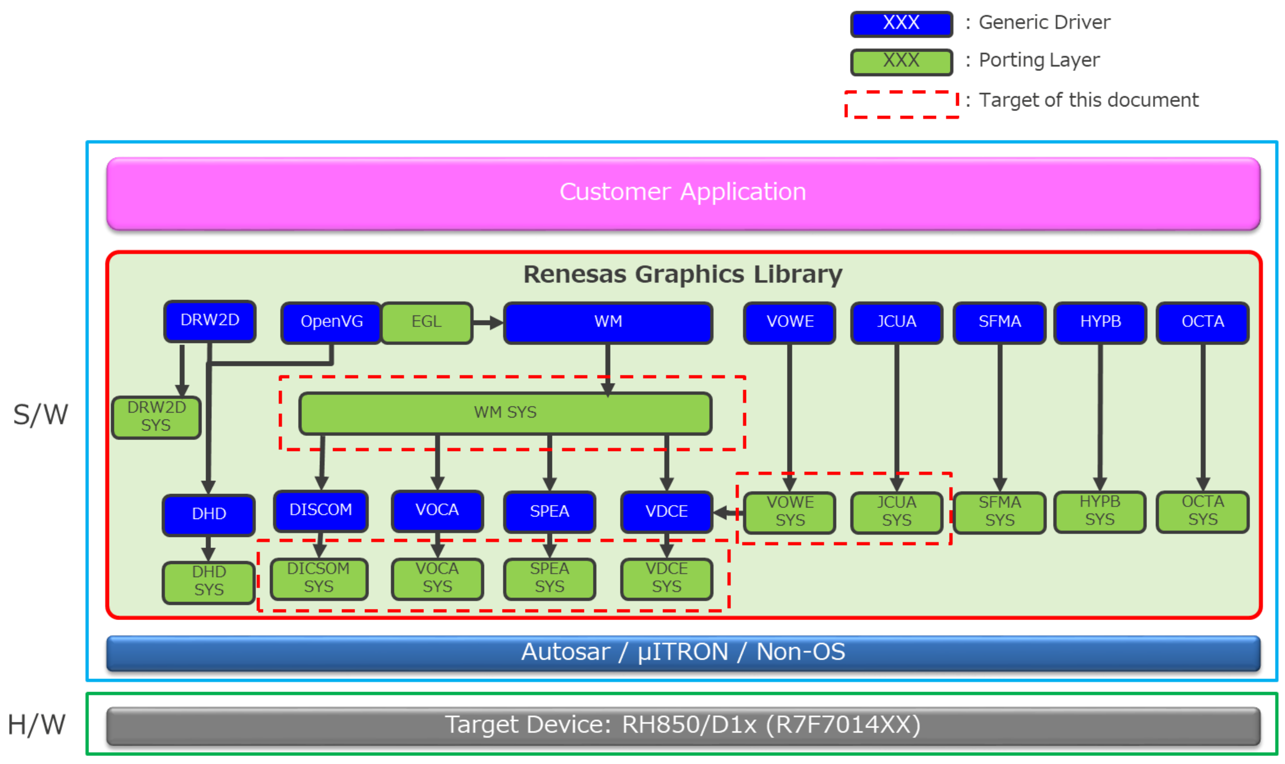


Figure 1-1 RGL component structure

Renesas Graphics Library is the Renesas driver stack for graphic related H/W.
Each RGL generic driver controls responsible H/W directly. However, outside H/W (like Clock, interrupt, Port etc...) is controlled via the porting layer. Poring layer also controls OS related part.

2. Basic Specification

2.1 Reserved word

Table 2-1 Prefixes

Prefix	Description
R_WM_*	Prefix for WM Module
r_wm_*	
R_WM_Sys_*	
R_SPEA_*	Prefix for SPEA Module
r_spea_*	
R_SPEA_SYS_*	
R_VDCE_*	Prefix for VDCE Module
r_vdce_*	
R_VDCE_Sys_*	
R_VOWE_*	Prefix for VOWE Module
r_vowe_*	
R_VOWE_Sys_*	
R_JCUA_*	Prefix for JCUA Module
r_jcua_*	
R_JCUA_Sys_*	
R_VOCA_*	Prefix for VOCA Module
r_voca_*	
R_VOCA_Sys_*	
R_DISCOM_*	Prefix for DISCOM Module
r_discom_*	
R_DISCOM_Sys_*	
r_dev_*	Prefix for Device module
R_DEV_*	
r_dbb_*	Prefix for Database for display timings module
R_DBB_*	

3.Window Manager (WM)

3.1 File list

Following table shows the file list for WM porting layer .

Table 3-1 File list for WM porting layer

File Name	Pass	Description
r_sys_wm.c	vlib/device/d1x_common/src/wm	Source file of poring layer for common part.
r_sys_wm_capture.c	vlib/device/d1x_common/src/wm	Source file of poring layer for capture control
r_sys_wm_screen.c	vlib/device/d1x_common/src/wm	Source file of poring layer for screen control.
r_sys_wm_sprite.c	vlib/device/d1x_common/src/wm	Source file of poring layer for sprite control.
r_sys_wm_window.c	vlib/device/d1x_common/src/wm	Source file of poring layer for window control.
r_sys_wm_discom.c	vlib/device/d1lx/src/wm vlib/device/d1mx/src/wm (*1)	Source file of poring layer for DISCOM control.
r_sys_wm_ecm.c	vlib/device/d1lx/src/wm vlib/device/d1mx/src/wm (*1)	Source file of poring layer for ECM control.
r_sys_wm_voca.c	vlib/device/d1lx/src/wm vlib/device/d1mx/src/wm (*1)	Source file of poring layer for VOCA control.
r_sys_wm.h	vlib/device/d1x_common/src/wm	Header file of poring layer internal.
r_config_wm.h	vlib/device/d1x_common/macro_cfg/wm	Header file for device configuration.
r_wm_sys.h	vlib/macro/wm/lib	Header file for porting layer interface.

(*1) Pass differs depending on RH850/D1L RGL or RH850/D1M RGL.

3.2 Driver support functions

The generic WM driver uses these functions. They have to be implemented within the driver library for a concrete device. (e.g. D1L, D1M). Support functions are functions that are not part of the driver itself but they must be provided to integrate the driver on a particular device, OS or board.

3.2.1 WM Basic Interface Functions

3.2.1.1 R_WM_Sys_DevInit

Function Prototype

```
uint32_t R_WM_Sys_DevInit(const uint32_t Unit,
                          void          (*EventCb) (uint32_t Unit,
                                                     const r_wm_Event_t *Event))
```

Input Parameter

Table 3-2 Input Parameter of R_WM_Sys_DevInit

Parameter	Description
Unit	Specifies the WM Unit number.
EventCb	Specifies the Callback function for receiving events. This can be NULL.

Input – Output Parameter

None

Output Parameter

None

Description

This function will initialize the platform specific portion of the device.
This function is called from R_WM_DevInit.

Default Behavior

This function executes the following processing.

- Initializes some global variables of specifies WM unit.
- Registers the Event Callback to notify the VDCE, DISCOM and VOCA event.
- Initializes VDCE by executing the function R_VDCE_Init.
- Sets the callback function to detect VDCE errors by R_VDCE_ErrorCallbackSet
- Disables the display output by executing the function R_VDCE_DisplayDisable.
- If another WM unit is uninitialized state (or does not exist):
  - Initializes some global variables of another WM unit.
  - Initializes SPEA by executing the function R_SPEA_Init.
  - Sets the callback function to detect SPEA errors by R_SPEA_SetErrorCallback.
  - Disables all the RLE engines and RLE units by executing the function R_SPEA_SetRle and R_SPEA_UnitEnable.
  - Initializes ECM sample driver
  - R_WM_Sys_IsrVocaErr is installed as ISR of INTVOCAERR interrupt.
  - Enable INTVOCAERR interrupt.
- Initializes DISCOM by executing the function R_DISCOM_Init.
- Sets the callback function to detect DISCOM errors by R_DISCOM_ErrorCallbackSet.

Customizing Points

If an OS is used, change the OS synchronization variables accordingly.
Change the implementation of ECM sample driver control when the user implements original ECM control.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.1.2 R_WM_Sys_DevDeinit

Function Prototype

```
uint32_t R_WM_Sys_DevDeinit(const uint32_t   Unit)
```

Input Parameter

Table 3-3 Input Parameter of R_WM_Sys_DevDeinit

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function will de-initializes the porting portion of the device.
This function is called from R_WM_DevDeinit.

Default Behavior

This function executes the following processing.

- Prepare shutdown flag for trying to escape from any wait states.
- Initializes some global variables.
- If another WM unit is uninitialized state (or does not exist):
  - De-initializes SPEA by executing the function R_SPEA_DeInit.
  - Disables INTVOCAERR interrupt of ECM.
  - De-initializes ECM sample driver.
  - Disables VOCA interrupt by executing R_VOCA_IntDisable.
  - De-initializes VOCA driver by executing R_VOCA_DeInit.
  - Clears global flags that shows initialized VOCA driver.
- Disables DISCOM interrupt by executing R_DISCOM_IntDisable.
- De-initializes DISCOM by executing the function R_DISCOM_DeInit.
- De-initializes VDCE by executing the function R_VDCE_DeInit.

Customizing Points

If an OS is used, change the OS synchronization variables accordingly.
Change the implementation of ECM sample driver control when the user implements original ECM control.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.1.3 R_WM_Sys_DevEventRegister

Function Prototype

```
uint32_t R_WM_Sys_DevEventRegister(const uint32_t          Unit,
                                   const r_wm_EventId_t     EventId,
                                   const uint32_t           Arg)
```

Input Parameter

Table 3-4 Input Parameter of R_WM_Sys_DevEventRegister

Parameter	Description
Unit	Specifies the WM Unit number.
EventId	Specifies the ID of the event. R_WM_EVENT_VBLANK R_WM_EVENT_SCANLINE R_WM_EVENT_VI_VBLANK R_WM_EVENT_VI_OVERFLOW R_WM_EVENT_LAYER0_UNDERFLOW R_WM_EVENT_LAYER1_UNDERFLOW R_WM_EVENT_LAYER2_UNDERFLOW R_WM_EVENT_LAYER3_UNDERFLOW R_WM_EVENT_LAYER1_VBLANK R_WM_EVENT_OIR_VBLANK R_WM_EVENT_OIR_SCANLINE R_WM_EVENT_DISCOM_MISMACTH R_WM_EVENT_VOCA_MISMACTH R_WM_EVENT_ACT_MON_ERROR
Arg	Specifies the scan line number on which the event will trigger. This argument is valid when EventId is R_WM_EVENT_SCANLINE or R_WM_EVENT_OIR_SCANLINE.

Input – Output Parameter

None

Output Parameter

None

Description

This function enables the VDCE, VOCA and DISCOM event callback.
This function is called from R_WM_DevEventRegister.

VDCE Callback is also controlled by following functions.

- R_WM_Sys_ScreenEnable
  - R_WM_EVENT_VBLANK
  - R_WM_EVENT_SCANLINE
  - R_WM_EVENT_LAYER1_VBLANK
- R_WM_Sys_CaptureEnable
  - R_WM_EVENT_VI_VBLANK

Regarding R_WM_EVENT_VI_VBLANK and R_WM_EVENT_VI_OVERFLOW, these events occurs when the capture is enabled. Specify the unit number of the capture side to Unit.

Default Behavior

This function executes the following processing.

- Sets the event trigger flag.
- Executes the following processing depending on EventId.

EventId	Processing
R_WM_EVENT_SCANLINE	• Sets the scan line by executing R_VDCE_IntcScanlineSet.
R_WM_EVENT_OIR_SCANLINE	• Sets the scan line by executing R_VDCE_IntcOirScanlineSet. • Sets the internal callback function by executing R_VDCE_IntcCallbackSet. • Enables the interrupt by executing R_VDCE_IntcEnable.
R_WM_EVENT_VI_OVERFLOW R_WM_EVENT_LAYER0_UNDERFLOW R_WM_EVENT_LAYER1_UNDERFLOW R_WM_EVENT_LAYER2_UNDERFLOW R_WM_EVENT_LAYER3_UNDERFLOW R_WM_EVENT_OIR_VBLANK	• Sets the internal callback function by executing R_VDCE_IntcCallbackSet. • Enables the interrupt by executing R_VDCE_IntcEnable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.1.4 R_WM_Sys_IsShutdownActive

Function Prototype

```
uint32_t R_WM_Sys_IsShutdownActive(const uint32_t Unit)
```

Input Parameter

Table 3-5 Input Parameter of R_WM_Sys_IsShutdownActive

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function checks the shutdown status.
Shutdown status is from the start of R_WM_Sys_DevDeinit process to next R_WM_Sys_DevInit.
This function is called from R_WM_FrameWait in order to early-exit the job handling.

Default Behavior

This function executes the following processing.

- In the shutdown status, returns 1. Otherwise returns 0.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

1 - Shutdown active
0 - Shutdown inactive

See also

None

3.2.1.5 R_WM_Sys_DevCountGet

Function Prototype

```
uint32_t R_WM_Sys_DevCountGet(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

This function will get the number of available WM units.
This function is called from several WM APIs.

Default Behavior

This function executes the following processing.

- This function returns the available WM (VDCE) unit number depending on the device.

Customizing Points

If user uses only WM Unit0, user can change the R_WM_DEV_NUM definition from 2 to 1.
It can be reduced the RAM size like a global variable defined in Porting layer.
In that case the user needs to guarantee not to specify Unit 1 for the WM API.

Return Codes

Number of WM units.

See also

None

3.2.1.6 R_WM_Sys_DevInfoGet

Function Prototype

```
void R_WM_Sys_DevInfoGet(const uint32_t        Unit,
                         uint32_t       *const MaxLayers,
                         uint32_t       *const MaxPitch,
                         uint32_t       *const MaxWidth,
                         uint32_t       *const MaxHeight,
                         uint32_t       *const SpritesOrderAscending)
```

Input Parameter

Table 3-6 Input Parameter of R_WM_Sys_DevInfoGet

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

Table 3-7 Output Parameter of R_WM_Sys_DevInfoGet

Parameter	Description
MaxLayers	Maximum number of layers supported.
MaxPitch	Maximum layer pitch (stride) supported.
MaxWidth	Maximum layer width supported.
MaxHeight	Maximum layer height supported.
SpritesOrderAscending	Sprites Z-order ascending or descending. 1 : ascending 0 : descending Ascending means that the sprite with the greater index has the greater Z-order priority (comes on top of the sprites with the lower index).

Description

This function gets the platform specific information.
This function is called from R_WM_DevInit.

Default Behavior

This function executes the following processing.

- Stores the default values to the argument.

Customizing Points

The value may be changed according to device specification and user usage restriction.
MaxPitch can be extended to VDCE maximum size (262136 / bpp).

Return Codes

None

See also

None

3.2.1.7 R_WM_Sys_DeviceFeature

Function Prototype

```
uint32_t R_WM_Sys_DeviceFeature(const r_wm_sys_DevFeature_t    Feature)
```

Input Parameter

Table 3-8 Input Parameter of R_WM_Sys_DeviceFeature

Parameter	Description
Feature	Specifies the feature support to be checked. R_WM_SYS_FEATURE_RLE_LAYER_NO R_WM_SYS_FEATURE_SPRITE_LAYER_NO R_WM_SYS_FEATURE_SWITCH_CAPABILITIES R_WM_SYS_FEATURE_GAMMA_CORRECTION R_WM_SYS_FEATURE_SCALING

Input – Output Parameter

None

Output Parameter

None

Description

This function will get device specific information about certain features.
This function is called from R_WM_WindowCapabilitiesSet and R_WM_Sys_DevInit.
Note that this function may be called before the R_WM_Sys_DevDeInit.

Default Behavior

This function executes the following processing.

- Return the values depending on target RH850/D1x device.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

0	- Not supported or not available.
Greater than 0	- Supported and Value will be the number of supported layer.

See also

r_wm_sys_DevFeature_t

3.2.1.8 R_WM_Sys_StateSet

Function Prototypes

```
r_wm_Error_t R_WM_Sys_StateSet(const uint32_t        Unit,
                               const r_wm_State_t     State)
```

Input Parameter

Table 3-9 Input Parameter of R_WM_Sys_StateSet

Parameter	Description
Unit	Specifies the WM Unit number.
State	Specifies the state of WM driver. R_WM_STATE_UNINITIALISED R_WM_STATE_INITIALISED R_WM_STATE_DISPLAY_INITIALIZED R_WM_STATE_DISPLAY_ACTIVE

Input – Output Parameter

None

Output Parameter

None

Description

This function changes the status of WM unit.

Default Behavior

This function executes the following processing.

- This function change the current state of WM driver to requested state.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_ERR_OK	- No error has occurred.
R_WM_ERR_NG	- Incorrect parameters.

See Also

r_wm_State_t

3.2.1.9 R_WM_Sys_StateGet

Function Prototypes

```
r_wm_State_t R_WM_Sys_StateGet(const uint32_t        Unit)
```

Input Parameter

Table 3-10 Input Parameter of R_WM_Sys_StateGet

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to get the state of WM driver.
This function is called from several APIs.

Default Behavior

This function executes the following processing.

- This function will return the current status of WM unit.
- If unsupported WM unit number is specified, this function returns R_WM_STATE_UNINITIALISED.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_STATE_UNINITIALISED	- WM Unit is Uninitialized state.
R_WM_STATE_INITIALISED	- WM Unit is Initialized state.
R_WM_STATE_DISPLAY_INITIALIZED	- WM Unit is Display Initialized state.
R_WM_STATE_DISPLAY_ACTIVE	- WM Unit is Display Active state.

See Also

r_wm_State_t

3.2.1.10 R_WM_Sys_InitGlobal

Function Prototypes

```
void R_WM_Sys_InitGlobal(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

Initializes the global variables in WM porting layer.

If R_BSP_SYS_INIT_USE is defined, user must call this function before calling WM APIs.
This function is called from R_DEV_SysInit provided as sample code.

If R_BSP_SYS_INIT_USE is not defined, global variables are declared with initial values.
This function call is not mandatory.

Default Behavior

This function executes the following processing.

- Initialize global variables.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

3.2.1.11 R_WM_Sys_IsrVocaErr

Function Prototypes

```
void R_WM_Sys_IsrVocaErr(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

This function is executed in response to the INTVOCAERR interrupt.
This function is called from WM porting layer internally.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Does nothing.

RH850/D1Mx

- Disables INTVOCAERR interrupt of ECM.
- Checks the initialized VOCA unit.
- Gets interrupt factor by executing R_VOCA_StatusGet.
- Clears interrupt factor by executing R_VOCA_StatusClear.
- Notifies R_WM_EVENT_VOCA_MISMACTH event to user if applicable.
- Notifies R_WM_EVENT_ACT_MON_ERROR event to user if applicable.
- Checks the initialized DISCOM unit.
- Gets interrupt factor by executing R_DISCOM_StatusGet.
- Clears interrupt factor by executing R_DISCOM_StatusClear.
- Notifies R_WM_EVENT_DISCOM_MISMACTH event to user if applicable.
- Enables INTVOCAERR interrupt of ECM.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

3.2.2 WM Basic Internal Frame Synchronous Control Functions

3.2.2.1 R_WM_Sys_DevFrameStarted

Function Prototype

```
void R_WM_Sys_DevFrameStarted(const uint32_t Unit)
```

Input Parameter

Table 3-11 Input Parameter of R_WM_Sys_DevFrameStarted

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function handles the start of processing a sequence of message queues.
This function is called from R_WM_FrameExecute.

Default Behavior

This function executes the following processing.

- Resets the all frame status flags to collect new status.
- Resets the all VOCA status flags to collect new status.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See also

None

3.2.2.2 R_WM_Sys_DevFrameFinished

Function Prototype

```
void R_WM_Sys_DevFrameFinished(const uint32_t Unit)
```

Input Parameter

Table 3-12 Input Parameter of R_WM_Sys_DevFrameFinished

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function handles the end of processing a sequence of message queues.
This function is called from R_WM_FrameExecuteNext.

Default Behavior

This function executes the following processing.

- Update frame buffer if update is pending. The pending occurs when the layer is locked in R_WM_EVENT_VI_VBLANK timing during capture.
- Reassign all Sprite data of the sprite unit that got changed during this frame.
  - Set configuration by R_SPEA_SetSprite.
  - Enable / Disable the sprite data by R_SPEA_SpriteEnable.
- Set update request for changing Sprite unit with function R_SPEA_UpdateUnit.
- Set update request for changing RLE unit with function R_SPEA_UpdateUnit.
- If no window was visible in this frame, creates a SYNC signal manually by toggling UPDTn bit in VDCECTL register (Refer HW UM 37.6).
- Updates CLUT by executing R_VDCE_LayerClutSet if the layer got changed during this frame.
- Updates global status flags that holds window visible state.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See also

None

3.2.2.3 R_WM_Sys_DevWaitForHwWriteReady

Function Prototype

```
void R_WM_Sys_DevWaitForHwWriteReady(const uint32_t Unit)
```

Input Parameter

Table 3-13 Input Parameter of R_WM_Sys_DevWaitForHwWriteReady

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

Waits until it is safe (in terms of time to finish) to start writing the hardware registers.
This function is called from R_WM_FrameWait.

Default Behavior

This function executes the following processing.

- Wait until Scan line interrupt triggers.

Customizing Points

If an OS is used, change the OS synchronization variables accordingly.

Return Codes

None

See also

None

3.2.2.4 R_WM_Sys_DevWaitForHwUpdated

Function Prototype

```
void R_WM_Sys_DevWaitForHwUpdated(const uint32_t   Unit)
```

Input Parameter

Table 3-14 Input Parameter of R_WM_Sys_DevWaitForHwUpdated

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

Wait until the hardware registers update is finished.
This function is called from R_WM_FrameWait.

Default Behavior

This function executes the following processing.

- Wait until VBLANK interrupt triggers.

Customizing Points

If an OS is used, change the OS synchronization variables accordingly.

Return Codes

None

See also

None

3.2.2.5 R_WM_Sys_DevRootWindowSet

Function Prototype

```
void R_WM_Sys_DevRootWindowSet(const uint32_t        Unit,
                               r_wm_Window_t  *const RootWin)
```

Input Parameter

Table 3-15 Input Parameter of R_WM_Sys_DevRootWindowSet

Parameter	Description
Unit	Specifies the WM Unit number.
RootWin	Specifies the root window for the device window linked list. R_NULL can be set to clear the root window.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the root window for the device window linked list.
This function is also called from message queue of following functions.

- R_WM_WindowCreate
- R_WM_WindowDelete
- R_WM_WindowMove

Default Behavior

This function executes the following processing.

- Set the root window in the window linked list.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See also

None

3.2.2.6 R_WM_Sys_DevRootCaptureSet

Function Prototype

```
void R_WM_Sys_DevRootCaptureSet(const uint32_t        CapUnit,
                                r_wm_Capture_t *const RootCapt)
```

Input Parameter

Table 3-16 Input Parameter of R_WM_Sys_DevRootCaptureSet

Parameter	Description
CapUnit	Specifies the WM unit number to capture.
RootCapt	Specifies the root capture surface for the capture surface linked list. R_NULL can be set to clear the root window.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the root capture surface for the capture surface linked list.
This function is also called from message queue of following functions.

- R_WM_CaptureCreate
- R_WM_CaptureDelete

Default Behavior

This function executes the following processing.

- Set the root capture surface in the capture surface linked list.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See also

None

3.2.2.7 R_WM_Sys_DevRootVocaSet

Function Prototype

```
void R_WM_Sys_DevRootVocaSet(const uint32_t        Unit,
                             r_wm_Voca_t    *const RootVoca)
```

Input Parameter

Table 3-17 Input Parameter of R_WM_Sys_DevRootVocaSet

Parameter	Description
Unit	Specifies the WM Unit number.
RootVoca	Specifies the root VOCA monitor area for the linked list. R_NULL can be set to clear the root VOCA monito area.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the root VOCA monitor area for the linked list.
This function is called from message queue of following functions.

- R_WM_ScreenVocaCreate
- R_WM_ScreenVocaDelete
- R_WM_ScreenVocaDeInit

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Does nothing.

RH850/D1Mx

- Sets the root VOCA monitor area in the linked list.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See also

None

3.2.2.8 R_WM_Sys_DevRootDiscomSet

Function Prototype

```
void R_WM_Sys_DevRootDiscomSet(const uint32_t        Unit,
                               r_wm_Discom_t   *const RootDiscom)
```

Input Parameter

Table 3-18 Input Parameter of R_WM_Sys_DevRootDiscomSet

Parameter	Description
Unit	Specifies the WM Unit number.
RootDiscom	Specifies the root DISCOM device for the linked list. R_NULL can be set to clear the root Discom device.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the root DISCOM device for the linked list.
This function is called from message queue of following functions.

- R_WM_DiscomCreate
- R_WM_DiscomDelete

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Does nothing.

RH850/D1Mx

- Sets the root DISCOM device in the linked list.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See also

None

3.2.3 WM Screen Interface Functions

3.2.3.1 R_WM_Sys_ScreenTimingSet

Function Prototype

```
uint32_t R_WM_Sys_ScreenTimingSet(const uint32_t              Unit,
                                  const r_ddb_Timing_t *const timing)
```

Input Parameter

Table 3-19 Input Parameter of R_WM_Sys_ScreenTimingSet

Parameter	Description
Unit	Specifies the WM Unit number.
Timing	Specifies the screen timing.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the screen timings manually.
This function enables VDCE interrupts and sets default scan line timing.
About r_ddb_Timing_t structure, see VDCE driver User's manual.
This function is called from R_WM_ScreenTimingSet and R_WM_ScreenTimingSetByName.

Default Behavior

This function executes the following processing.

- Sets the display timing, including display resolution, sync position, blank widths and pixel clocks by executing R_VDCE_DisplayTimingSet.
- Sets the scan line by executing the function R_VDCE_IntcScanlineSet.
  - Scan line is (timing->ScreenHeight - 50) in case of (timing->ScreenHeight > 240)
  - Scan line is (timing->ScreenHeight - (timing->ScreenHeight / 10)) in case of (timing->ScreenHeight <= 240)
- Resets the global variable to keep screen information.

Customizing Points

The default value of Scan Line can be customized.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.2 R_WM_Sys_ScreenTimingSetByName

Function Prototype

```
uint32_t R_WM_Sys_ScreenTimingSetByName(const uint32_t      Unit,
                                        const int8_t *const Name)
```

Input Parameter

Table 3-20 Input Parameter of R_WM_Sys_ScreenTimingSetByName

Parameter	Description
Unit	Specifies the WM Unit number.
Name	Specifies the name of the display database.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the video timings by referencing the display database.
This function is called from R_WM_ScreenTimingSetByName.

Default Behavior

This function executes the following processing.

- Get display timing parameter from R_DDB_GetDisplayTiming.
- Executes R_WM_Sys_ScreenTimingSet.

Customizing Points

Display database is described in vlib/macro/vo/ddb/src/r_ddb_timing.c.
User can add the display database to r_ddb_timing.c.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_ScreenTimingSet

3.2.3.3 R_WM_Sys_ScreenEnable

Function Prototype

```
uint32_t R_WM_Sys_ScreenEnable(const uint32_t     Unit,
                               const uint32_t     Enabled)
```

Input Parameter

Table 3-21 Input Parameter of R_WM_Sys_ScreenEnable

Parameter	Description
Unit	Specifies the WM Unit number.
Enabled	Specifies the control. 0 : Disable 1 : Enable

Input – Output Parameter

None

Output Parameter

None

Description

This function enables or disables the screen.
This function is called from R_WM_DevDeinit.
This function is also called from message queue of following functions.

- R_WM_ScreenEnable
- R_WM_ScreenDisable

Default Behavior

This function executes the following processing.
For Enable:

- Sets the callback for the following interrupt type by executing R_VDCE_IntcCallbackSet.
  Always;
  - R_VDCE_INTC_SCANLINE
  - R_VDCE_INTC_VBLANK
  - R_VDCE_INTC_VBLANK_1

  When the event is set by R_WM_Sys_DevEventRegister;
  - R_VDCE_INTC_ERR_LAYER0_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER1_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER2_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER3_UNDERFLOW
  - R_VDCE_INTC_OIR_VBLANK
  - R_VDCE_INTC_OIR_SCANLINE
- Enables the following interrupts by executing R_VDCE_IntcEnable.
  Always;
  - R_VDCE_INTC_SCANLINE
  - R_VDCE_INTC_VBLANK
  - R_VDCE_INTC_VBLANK_1

  When the event is set by R_WM_Sys_DevEventRegister;
  - R_VDCE_INTC_ERR_LAYER0_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER1_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER2_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER3_UNDERFLOW
  - R_VDCE_INTC_OIR_VBLANK
  - R_VDCE_INTC_OIR_SCANLINE

- Enables the display output by executing R_VDCE_DisplayEnable.
- When Unit is 0, checks global flags and if VOCA is not initialized, executes following processing.
  - Initializes the VOCA driver by executing R_VOCA_Init
  - Sets global flags that shows initialized VOCA driver.
  - Sets the callback function to detect VOCA errors by executing R_VOCA_ErrorCallbackSet.
  - Enables VOCA interrupt by executing R_VOCA_IntEnable

For Disable:

- Disables the following interrupts by executing R_VDCE_IntcDisable.
  - R_VDCE_INTC_SCANLINE
  - R_VDCE_INTC_VBLANK
  - R_VDCE_INTC_CAP_VBLANK
  - R_VDCE_INTC_ERR_CAP_WRITE_OVERFLOW
  - R_VDCE_INTC_ERR_LAYER0_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER1_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER2_UNDERFLOW
  - R_VDCE_INTC_ERR_LAYER3_UNDERFLOW
  - R_VDCE_INTC_VBLANK_1
  - R_VDCE_INTC_OIR_VBLANK
  - R_VDCE_INTC_OIR_SCANLINE
- Disables the display output by executing R_VDCE_DisplayDisable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.4 R_WM_Sys_ScreenBgColorSet

Function Prototype

```
uint32_t R_WM_Sys_ScreenBgColorSet(const uint32_t        Unit,
                                   const uint8_t         Red,
                                   const uint8_t         Green,
                                   const uint8_t         Blue)
```

Input Parameter

Table 3-22 Input Parameter of R_WM_Sys_ScreenBgColorSet

Parameter	Description
Unit	Specifies the WM Unit number.
Red	Specifies the red color component in scale 0 to 255
Green	Specifies the green color component in scale 0 to 255
Blue	Specifies the blue color component in scale 0 to 255

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the background color of the display.
This function is called from message queue of R_WM_ScreenBgColorSet.

Default Behavior

This function executes the following processing.

- Sets the display background color by executing R_VDCE_DisplayColorSet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.5 R_WM_Sys_ScreenColorCurveSet

Function Prototype

```
uint32_t R_WM_Sys_ScreenColorCurveSet(const uint32_t                 Unit,
                                      const r_wm_ClutEntry_t  *const ColorCurve,
                                      const uint32_t                 NumEntries)
```

Input Parameter

Table 3-23 Input Parameter of R_WM_Sys_ScreenColorCurveSet

Parameter	Description
Unit	Specifies the WM Unit number.
ColorCurve	Specifies the starting address of table of reference points of type r_wm_ClutEntry_t. Alpha value of data type is unused.
NumEntries	Specifies the number of elements in ColorCurve. NumEntries should specify 33 fixed.

Input – Output Parameter

None

Output Parameter

None

Description

This function will set a curve to be used as custom gamma or color correction curve. Using this curve, each RGB color channel is individually corrected according to the given curve. For the correction in VDCE, the incoming color information of each channel is split into 32 equal segments each covering 8 color values. For these 8 values, the same gain factor applies.

To configure the segments, a start and an end value need to be given. This requires 33 reference points to be passed to this function. For each segment of each color, the gain factor must be in range [x0.25 .. x2.0], thus the values between two reference points may have a difference in range of [0 .. 16]. This function will overwrite the settings of R_WM_ScreenGammaSet.

This function is called from message queue of R_WM_ScreenColorCurveSet.

Default Behavior

This function executes the following processing.

- Calculates RGB gamma correction parameters.
- Sets the RGB gamma correction parameters by executing R_VDCE_DisplayGammaCorrectSet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.6 R_WM_Sys_ScreenGammaSet

Function Prototype

```
uint32_t R_WM_Sys_ScreenGammaSet(const uint32_t    Unit,
                                const uint8_t     GammaRed,
                                const uint8_t     GammaGreen,
                                const uint8_t     GammaBlue)
```

Input Parameter

Table 3-24 Input Parameter of R_WM_Sys_ScreenGammaSet

Parameter	Description
Unit	Specifies the WM Unit number.
GammaRed	Specifies the gamma correction factor for red.
GammaGreen	Specifies the gamma correction factor for green.
GammaBlue	Specifies the gamma correction factor for blue.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the output gamma correction.
This function will overwrite the settings of R_WM_Sys_ScreenColorCurveSet.
This function is called from message queue of R_WM_ScreenGammaSet.

Default Behavior

This function executes the following processing.

- If all the three Gamma Parameters are 128, the Gamma Correction is turned OFF by executing R_VDCE_DisplayGammaCorrectSet.
- Otherwise, calculate gamma curve and make 33 entry points and executes same process as R_WM_Sys_ScreenColorCurveSet.

Customizing Points

This function calculates $y = x^{1/\gamma}$ by the approximate expression to get gamma curve.
If user wants to improve arithmetic accuracy, please change the implement. (e.g. use powf in <math.h>).

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.7 R_WM_Sys_ScreenColorFormatSet

Function Prototype

```
uint32_t R_WM_Sys_ScreenColorFormatSet(const uint32_t             Unit,
                                       const r_wm_OutColorFmt_t    OutFmt)
```

Input Parameter

Table 3-25 Input Parameter of R_WM_Sys_ScreenColorFormatSet

Parameter	Description
Unit	Specifies the WM Unit number.
OutFmt	Specifies the color format of the video output. R_WM_OUTCOLORFMT_RGB888 R_WM_OUTCOLORFMT_RGB666 R_WM_OUTCOLORFMT_RGB565 Optional flags can be available with OR operation R_WM_OUTCOLORFMT_FLAG_ENDIAN R_WM_OUTCOLORFMT_FLAG_SWAP_BR R_WM_OUTCOLORFMT_FLAG_DITHER

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the color format of the video output signal.
This function is called from R_WM_ScreenColorFormatSet.

Default Behavior

This function executes the following processing.

- Sets the video output data endian by executing R_VDCE_DisplayOutEndianSet.
- Swaps the video output data red and blue channel by executing R_VDCE_DisplayOutSwapBR.
- Sets the video output format by executing R_VDCE_DisplayOutFormatSet.
- Sets the dither mode to truncate mode or random pattern dither by executing R_VDCE_DisplayCalibrationSet.

Customizing Points

Since it does not correspond to the Serial RGB output of VDCE, this function can be modified to support Serial RGB output.
User can modify the dither mode.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.8 R_WM_Sys_ScreenVocaInit

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaInit(const uint32_t             Unit)
```

Input Parameter

Table 3-26 Input Parameter of R_WM_Sys_ScreenVocaInit

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function setups VOCA H/W for specified WM unit.
This function is called from message queue of R_WM_ScreenVocaInit.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx
- Returns error.

RH850/D1Mx
- Gets display timing by executing R_VDCE_DisplayTimingSet.
- Check minimum horizontal front porch size.
- Calculates r_voca_Param_t parameters. It's depending on active high or active low.
- Sets display timing by executing R_VOCA_ParamSet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.9 R_WM_Sys_ScreenVocaDeInit

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaDeInit(const uint32_t              Unit)
```

Input Parameter

Table 3-27 Input Parameter of R_WM_Sys_ScreenVocaDeInit

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function disables all VOCA monitoring (Video output monitor and Activity monitor) and deletes all created VOCA monitor areas of specified WM unit.
This function is called from message queue of R_WM_ScreenVocaDeInit.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx

- Gets created video output monitor area from global flags.
- Disables all created video output monitor area by executing R_VOCA_VideoOutputCheckDisable.
- Clears global flags that shows created video output monitor area.
- Disables activity monitor by executing R_VOCA_ActiveMonitorDisable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.10 R_WM_Sys_ScreenVocaCreate

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaCreate(const uint32_t            Unit,
                                   const r_wm_Voca_t      *const Voca)
```

Input Parameter

Table 3-28 Input Parameter of R_WM_Sys_ScreenVocaCreate

Parameter	Description
Unit	Specifies the WM Unit number.
Voca	Specifies the Voca structure pointer.

Input – Output Parameter

None

Output Parameter

None

Description

This function creates a video output monitor area to specified WM unit.
This function is called from message queue of R_WM_ScreenVocaCreate.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx

- Gets created video output monitor area from global flags of another WM unit.
- Checks for the duplication of creating monitor area number.
- Sets video output monitor area parameters by executing R_VOCA_MonitorAreaSet.
- Sets expected image to internal RAM by executing R_VOCA_ColorRamSet if Voca->Size is not 0.
- Sets global flags that shows created video output monitor area.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.11 R_WM_Sys_ScreenVocaDelete

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaDelete(const uint32_t            Unit,
                                   const r_wm_Voca_t      *const Voca)
```

Input Parameter

Table 3-29 Input Parameter of R_WM_Sys_ScreenVocaDelete

Parameter	Description
Unit	Specifies the WM Unit number.
Voca	Specifies the Voca structure pointer.

Input – Output Parameter

None

Output Parameter

None

Description

This function deletes a video output monitor area from specified WM unit.
This function is called from message queue of R_WM_ScreenVocaDelete.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx

- Disables the video output monito area by executing R_WM_Sys_ScreenVocaEnable.
- Clears global flags that shows created video output monitor area.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.12 R_WM_Sys_ScreenVocaEnable

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaEnable(const uint32_t            Unit,
                                   const r_wm_Voca_t      *const Voca,
                                   const uint32_t             Enabled)
```

Input Parameter

Table 3-30 Input Parameter of R_WM_Sys_ScreenVocaEnable

Parameter	Description
Unit	Specifies the WM Unit number.
Voca	Specifies the Voca structure pointer.
Enabled	Specifies the control. 0 : Disable 1 : Enable

Input – Output Parameter

None

Output Parameter

None

Description

This function enables / disables a video output monitor area.
This function is called from message queue of R_WM_ScreenVocaEnable and R_WM_ScreenVocaDisable.
This function is also called from R_WM_Sys_ScreenVocaDelete.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx
- Returns error.

RH850/D1Mx
For Enable:
- Enables video output checker by executing R_VOCA_VideoOutputCheckEnable.

For Disable:
- Disables video output checker by executing R_VOCA_VideoOutputCheckDisable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.13 R_WM_Sys_ScreenVocaExpImgSet

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaExpImgSet(const uint32_t             Unit,
                                      const r_wm_Voca_t     *const Voca,
                                      const uint32_t                Threshold,
                                      const uint16_t                RamAddr,
                                      const uint16_t                ExpSize,
                                      const uint16_t        *const ExpImg)
```

Input Parameter

Table 3-31 Input Parameter of R_WM_Sys_ScreenVocaExpImgSet

Parameter	Description
Unit	Specifies the WM Unit number.
Voca	Specifies the Voca structure pointer.
Threshold	Acceptable mismatch (difference) of a Monitor Area.
RamAddr	Internal RAM start index to update.
ExpSize	The number of array of Data to update. If ExpSize is 0, only Threshold and RamAddr for specified monitor area is updated.
ExpImg	Pointer to update expected image array.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the expected Image to internal RAM.
This function also updates the Threshold and RamAddr for specified monitor area.
This function is called from message queue of R_WM_ScreenVocaExpImgSet.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx

- If Size is not 0, sets VOCA status flags to update internal RAM in R_WM_Sys_ScreenVocaUpdate.
- If RamAddr or Threshold is changed from previous settings, sets VOCA status flags to update VOCA registers in R_WM_Sys_ScreenVocaUpdate.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.14 R_WM_Sys_ScreenVocaClutSet

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaClutSet(const uint32_t                     Unit,
                                    const r_wm_Voca_t          *const Voca,
                                    const uint8_t                      NumEntries,
                                    const r_wm_VocaClutEntry_t *const Clut)
```

Input Parameter

Table 3-32 Input Parameter of R_WM_Sys_ScreenVocaClutSet

Parameter	Description
Unit	Specifies the WM Unit number.
Voca	Specifies the Voca structure pointer.
NumEntries	Number of CLUT entries. The range is 1 to 4.
Clut	Pointer to array of CLUT.

Input – Output Parameter

None

Output Parameter

None

Description

This function changes the CLUT data of VOCA.
This function is called from message queue of R_WM_ScreenVocaClutSet.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx
- Returns error.

RH850/D1Mx
- Sets VOCA status flags to update VOCA registers in R_WM_Sys_ScreenVocaUpdate.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.15 R_WM_Sys_ScreenActMonEnable

Function Prototype

```
uint32_t R_WM_Sys_ScreenActMonEnable(const uint32_t            Unit,
                                     const uint32_t            UpperTime,
                                     const uint32_t            LowerTime,
                                     const uint32_t            Enabled)
```

Input Parameter

Table 3-33 Input Parameter of R_WM_Sys_ScreenActMonEnable

Parameter	Description
Unit	Specifies the WM Unit number.
UpperTime	Specifies the upper detection time. Unit is micro seconds. Valid range is 0 to 136467 [usec]. This is valid when Enabled = 1.
LowerTime	Specifies the lower detection time. Unit is micro seconds. Valid range is 0 to 136467 [usec]. This is valid when Enabled = 1.
Enabled	Specifies the control. 0 : Disable 1 : Enable

Input – Output Parameter

None

Output Parameter

None

Description

This function enables / disables activity monitor of VOCA.
LowerTime and UpperTime can be set in (100/3 = 33.3…) microseconds units. WM porting layer rounds to the nearest value.
This function is called from message queue of R_WM_ScreenActivityMonEnable and R_WM_ScreenActivityMonDisable.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx
For Enable:

- Converts UpperTime and LowerTime to 33.33... microseconds unit.
- Enables video output checker by executing R_VOCA_ActiveMonitorEnable.

For Disable:

- Disables video output checker by executing R_VOCA_ActiveMonitorDisable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.3.16 R_WM_Sys_ScreenVocaUpdate

Function Prototype

```
uint32_t R_WM_Sys_ScreenVocaUpdate(const uint32_t              Unit)
```

Input Parameter

Table 3-34 Input Parameter of R_WM_Sys_ScreenVocaUpdate

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function executes jobs for VOCA in message queue.
This function is called from R_WM_FrameExecuteVoca.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns R_WM_SYS_OK.

RH850/D1Mx

- Searches video output monitor area from root of linked list. If created video output monitor area is found, performs the following process.
    - Gets register update flag from VOCA status flags. The flag is set by R_WM_ScreenVocaExpImgSet and R_WM_ScreenVocaClutSet.
    - If register update flag is on, sets the expected CRC by executing R_VOCA_MonitorAreaSet
    - Gets internal RAM update flag from the VOCA status flags. The flag is set by R_WM_ScreenVocaExpImgSet.
    - If internal RAM update flag is on, sets the expected CRC by executing R_VOCA_ColorRamSet

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4 WM Windows interface functions

3.2.4.1 R_WM_Sys_WindowSetFb

Function Prototype

```
uint32_t R_WM_Sys_WindowSetFb(const uint32_t             Unit,
                              const r_wm_Window_t *const Win,
                              const void          *const Fb)
```

Input Parameter

Table 3-35 Input Parameter of R_WM_Sys_WindowSetFb

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
Fb	Specifies the buffer start address.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the visible (front) buffer for the window.
This function is called from message queue of following functions.

- R_WM_WindowSwap
- R_WM_WindowExternalBufSet

Default Behavior

This function executes the following processing.

- Lock the layer with the global flag while updating VDCE layer functions.

Following processing is depending on window type.

Frame buffer window:

- Sets the address of frame buffer by executing R_VDCE_LayerBaseSet.

Sprite window:

- Sets the address of virtual frame buffer by executing R_VDCE_LayerBaseSet.

RLE window:

- Checks if the layer is available as RLE window.
- Checks the alignment of start address.
- Enables the RLE engine and RLE unit.
  - Sets the RLE source address, color format and ebnables the RLE engine by executing R_SPEA_SetRle.
  - Sets the update flag to process by next R_WM_Sys_DevFrameFinished.
  - If RLE unit is disabled, enables the RLE unit by executing R_SPEA_UnitEnable.
- Sets the starting address of virtual frame buffer by executing R_VDCE_LayerBaseSet.

Customizing Points

It is not necessary to modify this function in general use-case.
If user wants to move the synthesized start address for sprite window, change the definition of R_WM_SYS_VIRTUAL_ADDR_SPRITE_0 and R_WM_SYS_VIRTUAL_ADDR_SPRITE_1 value.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_DevFrameFinished

3.2.4.2 R_WM_Sys_WindowCapabilitiesSet

Function Prototype

```
uint32_t R_WM_Sys_WindowCapabilitiesSet(const r_wm_WinCapbs_t   Capability0,
                                        const r_wm_WinCapbs_t   Capability1,
                                        const r_wm_WinCapbs_t   Capability2,
                                        const r_wm_WinCapbs_t   Capability3)
```

Input Parameter

Table 3-36 Input Parameter of R_WM_Sys_WindowCapabilitiesSet

Parameter	Description
Capability0	Specifies the selectable window type for WM unit0 Layer0 and WM unit1 Layer0. R_WM_WINCAPBS_RLE R_WM_WINCAPBS_SPRITES
Capability1	Specifies the selectable window type for WM unit0 Layer1 and WM unit1 Layer3. R_WM_WINCAPBS_RLE R_WM_WINCAPBS_SPRITES
Capability2	Specifies the selectable window type for WM unit0 Layer2 and WM unit1 Layer2. R_WM_WINCAPBS_RLE R_WM_WINCAPBS_SPRITES
Capability3	Specifies the selectable window type for WM unit0 Layer3 and WM unit1 Layer1. R_WM_WINCAPBS_RLE R_WM_WINCAPBS_SPRITES

Input – Output Parameter

None

Output Parameter

None

Description

This function configures the selectable window type, RLE window or Sprite window for each layer set.
One layer of WM Unit 0 and one layer of WM Unit 1 have the same configuration.
Attention: The layer order of WM unit 1 is different from the layer order of WM unit 0.
This function is called from R_WM_WindowCapabilitiesSet.
Note that this function can be executed before the initialization.

Default Behavior

This function executes the following processing.

- Configures the selectable window type by executing R_SPEA_UnitCapabilitiesSet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.3 R_WM_Sys_WindowCreate

Function Prototype

```
uint32_t R_WM_Sys_WindowCreate(const uint32_t              Unit,
                               const r_wm_Window_t *const   Win)
```

Input Parameter

Table 3-37 Input Parameter of R_WM_Sys_WindowCreate

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure. This argument is not referred directly. The same value will be obtained from window list created in advance.

Input – Output Parameter

None

Output Parameter

None

Description

This function reconfigures all windows of the specified WM unit.
This function is called from message queue of R_WM_WindowCreate.

Default Behavior

This function executes the following processing.

- Gets the target window from root window.
- Loop from Layer0 to Layer3 and executes the following processed.
  - Disables the layer by executing R_VDCE_LayerDisable.
  - Checks if the window can be created on the layer. If it can be created, executes the following processed.
    - Lock the layer with the global flag while updating VDCE layer functions.
    - If Chromakey is enabled:
      - Disables the constant alpha by executing R_VDCE_LayerAlphaConstDisable.
      - Enables the chromakey by executing R_VDCE_LayerChromaKeyEnable
    - If Chromakey is disabled:
      - Disables the chromakey by executing R_VDCE_LayerChromaKeyDisable
      - Enables the constant alpha by executing R_VDCE_LayerAlphaConstEnable.
    - Sets color format by executing R_VDCE_LayerFormatSet.
    - Sets the update flag to process by next R_WM_Sys_DevFrameFinished if CLUT is enabled.
    - Sets the scaling-up/down by executing R_VDCE_LayerImgScaleX and R_VDCE_LayerImgScaleY.
    - Enables alpha channel by executing R_VDCE_LayerAlphaChannelEnable.
    - Enables / Disables pre-multiple alpha channel by executing R_VDCE_LayerPremultipliedAlphaEnable / R_VDCE_LayerPremultipliedAlphaDisable.
    - Sets vertical mirroring by executing R_VDCE_LayerModeSet.
    - Gets the On-screen frame buffer and executes same process as R_WM_Sys_WindowSetFb to set the start address of the frame buffer.
    - Sets the re-assigned flag to process by next R_WM_Sys_DevFrameFinished in case of Sprite window.
    - Sets the layers memory geometry by executing R_VDCE_LayerMemGeometrySet.
    - Sets the layers viewport parameters by R_VDCE_LayerViewPortSet.
    - Enables the layer by executing R_VDCE_LayerEnable if the layer is already enabled before this sequence is executed.
    - Get next window from the window list.

- An error is returned if the layer to which the window can be assigned is not found.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_DevFrameFinished
R_WM_Sys_WindowSetFb

3.2.4.4 R_WM_Sys_WindowDelete

Function Prototype

```
uint32_t R_WM_Sys_WindowDelete(const uint32_t            Unit,
                               const r_wm_Window_t *const Win)
```

Input Parameter

Table 3-38 Input Parameter of R_WM_Sys_WindowDelete

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function deletes a window.
This function is called from message queue of R_WM_WindowDelete.

Default Behavior

This function executes the following processing.

- Disables the specified layer by executing the function R_VDCE_LayerDisable, if that layer is in enabled.
- If the window is RLE window:
  - Disables the RLE engine by R_SPEA_SetRle.
  - Disables the RLE unit by R_SPEA_UnitEnable if shared RLE unit is already disabled.
  - Sets the update flag to process by next R_WM_Sys_DevFrameFinished.
- Update the global variables to manage the layer.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.5 R_WM_Sys_WindowEnable

Function Prototype

```
uint32_t R_WM_Sys_WindowEnable(const uint32_t             Unit,
                               const r_wm_Window_t  *const Win,
                               const uint32_t             Enabled)
```

Input Parameter

Table 3-39 Input Parameter of R_WM_Sys_WindowEnable

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
Enabled	Specifies the control 1 : Enabled 0 : Disabled

Input – Output Parameter

None

Output Parameter

None

Description

This function will enable/disable the window.
If the window position is outside of the screen, this function does not enable the window.
This function is called from message queue of R_WM_WindowEnable.

Default Behavior

This function executes the following processing.
For Enable:

- Checks if the window position is outside of the screen.
- Enables the specified layer by executing the function R_VDCE_LayerEnable.

For Disable:

- Disables the specified layer by executing the function R_VDCE_LayerDisable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.6 R_WM_Sys_WindowPosSet

Function Prototype

```
uint32_t R_WM_Sys_WindowPosSet(const uint32_t              Unit,
                               const r_wm_Window_t   *const Win,
                               const int32_t               PosX,
                               const int32_t               PosY,
                               const int32_t               PosZ)
```

Input Parameter

Table 3-40 Input Parameter of R_WM_Sys_WindowPosSet

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
PosX	Specifies the new X position of the window on the screen.
PosY	Specifies the new Y position of the window on the screen.
PosZ	Specifies the new Z position of the window on the screen.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets window position as specified in the Paramets PosX & PosY.
Also sets the layer position as specified in parameter PosZ
This function is called from message queue of R_WM_WindowMove.

Default Behavior

This function executes the following processing.

- Lock the layer with the global flag while updating VDCE layer functions.
- When PosZ is changed from previous value, all window are re-constructed and changes layer assignment of the Windows. Executes same behavior as R_WM_Sys_WindowCreate.
- Set the layers memory geometry by executing R_VDCE_LayerMemGeometrySet.
- Set the layers viewport parameters by R_VDCE_LayerViewPortSet.
- If the layer is already enabled, following
  - If the window position is outside of screen, disables the layer by executing R_VDCE_LayerDisable.
  - If the window position is inside of screen, enables the layer by executing R_VDCE_LayerEnable.
- Sets the starting address of frame buffer by executing the function R_VDCE_LayerBaseSet
- Gets the On-screen frame buffer and executes same process as R_WM_Sys_WindowSetFb to set the start address of the frame buffer.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_WindowSetFb
R_WM_Sys_WindowCreate

3.2.4.7 R_WM_Sys_WindowGeomSet

Function Prototype

```
uint32_t R_WM_Sys_WindowGeomSet(const uint32_t             Unit,
                                const r_wm_Window_t *const  Win,
                                const uint32_t             Pitch,
                                const uint32_t             Width,
                                const uint32_t             Height)
```

Input Parameter

Table 3-41 Input Parameter of R_WM_Sys_WindowGeomSet

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
Pitch	Specifies the distance in pixels between subsequent rows in the frame buffer memory (>= Width)
Width	Specifies the window frame buffer width in pixels
Height	Specifies the window frame buffer height in pixels

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the window dimensions.
This function is called from message queue of R_WM_WindowResize.

Default Behavior

This function executes the following processing.

- Lock the layer with the global flag while updating VDCE layer functions.
- Disables the layer by executing R_VDCE_LayerDisable if the layer is enabled.
- Sets the layers memory geometry by executing R_VDCE_LayerMemGeometrySet.
- Sets the layers viewport parameters by R_VDCE_LayerViewPortSet.
- Sets the layers scaling parameters by R_VDCE_LayerImgScaleX and R_VDCE_LayerImgScaleY.
- Enables the layer by executing R_VDCE_LayerEnable if the layer is already enabled before this function is called.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.8 R_WM_Sys_WindowColorFmtSet

Function Prototype

```
uint32_t R_WM_Sys_WindowColorFmtSet(const uint32_t               Unit,
                                    const r_wm_Window_t   *const Win,
                                    const r_wm_WinColorFmt_t     ColorFmt)
```

Input Parameter

Table 3-42 Input Parameter of R_WM_Sys_WindowColorFmtSet

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
ColorFmt	Specifies the color Format of the window R_WM_COLORFMT_RGB565 R_WM_COLORFMT_ARGB1555 R_WM_COLORFMT_ARGB4444 R_WM_COLORFMT_RGB0888 R_WM_COLORFMT_ARGB8888 R_WM_COLORFMT_RGBA5551 R_WM_COLORFMT_RGBA8888 R_WM_COLORFMT_CLUT8 R_WM_COLORFMT_CLUT4 R_WM_COLORFMT_CLUT1 R_WM_COLORFMT_RLE24ARGB8888 R_WM_COLORFMT_RLE24RGB0888 R_WM_COLORFMT_YCBCR_422 R_WM_COLORFMT_YCBCR_444 R_WM_COLORFMT_YUV_YUYV R_WM_COLORFMT_YUV_UYVY R_WM_COLORFMT_YUV_YVYU R_WM_COLORFMT_YUV_VYUY

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the color format.
This function is called from message queue of R_WM_WindowColorFmtSet.

Default Behavior

This function executes the following processing.

- When window type is sprite window and bpp of color format is changed from previous setting, sets the layer memory geometry by executing R_VDCE_LayerMemGeometrySet.
- Disables the layer by executing R_VDCE_LayerDisable if the layer is enabled.
- Set the color format by executing R_VDCE_LayerFormatSet.
- Enables the layer by executing R_VDCE_LayerEnable if the layer is already enabled before this function is called.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.9 R_WM_Sys_WindowAlphaSet

Function Prototype

```
uint32_t R_WM_Sys_WindowAlphaSet(const uint32_t              Unit,
                                 const r_wm_Window_t   *const  Win,
                                 const uint8_t                 Alpha)
```

Input Parameter

Table 3-43 Input Parameter of R_WM_Sys_WindowAlphaSet

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
Alpha	Specifies the constant alpha value.

Input – Output Parameter

None

Output Parameter

None

Description

This function will set the constant alpha for the window.
If specified window is assigned to layer 0, this function does nothing.
This function is called from message queue of R_WM_WindowAlphaSet.

Default Behavior

This function executes the following processing.

- Returns the error when the Chromakey is Enabled.
- Enables the layers alpha constant by executing the function R_VDCE_LayerAlphaConstEnable

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.10 R_WM_Sys_WindowPremultipliedAlphaEnable

Function Prototype

```
uint32_t R_WM_Sys_WindowPremultipliedAlphaEnable(const uint32_t            Unit,
                                                 const r_wm_Window_t *const Win,
                                                 const uint8_t             Enabled)
```

Input Parameter

Table 3-44 Input Parameter of R_WM_Sys_WindowPremultipliedAlphaEnable

Parameter	Description
Unit	Specifies the WM Unit number
Win	Specifies the window structure.
Enabled	Specifies the control 1 : Enabled 0 : Disabled

Input – Output Parameter

None

Output Parameter

None

Description

This function enables or disables the pre-multiplied alpha mode for the window.
If specified window is assigned to layer 0, this function does nothing.
This function is called from message queue of following functions.
- R_WM_WindowPremultipliedAlphaEnable
- R_WM_WindowPremultipliedAlphaDisable.

Default Behavior

This function executes the following processing.
- Disable the layer by executing R_VDCE_LayerDisable if the layer is enabled.
- If Enabled is 1, enables the pre-multiplied alpha mode by executing R_VDCE_LayerPremultipliedAlphaEnable.
- If Enabled is 0, disables the pre-multiplied alpha mode by executing R_VDCE_LayerPremultipliedAlphaDisable.
- Enables the layer by executing R_VDCE_LayerEnable if the layer is already enabled before this function is called.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.11 R_WM_Sys_WindowFlagsUpdate

Function Prototype

```
uint32_t R_WM_Sys_WindowFlagsUpdate(const uint32_t              Unit,
                                    const r_wm_Window_t   *const Win,
                                    const r_wm_WinFlags_t        SetFlags,
                                    const r_wm_WinFlags_t        ClearFlags)
```

Input Parameter

Table 3-45 Input Parameter of R_WM_Sys_WindowFlagsUpdate

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
SetFlags	Specifies the new flags to be set R_WM_WINFLAG_NONE R_WM_WINFLAG_V_MIRROR
ClearFlags	Specifies the new flags to be cleared R_WM_WINFLAG_NONE R_WM_WINFLAG_V_MIRROR

Input – Output Parameter

None

Output Parameter

None

Description

This function will provide various ON/OFF switches for different functionalities of window.
This function is called from message queue of following functions.

- R_WM_WindowVerticalMirrorEnable
- R_WM_WindowVerticalMirrorDisable.

Default Behavior

This function executes the following processing.

- Lock the layer with the global flag while updating VDCE layer functions.
- Disables the layer by executing R_VDCE_LayerDisable if the layer is enabled.
- If the SetFlags = R_WM_WINFLAG_V_MIRROR, enable the vertical mirroring by executing R_VDCE_LayerModeSet.
- If the ClearFlags = R_WM_WINFLAG_V_MIRROR , disable the vertical mirroring by executing R_VDCE_LayerModeSet.
- Enables the layer by executing R_VDCE_LayerEnable if the layer is already enabled before this function is called.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.12 R_WM_Sys_WindowClutSet

Function Prototype

```
uint32_t R_WM_Sys_WindowClutSet(const uint32_t              Unit,
                                const r_wm_Window_t *const    Win,
                                const uint32_t              NumEntries,
                                const r_wm_ClutEntry_t *const Clut)
```

Input Parameter

Table 3-46 Input Parameter of R_WM_Sys_WindowClutSet

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
NumEntries	Specifies the number of color lookup-table entries
Clut	Specifies the starting address of an array of type r_wm_ClutEntry_t.

Input – Output Parameter

None

Output Parameter

None

Description

This function will set the color lookup-table for the window.
This function is called from message queue of R_WM_WindowClutSet.
The parameter NumEntires and Clut should be updated to Window structure before this function is called.

Default Behavior

This function executes the following processing.

- If specified window is assigned to Layer 0 and one of alpha value of CLUT is not 0xFF (Clut[n].A != 0xFF), all windows will be re-constructed and changes layer assignment of the Windows. Executes same behavior as R_WM_Sys_WindowCreate.
- Sets the update flag to process by next R_WM_Sys_DevFrameFinished.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_WindowCreate
R_WM_Sys_DevFrameFinished

3.2.4.13 R_WM_Sys_WindowColorKeyEnable

Function Prototype

```
uint32_t R_WM_Sys_WindowColorKeyEnable(const uint32_t            Unit,
                                       const r_wm_Window_t *const Win,
                                       const uint32_t            Enabled)
```

Input Parameter

Table 3-47 Input Parameter of R_WM_Sys_WindowColorKeyEnable

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
Enabled	Specifies the control. 1 : Enabled 0 : Disabled

Input – Output Parameter

None

Output Parameter

None

Description

This function enable or disables the chroma keying for the window.
This function returns error if specified window is assigned to Layer0.
This function is called from message queue of following functions.

- R_WM_WindowColorKeyEnable
- R_WM_WindowColorKeyDisable

Default Behavior

This function executes the following processing.

For Enable:

- Disables the constant alpha by executing R_VDCE_LayerAlphaConstDisable.
- Enables the chroma keying by executing R_VDCE_LayerChromaKeyEnable.

For Disable:

- Disables the chroma keying by executing R_VDCE_LayerChromaKeyDisable.
- Enables the alpha constant by executing R_VDCE_LayerAlphaConstEnable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.4.14 R_WM_Sys_WindowScaleSet

Function Prototype

```
uint32_t R_WM_Sys_WindowScaleSet(const uint32_t            Unit,
                                 const r_wm_Window_t *const Win,
                                 const r_wm_ScaleChg_t      ChangeMode,
                                 const uint32_t             ScaledWidth,
                                 const uint32_t             ScaledHeight)
```

Input Parameter

Table 3-48 Input Parameter of R_WM_Sys_WindowScaleSet

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.
ChangeMode	Specifies the change behavior. R_WM_SCALE_CHANGE_SMALL R_WM_SCALE_CHANGE_LARGE
ScaledWidth	Specifies the horizontal scaling-down or scaling-up size. If horizontal scaling is disabled, set to 0.
ScaledHeight	Specifies the vertical scaling-down or scaling-up size. If vertical scaling is disabled, set to 0.

Input – Output Parameter

None

Output Parameter

None

Description

This function changes the scaling mode and scaled size for the window.
This function is called from message queue of following functions.

- R_WM_WindowScaledSizeSet
- R_WM_CaptureScaledSizeSet

Default Behavior

This function executes the following processing.

- Lock the layer with the global flag while updating VDCE layer functions.
- Changes the horizontal scaling behavior and scaling size according to following table by executing R_VDCE_LayerImgScaleX.
- Changes the vertical scaling behavior and scaling size according to following table by executing R_VDCE_LayerImgScaleY.
- If both scaling-down and scaling-up are enabled, only scaling-down is effective.

Table 3-49 Decision Table of R_WM_Sys_WindowScaleSet

Parameter setting		before status			after status		
ChangeMode	ScaledWidth or ScaledHeight	scaling-down size	scaling-up size	behavior	scaling-down size	scaling-up size	behavior
SMALL	n	0	0	no-scaling	n	0	scaling-down
		0	u	scaling-up	n	u	scaling-down
		d	0	scaling-down	n	0	scaling-down
		d	u	scaling-down	n	u	scaling-down
SMALL	0	0	0	no-scaling	0	0	no-scaling
		0	u	scaling-up	0	u	scaling-up
		d	0	scaling-down	0	0	no-scaling
		d	u	scaling-down	0	u	scaling-up
LARGE	n	0	0	no-scaling	0	n	scaling-up
		0	u	scaling-up	0	n	scaling-up
		d	0	scaling-down	d	n	scaling-down
		d	u	scaling-down	d	n	scaling-down
LARGE	0	0	0	no-scaling	0	0	no-scaling
		0	u	scaling-up	0	0	no-scaling
		d	0	scaling-down	d	0	scaling-down
		d	u	scaling-down	d	0	scaling-down

d, u : non zero value set in advance
n : non zero new setting value

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

r_wm_ScaleChg_t

3.2.5 WM Sprite interface functions

3.2.5.1 R_WM_Sys_SpriteCreate

Function Prototype

```
uint32_t R_WM_Sys_SpriteCreate(const uint32_t             Unit,
                               const r_wm_Sprite_t *const Sprite)
```

Input Parameter

Table 3-50 Input Parameter of R_WM_Sys_SpriteCreate

Parameter	Description
Unit	Specifies the WM Unit number.
Sprite	Specifies the Sprite structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function will add a sprite data to the sprite window.
This function is called from message queue of R_WM_SpriteCreate.

Default Behavior

This function executes the following processing.

- Checks if Sprite->PosY and (Sprite->PosY + Sprite->Height) are within range.
- Sets the re-assigned flag to process by next R_WM_Sys_DevFrameFinished.
- Execute R_WM_Sys_SpriteEnable to disable the sprite data as default.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_SpriteEnable
R_WM_Sys_DevFrameFinished

3.2.5.2 R_WM_Sys_SpriteEnable

Function Prototype

```
uint32_t R_WM_Sys_SpriteEnable(const uint32_t              Unit,
                               const r_wm_Sprite_t  *const Sprite,
                               const uint32_t              Enabled)
```

Input Parameter

Table 3-51 Input Parameter of R_WM_Sys_SpriteEnable

Parameter	Description
Unit	Specifies the WM Unit number.
Sprite	Specifies the Sprite structure.
Enabled	Specifies the control. 1 : Enabled 0 : Disabled

Input – Output Parameter

None

Output Parameter

None

Description

This function enables or disables the sprite data.
This function is called from message queue of following functions.

- R_WM_SpriteEnable
- R_WM_SpriteDisable

Default Behavior

This function executes the following processing.

- Gets the sprite index from sprite window list in PosZ order. If valid sprite index is not obtained, this function returns error.
- Enables or disables the sprite data by R_SPEA_SpriteEnable.
- Sets the re-assigned flag to process by next R_WM_Sys_DevFrameFinished.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_DevFrameFinished

3.2.5.3 R_WM_Sys_SpriteDelete

Function Prototype

```
uint32_t R_WM_Sys_SpriteDelete(const uint32_t             Unit,
                               const r_wm_Sprite_t *const Sprite)
```

Input Parameter

Table 3-52 Input Parameter of R_WM_Sys_SpriteDelete

Parameter	Description
Unit	Specifies the WM Unit number.
Sprite	Specifies the Sprite structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function deletes the sprite from the window.
This function is called from message queue of R_WM_SpriteDelete.

Default Behavior

This function executes the following processing.

- Sets the re-assigned flag to process by next R_WM_Sys_DevFrameFinished.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_DevFrameFinished

3.2.5.4 R_WM_Sys_SpriteMove

Function Prototype

```
uint32_t R_WM_Sys_SpriteMove(const uint32_t             Unit,
                             const r_wm_Sprite_t    *const Sprite,
                             const uint32_t              PosX,
                             const uint32_t              PosY,
                             const uint32_t              PosZ)
```

Input Parameter

Table 3-53 Input Parameter of R_WM_Sys_SpriteMove

Parameter	Description
Unit	Specifies the WM Unit number.
Sprite	Specifies the Sprite structure.
PosX	Specifies the new sprite X position.
PosY	Specifies the new sprite Y position.
PosZ	Specifies the new sprite Z position.

Input – Output Parameter

None

Output Parameter

None

Description

This function moves the sprite data on the sprite window.
This function is called from message queue of R_WM_SpriteMove.

Default Behavior

This function executes the following processing.

- Checks if Sprite->PosY and (Sprite->PosY + Sprite->Height) are within range.
- If Unit is 1, adds the offset to PosY value.
- Changes the sprite position by R_SPEA_SetSpritePos.
- Sets the re-assigned flag to process by next R_WM_Sys_DevFrameFinished if PosZ is changed.
- Sets the update flag to process by next R_WM_Sys_DevFrameFinished

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.5.5 R_WM_Sys_SpriteBufSet

Function Prototype

```
uint32_t R_WM_Sys_SpriteBufSet(const uint32_t            Unit,
                               const r_wm_Sprite_t *const Sprite,
                               const void          *const Buf)
```

Input Parameter

Table 3-54 Input Parameter of R_WM_Sys_SpriteBufSet

Parameter	Description
Unit	Specifies the WM Unit number.
Sprite	Specifies the Sprite structure.
Buf	Specifies the start address of the source data.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the sprite source data address.
This function is called from message queue of R_WM_SpriteBufSet.

Default Behavior

This function executes the following processing.

- Sets sprite data source address by executing R_SPEA_SetSprite.
- Sets the update flag to process by next R_WM_Sys_DevFrameFinished

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_DevFrameFinished

3.2.5.6 R_WM_Sys_WindowDeleteAllSprites

Function Prototype

```
uint32_t R_WM_Sys_WindowDeleteAllSprites(const uint32_t            Unit,
                                         const r_wm_Window_t *const Win)
```

Input Parameter

Table 3-55 Input Parameter of R_WM_Sys_WindowDeleteAllSprites

Parameter	Description
Unit	Specifies the WM Unit number.
Win	Specifies the window structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function deletes all sprite data on the window.
This function is called from message queue of R_WM_WindowDeleteAllSprites.

Default Behavior

This function executes the following processing.

- Disables the all sprite data on the window by executing R_SPEA_SpriteEnable.
- Sets the re-assigned flag to process by next R_WM_Sys_DevFrameFinished.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_DevFrameFinished

3.2.6 Video Capture interface functions

3.2.6.1 R_WM_Sys_CaptureCreate

Function Prototype

```
uint32_t R_WM_Sys_CaptureCreate(const uint32_t             Unit,
                                const r_wm_Capture_t *const Capt)
```

Input Parameter

Table 3-56 Input Parameter of R_WM_Sys_CaptureCreate

Parameter	Description
Unit	Specifies the WM Unit number.
Capt	Specifies the r_wm_Capture_t structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function creates a video capture surface.
This function is called from message queue of R_WM_CaptureCreate.

Default Behavior

This function executes the following processing.

- All windows are re-constructed and changes layer assignment of the Windows. Executes same behavior as R_WM_Sys_WindowCreate.
- Sets VDCECTL register depending on Input video format.
- Sets the capturing mode by executing R_VDCE_CapModeSet.

If R_WM_CAPMODE_SYNC_ONLY is not set;

- Sets capture buffer and viewport by executing R_VDCE_CapBufGeometrySetup.
- Sets writing rate and field mode by executing R_VDCE_CapRateSet.
- Sets global status flag to BOB deinterlace mode or not.
- Sets the Vsync delay by executing R_VDCE_LayerVSyncDelaySet.
- Sets color matrix by executing R_VDCE_LayerMatrixBT601Set.

Customizing Points

- User can modify VDCECTL setting and color matrix selection.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_WindowCreate

3.2.6.2 R_WM_Sys_CaptureDelete

Function Prototype

```
uint32_t R_WM_Sys_CaptureDelete(const uint32_t             Unit,
                                const r_wm_Capture_t *const Capt)
```

Input Parameter

Table 3-57 Input Parameter of R_WM_Sys_CaptureDelete

Parameter	Description
Unit	Specifies the WM Unit number.
Capt	Specifies the r_wm_Capture_t structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function deletes a video capture surface.
This function is called from message queue of R_WM_CaptureDelete.

Default Behavior

This function executes the following processing

- Disables the video capturing by executing R_WM_Sys_CaptureEnable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

R_WM_Sys_CaptureEnable

3.2.6.3 R_WM_Sys_CaptureEnable

Function Prototype

```
uint32_t R_WM_Sys_CaptureEnable(const uint32_t            Unit,
                                const r_wm_Capture_t *const Cap,
                                const uint32_t            Enabled)
```

Input Parameter

Table 3-58 Input Parameter of R_WM_Sys_CaptureEnable

Parameter	Description
Unit	Specifies the WM Unit number.
Cap	Specifies the r_wm_Capture_t structure.
Enabled	Specifies the control. 1 : Enabled 0 : Disabled

Input – Output Parameter

None

Output Parameter

None

Description

This function enables or disables the capture surface.
This function is called from message queue of following functions.

- R_WM_CaptureEnable
- R_WM_CaptureDisable

Default Behavior

This function executes the following processing.

For Enable:

- Sets callback of R_VDCE_INTC_CAP_VBLANK event by executing R_VDCE_IntcCallbackSet.
- Enables the capture window by executing R_VDCE_CapEnable.
- Enables the R_VDCE_INTC_CAP_VBLANK interrupt by executing R_VDCE_IntcEnable.
- Updates global flag to manage capturing status.

For Disable:

- Disables the R_VDCE_INTC_CAP_VBLANK interrupt by executing R_VDCE_IntcDisable.
- Disables the capture window by executing R_VDCE_CapDisable
- Updates global flag to manage capturing status.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.6.4 R_WM_Sys_CaptureViewPortSet

Function Prototype

```
uint32_t R_WM_Sys_CaptureViewPortSet(const uint32_t              Unit,
                                    const r_wm_Capture_t *const Cap,
                                    const uint32_t              StartX,
                                    const uint32_t              StartY,
                                    const uint32_t              Width,
                                    const uint32_t              Height)
```

Input Parameter

Table 3-59 Input Parameter of R_WM_Sys_CaptureViewPortSet

Parameter	Description
Unit	Specifies the WM Unit number.
Cap	Specifies the r_wm_Capture_t structure.
StartX	X position of capturing start. Unit is pixels.
StartY	Y position of capturing start. Unit is pixels.
Width	Width of capturing video data. Unit is pixels.
Height	Height of capturing video data. Unit is pixels.

Input – Output Parameter

None

Output Parameter

None

Description

This function changes the capturing position and size.
This function is called from message queue of following functions.

- R_WM_CaptureMove
- R_WM_CaptureResize

Default Behavior

This function executes the following processing.

- Changes the capture position and size by executing R_VDCE_CapViewPortSet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.6.5 R_WM_Sys_CaptureExtVsyncSet

Function Prototype

```
uint32_t R_WM_Sys_CaptureExtVsyncSet(const uint32_t            Unit,
                                     const r_wm_Capture_t *const Cap,
                                     const uint16_t            HsyncCycle,
                                     const uint32_t            VsyncMaskUs,
                                     const uint32_t            VsyncLackUs)
```

Input Parameter

Table 3-60 Input Parameter of R_WM_Sys_CaptureExtVsyncSet

Parameter	Description
Unit	Specifies the WM Unit number.
Cap	Specifies the r_wm_Capture_t structure.
HsyncCycle	Horizontal cycle of input signal. Unit is cycle (pixel).
VsyncMaskUs	Prevent Vsync coming faster than VsyncMaskUs [usec].
VsyncLackUs	Compensate Vsync coming slower than VsyncLackUs [usec].

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the Hsync cycle of input signal and Vsync protection.
This function is called from message queue of following functions.

- R_WM_CaptureExtVsyncSet

Default Behavior

This function executes the following processing.

- Sets the Hsync cycle of input signal and Vsync protection by executing R_VDCE_CapExtVsyncSet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.7 WM Messaging interface functions

3.2.7.1 R_WM_Sys_MsgQueueSetup

Function Prototype

```
uint32_t R_WM_Sys_MsgQueueSetup(const uint32_t       Unit,
                                void         *const   MsgQueueStorage,
                                const uint32_t       Size)
```

Input Parameter

Table 3-61 Input Parameter of R_WM_Sys_MsgQueueSetup

Parameter	Description
Unit	Specifies the WM Unit number.
MsgQueueStorage	Specifies the storage for the message queue.
Size	Specifies the number of elements in the message queue

Input – Output Parameter

None

Output Parameter

None

Description

This function setups the message queue.
This function is called from R_WM_DevInit.

Default Behavior

This function executes the following processing.

- Sets ring buffer for message queue by executing R_CDI_RbSetup.

Customizing Points

It is necessary to modify this function when not using CDI.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.7.2 R_WM_Sys_MsgQueueRead

Function Prototype

```
uint32_t R_WM_Sys_MsgQueueRead(const uint32_t        Unit,
                               r_wm_Msg_t     *const Msg)
```

Input Parameter

Table 3-62 Input Parameter of R_WM_Sys_MsgQueueRead

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

Table 3-63 Output Parameter of R_WM_Sys_MsgQueueRead

Parameter	Description
Msg	Read message

Description

This function reads one message and removes it from the queue.
If message queue is empty, returns error R_WM_ERR_NG.

This function is called from R_WM_FrameExecuteNext.

Default Behavior

This function executes the following processing.
- Read the data from the message queue by executing R_CDI_RbRead.

Customizing Points

It is necessary to modify this function when not using CDI.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.7.3 R_WM_Sys_MsgQueueWrite

Function Prototype

```
uint32_t R_WM_Sys_MsgQueueWrite(const uint32_t     Unit,
                                r_wm_Msg_t  *const Msg)
```

Input Parameter

Table 3-64 Input Parameter of R_WM_Sys_MsgQueueWrite

Parameter	Description
Unit	Specifies the WM Unit number.
Msg	Specifies the message

Input – Output Parameter

None

Output Parameter

None

Description

This function will write one message to the queue.
If message queue is full, returns error R_WM_ERR_NG.
This function is called from several functions.

Default Behavior

This function executes the following processing.

- Writes the values to the message queue by executing R_CDI_RbWrite.

Customizing Points

It is necessary to modify this function when not using CDI.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.8 WM Memory functions

3.2.8.1 R_WM_Sys_Heap_Set

Function Prototype

```
void R_WM_Sys_Heap_Set(void      *const Cpu,
                       void      *const Video)
```

Input Parameter

Table 3-65 Input Parameter of R_WM_Sys_Heap_Set

Parameter	Description
Cpu	Specifies the descriptor of the Heap memory to allocate in CPU RAM. As the default implementation, pointer of r_cdi_Heap_t structure is specified. This can be NULL.
Video	Specifies the descriptor of the Heap memory to allocate in Video RAM. As the default implementation, pointer of r_cdi_Heap_t structure is specified. This can be NULL.

Input – Output Parameter

None

Output Parameter

None

Description

This sets the descriptor of the heap memory.
This function is called from R_WM_DevInit.

If user uses only external mode as frame buffer allocation mode, the heap memory is not required.
Then, the arguments can set to R_NULL.

Default Behavior

This function executes the following processing.

- The referenced heaps are stored porting layer-internal.

Customizing Points

If a custom allocator is used, update this function accordingly.

Return Codes

None

See also

None

3.2.8.2 R_WM_Sys_Alloc

Function Prototype

```
void *R_WM_Sys_Alloc(const uint32_t        Size,
                     const r_wm_Memory_t    MemType)
```

Input Parameter

Table 3-66 Input Parameter of R_WM_Sys_Alloc

Parameter	Description
Size	Specifies the size to allocate.
MemType	Specifies the type of the memory. R_WM_MEM_CPU R_WM_MEM_VIDEO

Input – Output Parameter

None

Output Parameter

None

Description

This function allocates memory from heap memory.
This function is called from R_WM_WindowCreate when the frame buffer allocation mode is internal mode.
In case of external mode, this function is not called.

Default Behavior

This function executes the following processing.

- Allocate one or more blocks of the given heap with the size of memory requested by executing the function R_CDI_Alloc.

Customizing Points

If a custom allocator is used, update this function accordingly.

Return Codes

not R_NULL	- Pointer to the allocated memory.
R_NULL	- Memory allocation failed.

See also

r_wm_Memory_t

3.2.8.3 R_WM_Sys_Free

Function Prototype

```
uint32_t R_WM_Sys_Free(const void *const       Memory,
                       const r_wm_Memory_t     MemType)
```

Input Parameter

Table 3-67 Input Parameter of R_WM_Sys_Free

Parameter	Description
Memory	Specifies the Memory pointer
MemType	Specifies the type of the memory.

Input – Output Parameter

None

Output Parameter

None

Description

This function frees memory allocated with R_WM_Sys_Alloc.
This function is called from R_WM_WindowCreate.
This function is also called from message queue of following functions.

- R_WM_WindowDelete
- R_WM_WindowExternalBufSet

Default Behavior

This function executes the following processing.

- Frees allocated memory by executing the function R_CDI_Free .

Customizing Points

If a custom allocator is used, update this function accordingly.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

r_wm_Memory_t

3.2.9 WM Discom interface functions

3.2.9.1 R_WM_Sys_DsicomCreate

Function Prototype

```
uint32_t R_WM_Sys_DiscomCreate(const uint32_t             Unit,
                               const r_wm_Discom_t  *const Discom)
```

Input Parameter

Table 3-68 Input Parameter of R_WM_Sys_DiscomCreate

Parameter	Description
Unit	Specifies the WM Unit number.
Discom	Specifies the r_wm_Discom_t structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function creates a DISCOM device to specified WM unit.
This function is called from message queue of R_WM_DiscomCreate.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx

- Checks the consistency between WM Unit and Discom->DiscomUnit.
- Sets start position and length by executing R_DISCOM_ParamSet.
- Sets expected CRC value by executing R_DISCOM_CrcSet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.9.2 R_WM_Sys_DiscomDelete

Function Prototype

```
uint32_t R_WM_Sys_DiscomDelete(const uint32_t             Unit,
                                const r_wm_Discom_t  *const Discom)
```

Input Parameter

Table 3-69 Input Parameter of R_WM_Sys_DiscomDelete

Parameter	Description
Unit	Specifies the WM Unit number.
Discom	Specifies the r_wm_Discom_t structure.

Input – Output Parameter

None

Output Parameter

None

Description

This function deletes a DISCOM device from specified WM unit.
This function is called from message queue of R_WM_DiscomDelete.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx

- Disables DISCOM device by R_WM_Sys_DiscomEnable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.9.3 R_WM_Sys_DiscomEnable

Function Prototype

```
uint32_t R_WM_Sys_DiscomEnable(const uint32_t             Unit,
                               const r_wm_Discom_t  *const Discom ,
                               const uint32_t             Enabled)
```

Input Parameter

Table 3-70 Input Parameter of R_WM_Sys_DiscomEnable

Parameter	Description
Unit	Specifies the WM Unit number.
Discom	Specifies the r_wm_Discom_t structure.
Enabled	Specifies the control. 1 : Enabled 0 : Disabled

Input – Output Parameter

None

Output Parameter

None

Description

This function enables / disables a DISCOM device.
This function is called from message queue of R_WM_DiscomEnable and R_WM_DiscomDisable.
This function is also called from R_WM_Sys_DiscomDelete.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx
For Enable:

- Enables DISCOM interrupts by executing R_DISCOM_IntEnable.
- Enables calculation and comparator by executing R_DISCOM_Enable.

For Disable:

- Disables calculation and comparator by executing R_DISCOM_Disable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.9.4 R_WM_Sys_DiscomCrcSet

Function Prototype

```
uint32_t R_WM_Sys_DiscomCrcSet(const uint32_t              Unit,
                               const r_wm_Discom_t  *const Discom ,
                               const uint32_t              ExpCrc)
```

Input Parameter

Table 3-71 Input Parameter of R_WM_Sys_DiscomCrcSet

Parameter	Description
Unit	Specifies the WM Unit number.
Discom	Specifies the r_wm_Discom_t structure.
ExpCrc	Specifies the expected CRC value.

Input – Output Parameter

None

Output Parameter

None

Description

This function changes expected CRC to DISCOM device.
This function is called from message queue of R_WM_DiscomCrcSet.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx

- Returns error.

RH850/D1Mx

- Sets global flags to update CRC in R_WM_Sys_DiscomUpdate.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.9.5 R_WM_Sys_DiscomCrcGet

Function Prototype

```
uint32_t R_WM_Sys_DiscomCrcGet(const uint32_t              Unit,
                               const r_wm_Discom_t  *const Discom ,
                               uint32_t             *const Crc)
```

Input Parameter

Table 3-72 Input Parameter of R_WM_Sys_DiscomCrcGet

Parameter	Description
Unit	Specifies the WM Unit number.
Discom	Specifies the r_wm_Discom_t structure.

Input – Output Parameter

None

Output Parameter

Table 3-73 Output Parameter of R_WM_Sys_DiscomCrcGet

Parameter	Description
Crc	The latest calculated CRC value.

Description

This function gets the latest calculated CRC value.
This function is called from R_WM_DiscomCrcGet.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx
- Returns error.

RH850/D1Mx
- Gets current CRC value by executing R_DISCOM_CrcGet.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.2.9.6 R_WM_Sys_DsicomUpdate

Function Prototype

```
uint32_t R_WM_Sys_DiscomUpdate(const uint32_t              Unit)
```

Input Parameter

Table 3-74 Input Parameter of R_WM_Sys_DiscomUpdate

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function executes jobs for DISCOM in message queue.
This function is called from R_WM_FrameWait and R_WM_FrameExecuteDiscom.

Default Behavior

This function executes the following processing depending on device.
RH850/D1Lx
- Returns R_WM_SYS_OK.

RH850/D1Mx
- Searches Discom device from root of linked list. If created Discom device is found, performs the following process.
    - Gets update flag from the frame status flags. The flag is set by R_WM_Sys_DiscomCrcSet.
    - If update flag is on, sets the expected CRC by executing R_DISCOM_CrcSet

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_WM_SYS_OK - Successful
R_WM_SYS_NG - Failure

See also

None

3.3 WM OS interface functions

3.3.1 R_WM_Sys_LockWindows

Function Prototype

```
void R_WM_Sys_LockWindows(const uint32_t    Unit)
```

Input Parameter

Table 3-75 Input Parameter of R_WM_Sys_LockWindows

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the windows of WM driver access to the specified unit from other threads.
This function is called from several WM APIs.

Default Behavior

This function is empty as default.

Customizing Points

If an OS is used and the WM API is called from multiple threads, implement the lock process by mutex or semaphore.

Return Codes

None

See also

None

3.3.2 R_WM_Sys_UnlockWindows

Function Prototype

```
void R_WM_Sys_UnlockWindows(const uint32_t    Unit)
```

Input Parameter

Table 3-76 Input Parameter of R_WM_Sys_UnlockWindows

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the window of WM driver access to the specified unit from other threads.
This function is called from several WM APIs.

Default Behavior

This function is empty as default.

Customizing Points

If an OS is used and the WM API is called from multiple threads, implement the unlock process by mutex or semaphore.

Return Codes

None

See also

None

3.3.3 R_WM_Sys_LockMsgQueue

Function Prototype

```
void R_WM_Sys_LockMsgQueue(const uint32_t    Unit)
```

Input Parameter

Table 3-77 Input Parameter of R_WM_Sys_LockMsgQueue

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the message queue of the specified Unit in order to prevent concurrent access from multiple threads.
This function is called from several WM APIs.

Default Behavior

This function is empty as default.

Customizing Points

If an OS is used and the WM API is called from multiple threads, implement the lock process by mutex or semaphore.

Return Codes

None

See also

None

3.3.4 R_WM_Sys_UnlockMsgQueue

Function Prototype

```
void R_WM_Sys_UnlockMsgQueue(const uint32_t    Unit)
```

Input Parameter

Table 3-78 Input Parameter of R_WM_Sys_UnlockMsgQueue

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the message queue of WM driver access to the specified unit from other threads.
This function is called from several WM APIs.

Default Behavior

This function is empty as default.

Customizing Points

If an OS is used and the WM API is called from multiple threads, implement the unlock process by mutex or semaphore according to R_WM_Sys_LockMsgQueue.

Return Codes

None

See also

None

3.3.5 R_WM_Sys_LockBuffers

Function Prototype

```
void R_WM_Sys_LockBuffers(const uint32_t    Unit)
```

Input Parameter

Table 3-79 Input Parameter of R_WM_Sys_LockBuffers

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the buffers of WM driver for access to the specified unit from other threads.
This function is called from several WM APIs.

Default Behavior

This function is empty as default.

Customizing Points

If an OS is used and the WM API is called from multiple threads, implement the lock process by mutex or semaphore.

Return Codes

None

See also

None

3.3.6 R_WM_Sys_UnlockBuffers

Function Prototype

```
void R_WM_Sys_UnlockBuffers(const uint32_t    Unit)
```

Input Parameter

Table 3-80 Input Parameter of R_WM_Sys_UnlockBuffers

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the buffers of WM driver for access to the specified unit from other threads.
This function is called from several WM APIs.

Default Behavior

This function is empty as default.

Customizing Points

If an OS is used and the WM API is called from multiple threads, implement the lock process by mutex or semaphore according to R_WM_Sys_LockBuffers.

Return Codes

None

See also

None

3.3.7 R_WM_Sys_LockDevice

Function Prototype

```
void R_WM_Sys_LockDevice(const uint32_t    Unit)
```

Input Parameter

Table 3-81 Input Parameter of R_WM_Sys_LockDevice

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the all devices.
There is only one resource to lock, not each WM Unit.
This function is called from R_WM_DevInit.
This function is also called from following porting layer functions.

- R_WM_Sys_DevFrameFinished
- R_WM_Sys_SpriteEnable
- R_WM_Sys_SpriteMove
- R_WM_Sys_SpriteBufSet
- R_WM_Sys_WindowDeleteAllSprites
- R_WM_Sys_WindowCreate
- R_WM_Sys_WindowSetFb
- R_WM_Sys_WindowPosSet
- R_WM_Sys_CaptureCreate
- R_WM_Sys_CaptureEnable

Default Behavior

This function is empty as default.

Customizing Points

Implement the lock process by mutex or semaphore if WM API is called from multi-thread.

Return Codes

None

See also

None

3.3.8 R_WM_Sys_UnlockDevice

Function Prototype

```
void R_WM_Sys_UnlockDevice(const uint32_t    Unit)
```

Input Parameter

Table 3-82 Input Parameter of R_WM_Sys_UnlockDevice

Parameter	Description
Unit	Specifies the WM Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the all devices.
There is only one resource to lock, not each WM Unit.
This function is called from R_WM_DevInit.
This function is also called from following porting layer functions.

- R_WM_Sys_DevFrameFinished
- R_WM_Sys_SpriteEnable
- R_WM_Sys_SpriteMove
- R_WM_Sys_SpriteBufSet
- R_WM_Sys_WindowDeleteAllSprites
- R_WM_Sys_WindowCreate
- R_WM_Sys_WindowSetFb
- R_WM_Sys_WindowPosSet
- R_WM_Sys_CaptureCreate
- R_WM_Sys_CaptureEnable

Default Behavior

This function is empty as default.

Customizing Points

If an OS is used and the WM API is called from multiple threads, implement the unlock process by mutex or semaphore according to R_WM_Sys_LockDevice.

Return Codes

None

See also

None

3.4 WM Other interface functions

3.4.1 R_WM_Sys_GetLastError

Function Prototype

```
r_wm_Error_t R_WM_Sys_GetLastError(uint32_t        *const Unit,
                                    uint32_t        *const AdditionalInfo,
                                    uint32_t        *const Overflow,
                                    uint32_t        *const New)
```

Input Parameter

None

Output Parameter

Table 3-83 Input Parameter of R_WM_Sys_GetLastError

Parameter	Description
Unit	Reference parameter to get the WM Unit that caused the error.
AdditionalInfo	Reference parameter to get additional error codes from VDCE or SPEA.
Overflow	Reference parameter to get information if errors happened after this one occurred.
New	Reference parameter to optionally check if this is a new error.

Input – Output Parameter

None

Description

Due to technical reasons, the WM porting layer only returns success/failure without specific error codes.
This function can request information about the last error that happened in the porting layer of the WM.
Calling this function will reset the error flags and allows for a new error to be recorded.

Default Behavior

This function is not compiled as default.

Customizing Points

To use this function, the WM porting layer must be recompiled with the flag R_WM_SYS_ERROR_TRACKING.
For further debugging, the function may be expanded to provide ‘File’ and ‘Line’ of the error.

Return Codes

R_WM_ERR_OK	- No error has occurred.
Other return code of r_wm_Error_t	- The latest error code.

See also

None

3.5 Data types

3.5.1 r_wm_sys_DevFeature_t

Description

Device specific features.

Definition

```
typedef enum
{
    R_WM_SYS_FEATURE_RLE_LAYER_NO,
    R_WM_SYS_FEATURE_SPRITE_LAYER_NO,
    R_WM_SYS_FEATURE_SWITCH_CAPABILITIES,
    R_WM_SYS_FEATURE_GAMMA_CORRECTION,
    R_WM_SYS_FEATURE_SCALING,
    R_WM_SYS_FEATURE_LAST,
} r_wm_sys_DevFeature_t
```

Table 3-84 Enumerator of r_wm_sys_DevFeature_t

Name	Description
R_WM_SYS_FEATURE_RLE_LAYER_NO	The number of layers that support RLE decoding
R_WM_SYS_FEATURE_SPRITE_LAYER_NO	The number of layers that support Sprite handling
R_WM_SYS_FEATURE_SWITCH_CAPABILITIES	Availability of RLE/Sprite configuration.
R_WM_SYS_FEATURE_GAMMA_CORRECTION	Availability of Gamma correction feature.
R_WM_SYS_FEATURE_SCALING	Availability of scaling-up/ scaling-down feature.
R_WM_SYS_FEATURE_LAST	Delimiter, must be last element

See also

R_WM_Sys_DeviceFeature

3.5.2 r_wm_Memory_t

Description

The window manager requires different access types to the memory. Therefore the function R_WM_SysAlloc is called with a parameter, which indicates the memory type requested by the window manager.

Definition

```
typedef enum
{
    R_WM_MEM_CPU = 0,
    R_WM_MEM_VIDEO,
} r_wm_Memory_t
```

Table 3-85 Enumerator of r_wm_Memory_t

Name	Description
R_WM_MEM_CPU	The memory needs to be accessible by the CPU.
R_WM_MEM_VIDEO	The memory needs to be accessible by the VDCE, GPU and CPU.

See also

R_WM_Sys_Alloc
R_WM_Sys_Free

3.5.3 r_wm_State_t

Description

The window manager can be in different states which may only support a subset of all available API functions. This type helps to keep track of the current state of the WM.

Definition

```
typedef enum
{
    R_WM_STATE_UNINITIALIZED = 0,
    R_WM_STATE_INITIALIZED,
    R_WM_STATE_DISPLAY_INITIALIZED,
    R_WM_STATE_DISPLAY_ACTIVE
} r_wm_State_t;
```

Table 3-86 Enumerator of r_wm_State_t

Name	Description
R_WM_STATE_UNINITIALISED	The driver is not yet initialized
R_WM_STATE_INITIALISED	The window is initialized by R_WM_DevInit
R_WM_STATE_DISPLAY_INITIALIZED	The display is initialized and can be turned on
R_WM_STATE_DISPLAY_ACTIVE	The display is activated

See also

R_WM_Sys_StateSet
R_WM_Sys_StateGet

3.5.4 r_wm_ScaleChg_t

Description

The type is used to specify scaled size change behavior.

Definition

```
typedef enum
{
    R_WM_SCALE_CHANGE_NONE = 0,
    R_WM_SCALE_CHANGE_SMALL,
    R_WM_SCALE_CHANGE_LARGE,
    R_WM_SCALE_CHANGE_WIN_SIZE
} r_wm_ScaleChg_t;
```

Table 3-87 Enumerator of r_wm_ScaleChg_t

Name	Description
R_WM_SCALE_CHANGE_NONE	No change. Set from instance value. (Internal use)
R_WM_SCALE_CHANGE_SMALL	Change scale-down size.
R_WM_SCALE_CHANGE_LARGE	Change scale-up size.
R_WM_SCALE_CHANGE_WIN_SIZE	Change window size. (Internal use)

See also

R_WM_Sys_WindowScaleSet

3.6 Definition

3.6.1 Return code

Description

This section shows the return code of WM porting layer functions.

Table 3-88 Definition of return code

Name	Description
R_WM_SYS_OK	Function succeeded.
R_WM_SYS_NG	Function failed.

3.6.2 Sample OS

Description

This section shows the sample OS option.
If USE_ROS is defined, WM porting layer controls the sample OS.
It changes the behavior of fallowing functions and R_WM_FrameWait waits for the interrupt with semaphore features instead of a permanent loop.

- R_WM_Sys_DevWaitForHwWriteReady
- R_WM_Sys_DevWaitForHwUpdated

It changes the behavior of fallowing functions and exclusive control is enabled.

- R_WM_Sys_LockWindows
- R_WM_Sys_UnlockWindows
- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnlockMsgQueue
- R_WM_Sys_LockBuffers
- R_WM_Sys_UnlockBuffers
- R_WM_Sys_LockDevice
- R_WM_Sys_UnlockDevice

The sample OS is stored in the directory vlib/os/ros.
This option is referred from several files. So, set the option globally, e.g. in a makefile.

Table 3-89 Definition of return code

Name	Description
USE_ROS	If this definition is defined, WM porting layer uses the sample OS.

3.6.3 Error tracking

Description

This section shows the debug option.
If following option is enabled, the error tracking is activate and user can call R_WM_Sys_GetLastError.
This option is referred from several files. So, set the option globally, e.g. in a makefile.

Table 3-90 Definition of return code

Name	Description
R_WM_SYS_ERROR_TRACKING	If this definition is defined, error tracking is enabled.

4.Video Data Controller E (VDCE)

4.1 File list

Following table shows the file list for VDCE porting layer.

Table 4-1 File list for VDCE porting layer

File Name	Pass	Description
r_sys_wm.c	vlib/device/d1x_common/src/vdce	Source file of poring layer.
r_config_vdce.h	vlib/device/d1x_common/macro_cfg/vdce	Header file for common device configuration.
r_config_vdce_num.h	vlib/device/d1mx/macro_cfg	Header file for RH850/D1M configuration.
r_config_vdce_num.h	vlib/device/d1lx/macro_cfg	Header file for RH850/D1L configuration.
r_vdce_sys.h	vlib/macro/vdce/lib	Header file for porting layer interface.

* r_config_vdce_num.h is included one file depending on RGL package.

4.2 VDCE Driver Basic Interface functions

The VDCE driver itself shall be able to support different devices with different video output and video input macros. It also has to be possible to support different macros in the same device. Integration into different systems (OS, memory manager etc.) shall be possible. The VDCE driver must not rely on a specific environment. To achieve this, the VDCE driver can use driver support functions. These are not part of the driver itself but they must be provided to integrate the driver on a particular device, OS, board. Driver support functions shall never be called by the application directly, instead they are indirectly called when the application calls the generic VDCE driver API.

4.2.1 R_VDCE_Sys_Init

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_Init(const uint32_t Unit)
```

Input Parameter

Table 4-2 Input Parameter of R_VDCE_Sys_Init

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

The initialization code of environment-dependent (e.g. interrupt priority, power control or clock control) is to be implemented by this function. This function is called from R_VDCE_Init function.

Default Behavior

This function is empty as default. It always returns R_VDCE_ERR_OK as the return value.

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user want to add the initialization code of environment-depend (e.g. clock control), implement in this function.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.

See also

None

4.2.2 R_VDCE_Sys_DeInit

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_DeInit(const uint32_t Unit)
```

Input Parameter

Table 4-3 Input Parameter of R_VDCE_Sys_DeInit

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

The de-initialization code of environment-depend is implemented by this function. This function is called from R_VDCE_DeInit function.

Default Behavior

This function is empty as default. It always returns R_VDCE_ERR_OK as the return value.

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user want to add the de-initialization code of environment-depend (e.g. clock control), implement in this function.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.

See Also

None

4.2.3 R_VDCE_Sys_BaseAddrGet

Function Prototypes

```
uint32_t R_VDCE_Sys_BaseAddrGet(const uint32_t Unit)
```

Input Parameter

Table 4-4 Input Parameter of R_VDCE_Sys_BaseAddrGet

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function gets base address of VDCE unit.

Default Behavior

This function executes the following processing

- Returns the base address of VDCE macro.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- Base address of specified VDCE unit.
0	- Specified unit does not exist.

See Also

None

4.2.4 R_VDCE_Sys_PixelClockSet

Function Prototypes

```
uint32_t R_VDCE_Sys_PixelClockSet(const uint32_t  Unit,
                                  const uint32_t  Clock,
                                  const uint32_t  OtherUnitActive,
                                  const uint32_t  Flags
                                  )
```

Input Parameter

Table 4-5 Input Parameter of R_VDCE_Sys_PixelClockSet

Parameter	Description
Unit	Specifies the VDCE Unit number
Clock	Specifies the wanted pixel clock frequency (in MHz). The pixel clock frequency that is really set can differ from this value due to hardware restrictions.
OtherUnitActive	Maximum supported units for VDCE is two units. If the opposite unit (Unit XOR 1) is in one of the states Idle or Executing, set this flag to '1'. Set it to '0' if the other Unit is Uninitialized or Initialized. If the opposite unit does not exist, set it to '0'. Used to enable VODDR if both Units are configured and running.
Flags	Display setting Flags.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the pixel clock and video output settings. This function is called from R_VDCE_Init, R_VDCE_DeInit, R_VDCE_DisplayEnable, R_VDCE_DisplayDisable and R_VDCE_DisplayTimingAdjust.

Default Behavior

This function executes the following processing.

- Calculates the division ratio of DOTCLK0/DOTCLK1 so that it is close to the specified clock.
- Returns the actual clock of DOTCLK0/DOTCLK1.
- Executes the following processing depending on the device.
    - RH850/D1M2(H)
        - Selects DOTCLK0/DOTCLK1 as the divider of PLL2CLK.
        - Selects C_ISO_VDCE0CLK/C_ISO_VDCE1CLK as DOTCLK0/DOTCLK1 in case of LVTTL.
        - Selects C_ISO_VDCE0CLK/C_ISO_VDCE1CLK as the divider of DOTCLK0/DOTCLK1 in case of RSDS.
        - Selects C_ISO_RSDSCLK as DOTCLK0 or DOTCLK1 in case of RSDS.
        - Sets RSDSCFG register.
        - Sets CKSC_IVOEXS_CTL depending on VOSL field of VDCECTL.
    - RH850/D1M1A
        - Selects DOTCLK0/DOTCLK1 as the divider of PLL1CLK.
        - Selects C_ISO_VDCE0CLK/C_ISO_VDCE1CLK as DOTCLK0/DOTCLK1 in case of LVTTL/Serial RGB/VODDR.
        - Selects C_ISO_VDCE0CLK/C_ISO_VDCE1CLK as the divider of DOTCLK0/DOTCLK1 in case of OpenLDI.
        - Sets VODDR0SYSCNT, VODDR0CLKDIV register in case of VODDR.

        - Sets CKSC_IVOEXS_CTL depending on VOSL field of VDCECTL.
    - RH850/D1M1-V2, RH850/D1M1(H), RH850/D1L2(H)
        - Selects DOTCLK0 as the divider of PLL0CLK.
        - Selects C_ISO_VDCE0CLK as DOTCLK0.

Customizing Points

- In the case to use LVTTL output, it is not necessary to modify this function in general use-case.
- Change output options if user needs to use other outputs. See 4.5.1.
- If user want to change the clock settings, modify this function. Refer H/W UM Figure 37.10 ~ 37.14.

Return Codes

not 0	- Pixel clock really set.
0	- Error occurred.

See Also

None

4.2.5 R_VDCE_Sys_IntcInit

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_IntcInit(const uint32_t Unit)
```

Input Parameter

Table 4-6 Input Parameter of R_VDCE_Sys_IntcInit

Parameter	Description
Unit	Specifies the VDCE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function enables interrupt controller (INTC) for specified VDCE unit except for VDCE0GR3VBLANK and VDCE1GR3VBLANK.
VDCE0GR3VBLANK and VDCE1GR3VBLANK are not level interrupt, so these are enabled with individual control.
This function is called from R_VDCE_DeInit.

Default Behavior

This function executes the following processing.

- Other than VDCE0GR3VBLANK and VDCE1GR3VBLANK, Clears and Enables the VDCE interrupts of INTC.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range.

See Also

None

4.2.6 R_VDCE_Sys_IntcDeinit

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_IntcDeinit(const uint32_t Unit)
```

Input Parameter

Table 4-7 Input Parameter of R_VDCE_Sys_IntcDeinit

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function disables interrupt controller (INTC) for specified VDCE unit.
This function is called from R_VDCE_DeInit.

Default Behavior

This function executes the following processing.
- Clears and Disables the VDCE interrupts of INTC.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range.

See Also

None

4.2.7 R_VDCE_Sys_IntcEnable

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_IntcEnable(const uint32_t       Unit,
                                    const r_dev_IntSel_t IntSel)
```

Input Parameter

Table 4-8 Input Parameter of R_VDCE_Sys_IntcEnable

Parameter	Description
Unit	Specifies the VDCE Unit number
IntSel	Specifies the interrupt selection. R_DEV_INT_VDCE0ERR R_DEV_INT_VDCE0GR3VBLANK R_DEV_INT_VDCE0S0VIVSYNC R_DEV_INT_VDCE0S0LOVSYNC R_DEV_INT_VDCE0GR3VLINE R_DEV_INT_VDCE0S0VFIELD R_DEV_INT_VDCE0S1LOVSYNC R_DEV_INT_VDCE0OIRVIVSYNC R_DEV_INT_VDCE0OIRLOVSYNC R_DEV_INT_VDCE0IRVLINE

Input – Output Parameter

None

Output Parameter

None

Description

This function enables the specified interrupt of interrupt controller (INTC).
This function is called from R_VDCE_IntcEnable.

Default Behavior

This function executes the following processing.

- Clears and Enables the specified interrupts of INTC.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range.
R_VDCE_ERR_RANGE_PARAM	- A parameter is outside the range.

See Also

r_dev_IntSel_t

4.2.8 R_VDCE_Sys_IntcDisable

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_IntcDisable(const uint32_t       Unit,
                                      const r_dev_IntSel_t  IntSel)
```

Input Parameter

Table 4-9 Input Parameter of R_VDCE_Sys_IntcDisable

Parameter	Description
Unit	Specifies the VDCE Unit number
IntSel	Specifies the interrupt selection. R_DEV_INT_VDCE0ERR R_DEV_INT_VDCE0GR3VBLANK R_DEV_INT_VDCE0S0VIVSYNC R_DEV_INT_VDCE0S0LOVSYNC R_DEV_INT_VDCE0GR3VLINE R_DEV_INT_VDCE0S0VFIELD R_DEV_INT_VDCE0S1LOVSYNC R_DEV_INT_VDCE0OIRVIVSYNC R_DEV_INT_VDCE0OIRLOVSYNC R_DEV_INT_VDCE0IRVLINE

Input – Output Parameter

None

Output Parameter

None

Description

This function disables the specified interrupt of interrupt controller (INTC).
This function is called from R_VDCE_IntcDisable.

Default Behavior

This function executes the following processing.

- Clears and Disables the specified interrupts of INTC.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range
R_VDCE_ERR_RANGE_PARAM	- A parameter is outside the range

See Also

r_dev_IntSel_t

4.2.9 R_VDCE_Sys_IntcPrepareClearCheck

Function Prototypes

```
uint32_t R_VDCE_Sys_IntcPrepareClearCheck(const uint32_t SyncFreq)
```

Input Parameter

Table 4-10 Input Parameter of R_VDCE_Sys_IntcPrepareClearCheck

Parameter	Description
SyncFreq	Specifies the frequency of synchronization clock. If 0 is specified, the default value is used.

Input – Output Parameter

None

Output Parameter

None

Description

This function prepares the interrupt clear checking.
This function is called from R_VDCE_IntcEnable.

Default Behavior

This function executes the following processing.

- Calculates the waiting count corresponding to four cycle of specified synchronization clock.
- Waits for INT_STAx to change 0 to1 by NOP instruction.
- Returns the time-out count of register polling (INT_STAx to change 1 to 0).

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user use external video input (capture), set minimum input clock to LOC_SYNC_FREQ_MIN. LOC_SYNC_FREQ_MIN also can be small enough like default setting.
- User can change LOC_CPU_FREQ_MAX definition depending on the target device.

Return Codes

Time-out count of register polling.

See Also

None

4.2.10 R_VDCE_Sys_IntcClearCheck

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_IntcClearCheck(const uint32_t         Unit,
                                         const r_vdce_IntType_t  IntType)
```

Input Parameter

Table 4-11 Input Parameter of R_VDCE_Sys_IntcClearCheck

Parameter	Description
Unit	Specifies the VDCE Unit number
IntType	Specifies the interrupt type.

Input – Output Parameter

None

Output Parameter

None

Description

This function waits until the interrupt is cleared.
This function is called from R_VDCE_Isr.

Default Behavior

This function executes the following processing.

- Waits for EICn.EIRFn to change 1->0.

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user want to change the time-out count, modify the definition of LOC_WAIT_CLEAR_COUNT.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_PARAM	- A parameter is outside the range.
R_VDCE_ERR_FATAL_HW	- EICn.EIRFn is not changed to 0.

See Also

None

4.2.11 R_VDCE_Sys_PortInit

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_PortInit(const uint32_t Unit)
```

Input Parameter

Table 4-12 Input Parameter of R_VDCE_Sys_PortInit

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function initializes port for the given VDCE unit.
This function is called from R_VDCE_Init function.

Default Behavior

This function is empty as default. It always returns R_VDCE_ERR_OK as the return value.

Customizing Points

- It is not necessary to modify this function in general use-case.
- User can add the port setting in this function.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.

See Also

None

4.2.12 R_VDCE_Sys_PortDeInit

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_PortDeInit(const uint32_t Unit)
```

Input Parameter

Table 4-13 Input Parameter of R_VDCE_Sys_PortDeInit

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function sets back port settings. This function is called from R_VDCE_DeInit.

Default Behavior

This function is empty as default. It always returns R_VDCE_ERR_OK as the return value.

Customizing Points

- It is not necessary to modify this function in general use-case.
- User can add the port setting in this function.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.

See Also

None

4.2.13 R_VDCE_Sys_HsyncActLevelSet

Function Prototypes

```
uint32_t R_VDCE_Sys_HsyncActLevelSet(const uint32_t Unit,
                                     const int32_t  Level)
```

Input Parameter

Table 4-14 Input Parameter of R_VDCE_Sys_HsyncActLevelSet

Parameter	Description
Unit	Specifies the VDCE Unit number
Level	Specifies the Hsync signal level. 0: Active Low 1: Active High

Input – Output Parameter

None

Output Parameter

None

Description

This function sets output Hsync signal level.
If this function returns R_FALSE , this function does not handle the level, and VDCE driver will handle the Hsync level.
If this function returns R_TRUE, this function handles the level, and VDCE driver will not handle the Hsync level.
VDCE outputs Hsync with active high.

This function is called from R_VDCE_DeInit, R_VDCE_DisplayEnable and R_VDCE_DisplayTimingAdjust.

Default Behavior

This function is empty and returns R_FALSE as default.

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user want to handle Hsync level outside VDCE (e.g. Port), implement to this function and returns 1.

Return Codes

R_FALSE	- This function doesn't handle Hsync level.
R_TRUE	- This function handles Hsync level.

See Also

None

4.2.14 R_VDCE_Sys_VsyncActLevelSet

Function Prototypes

```
uint32_t R_VDCE_Sys_VsyncActLevelSet(const uint32_t Unit,
                                     const int32_t  Level)
```

Input Parameter

Table 4-15 Input Parameter of R_VDCE_Sys_VsyncActLevelSet

Parameter	Description
Unit	Specifies the VDCE Unit number
Level	Specifies the Vsync signal level. 0: Active Low 1: Active High

Input – Output Parameter

None

Output Parameter

None

Description

This function sets output Vsync signal level.
If this function returns R_FALSE, this function does not handle the level, and VDCE driver will handle the Vsync level.
If this function returns R_TRUE, this function handles the level, and VDCE driver will not handle the Vsync level. VDCE outputs Vsync with active high.

This function is called from R_VDCE_DeInit, R_VDCE_DisplayEnable and R_VDCE_DisplayTimingAdjust.

Default Behavior

This function is empty and returns R_FALSE as default.

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user want to handle Vsync level outside VDCE (e.g. Port), implement to this function and returns 1.

Return Codes

R_FALSE	- This function doesn't handle Vsync level.
R_TRUE	- This function handles Vsync level.

See Also

None

4.2.15 R_VDCE_Sys_ClockActEdgeSet

Function Prototypes

```
uint32_t R_VDCE_Sys_ClockActEdgeSet(const uint32_t Unit,
                                    const uint32_t Flags)
```

Input Parameter

Table 4-16 Input Parameter of R_VDCE_Sys_ClockActEdgeSet

Parameter	Description
Unit	Specifies the VDCE Unit number
Flags	Specifies the display setting Flags. If R_DDB_DISP_FLAG_NEGCLK flag is off, data change with rising edge of clock. If R_DDB_DISP_FLAG_NEGCLK flag is on, data change with falling edge of clock.

Input – Output Parameter

None

Output Parameter

None

Description

This function sets VO_DATA[23:0] change timing with clock edge.
If this function returns R_FALSE, this function does not handle the clock edge, and VDCE driver will handle the clock edge.
If this function returns 1, this function handles the level, and VDCE driver will not handle the clock edge. VDCE outputs VO_DATA[23:0] with rising edge of clock.

This function is called from R_VDCE_DeInit, R_VDCE_DisplayEnable and R_VDCE_DisplayTimingAdjust.

Default Behavior

This function executes the following processing.

- Sets PINV45_0 and returns R_TRUE in case of VDCE0_VO LVTTL mode.
- Sets PINV47_8 and returns R_TRUE in case of VDCE1_VO LVTTL mode.
- Does nothing and returns R_TRUE in case of RSDS mode.
- Does nothing and returns R_FALSE in case of RH850/D1M1A device and output form VDCE1_VO port.
- If unit number is invalid, returns R_FALSE.

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user want to change the handling, modify this function.

Return Codes

R_FALSE	- This function doesn't handle data change timing with the clock edge.
R_TRUE	- This function handles data change timing with the clock edge.

See Also

None

4.2.16 R_VDCE_Sys_DesyncActLevelSet

Function Prototypes

```
uint32_t R_VDCE_Sys_DesyncActLevelSet(const uint32_t Unit,
                                      const int32_t  Level)
```

Input Parameter

Table 4-17 Input Parameter of R_VDCE_Sys_DesyncActLevelSet

Parameter	Description
Unit	Specifies the VDCE Unit number
Level	Specifies the DE sync level to be set. 0: Active Low 1: Active High

Input – Output Parameter

None

Output Parameter

None

Description

This function sets output DE (Data Enable) signal level.
If this function returns R_FALSE, this function does not handle the level, and VDCE driver will handle the DE signal level.
If this function returns R_TRUE, this function handles the level, and VDCE driver will not handle the DE signal level.
VDCE outputs DE signal with active high.

This function is called from R_VDCE_DeInit, R_VDCE_DisplayEnable and R_VDCE_DisplayTimingAdjust.

Default Behavior

This function is empty and returns R_FALSE as default.

Customizing Points

- It is not necessary to modify this function in general use-case.
- If user want to handle DE signal level outside VDCE (e.g. Port), implement to this function and returns 1.

Return Codes

R_FALSE	- This function doesn't handle DE signal level.
R_TRUE	- This function handles DE signal level.

See Also

None

4.2.17 R_VDCE_Sys_StateSet

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_StateSet(const uint32_t       Unit,
                                   const uint32_t       LayerNr,
                                   const r_vdce_State_t State)
```

Input Parameter

Table 4-18 Input Parameter of R_VDCE_Sys_StateSet

Parameter	Description
Unit	Specifies the VDCE Unit number.
LayerNr	Specifies the layer number to use. R_VDCE_LAYER_SCALER0 R_VDCE_LAYER_SCALER1 R_VDCE_LAYER_IMAGE2 R_VDCE_LAYER_IMAGE3 R_VDCE_LAYER_INPUT R_VDCE_SYS_LAYER_OIR R_VDCE_SYS_LAYER_CAP R_VDCE_SYS_LAYER_ALL
State	Specifies the requested state of VDCE driver. R_VDCE_STATE_UNINITIALIZED R_VDCE_STATE_INITIALIZED R_VDCE_STATE_IDLE R_VDCE_STATE_EXECUTING

Input – Output Parameter

None

Output Parameter

None

Description

This function sets the VDCE driver status.
VDCE driver requires managing 6 layer (Scaler0/Scaler1/Image synthesizer2/ Image synthesizer3/OIR/Capture) status per unit.
If R_VDCE_LAYER_INPUT or R_VDCE_SYS_LAYER_CAP is specified, this function changes the capture status.
If R_VDCE_SYS_LAYER_ALL is specified, this function changes the status of all 6 layers.
This function is called from several APIs.

Default Behavior

This function executes the following processing.

- If individual layer is specified, this function changes the status of specified layer.
- If R_VDCE_SYS_LAYER_ALL is specified, this function changes the status of all 6 layers.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range
R_VDCE_ERR_RANGE_LAYER	- The layer-number is outside the range

See Also

r_vdce_State_t

4.2.18 R_VDCE_Sys_StateGet

Function Prototypes

```
r_vdce_State_t R_VDCE_Sys_StateGet(const uint32_t Unit,
                                   const uint32_t LayerNr)
```

Input Parameter

Table 4-19 Input Parameter of R_VDCE_Sys_StateGet

Parameter	Description
Unit	Specifies the VDCE Unit number.
LayerNr	Specifies the layer number to use. R_VDCE_LAYER_SCALER0 R_VDCE_LAYER_SCALER1 R_VDCE_LAYER_IMAGE2 R_VDCE_LAYER_IMAGE3 R_VDCE_LAYER_INPUT R_VDCE_SYS_LAYER_OIR R_VDCE_SYS_LAYER_CAP R_VDCE_SYS_LAYER_ALL

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to get the status of VDCE driver.
VDCE driver requires managing 6 layer (Scaler0/Scaler1/Image synthesizer2/ Image synthesizer3/OIR/Capture) status per unit.
If R_VDCE_LAYER_INPUT or R_VDCE_SYS_LAYER_CAP is specified, this function returns the capture status.
If R_VDCE_SYS_LAYER_ALL is specified, this function returns the most advanced status of the 6 layers.
This function is called from several APIs.

Default Behavior

This function executes the following processing.

- If individual layer is specified, this function returns the status of specified layer.
- If R_VDCE_SYS_LAYER_ALL is specified, this function returns the most advanced status of the 6 layers.
- If invalid unit or layer number is specified, this function returns R_VDCE_STATE_UNINITIALIZED.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_STATE_UNINITIALIZED	- Specified layer is in an Uninitialized state.
R_VDCE_STATE_INITIALIZED	- Specified layer is in an Initialized state.
R_VDCE_STATE_IDLE	- Specified layer is in an Idle state.
R_VDCE_STATE_EXECUTING	- Specified layer is in an Executing state.

See Also

r_vdce_State_t

4.2.19 R_VDCE_Sys_VIChannelCheck

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_VIChannelCheck(const uint32_t Unit)
```

Input Parameter

Table 4-20 Input Parameter of R_VDCE_Sys_VIChannelCheck

Parameter	Description
Unit	Specifies the VDCE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function returns the availability of video input feature.
This function is called from VDCE capture APIs.

Default Behavior

This function executes the following processing.

- Returns R_VDCE_ERR_NOT_ACCEPTABLE if target device does not support the Video input.
- Returns R_VDCE_ERR_RANGE_UNIT if specified unit does not support the Video input.
- Returns R_VDCE_ERR_OK if target device of specified unit supports the Video input.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range.
R_VDCE_ERR_NOT_ACCEPTABLE	- Not supported device

See Also

None

4.2.20 R_VDCE_Sys_MaxResolutionGet

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_MaxResolutionGet(uint32_t *const    ResolutionHmax,
                                            uint32_t *const    ResolutionVmax)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

Table 4-21 Output Parameter of R_VDCE_Sys_MaxResolutionGet

Parameter	Description
ResolutionHmax	Maximum horizontal resolution (pixel).
ResolutionVmax	Maximum vertical resolution (pixel).

Description

This function returns the resolution information. This function is called from R_VDCE_DisplayTimingSet.

Default Behavior

This function executes the following processing.

- Stores the maximum video output resolution of VDCE macro depending on the target device.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_VDCE_ERR_NOT_SUPPORTED	- Target device is unknown.

See Also

None

4.2.21 R_VDCE_Sys_DeviceInfoGet

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_DeviceInfoGet(r_vdce_DeviceInfo_t  *DevInfo)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

Table 4-22 Output Parameter of R_VDCE_Sys_DeviceInfoGet

Parameter	Description
DevInfo	Device information.

Description

This function returns the device information. This function is called from several VDCE APIs.

Default Behavior

This function executes the following processing.

- Stores the VDCE macro information to DevInfo depending on the target device.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_PARAM_INCORRECT	- A parameter provided to a function is incorrect.
R_VDCE_ERR_NOT_SUPPORTED	- Target device is unknown.

See Also

r_vdce_DeviceInfo_t

4.2.22 R_VDCE_Sys_InitGlobal

Function Prototypes

```
void R_VDCE_Sys_InitGlobal(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

Initializes the global variables in VDCE porting layer.

If R_BSP_SYS_INIT_USE is defined, user must call this function before calling VDCE APIs.
This function is called from R_DEV_SysInit provided as sample code.

If R_BSP_SYS_INIT_USE is not defined, global variables are declared with initial values.
This function call is not mandatory.

Default Behavior

This function executes the following processing.
- Initialize global variables.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

4.3 OS interface

The VDCE driver shall support access from multiple threads in a multi-threading environment. At least in case of atomic data manipulation (e.g. adding events to a queue), it has to be possible to avoid concurrent access to the same data structure. It might be necessary to add further OS interface functions here.

4.3.1 R_VDCE_Sys_Lock

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_Lock(const uint32_t   Unit)
```

Input Parameter

Table 4-23 Input Parameter of R_VDCE_Sys_Lock

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the VDCE driver access to the specified unit for other threads.

Default Behavior

This function is empty as default. It always returns R_VDCE_ERR_OK as the return value.

Customizing Points

User need to implement the lock process by mutex or semaphore if VDCE API is called from multi-thread.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range

See Also

None

4.3.2 R_VDCE_Sys_Unlock

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_Unlock(const uint32_t Unit)
```

Input Parameter

Table 4-24 Input Parameter of R_VDCE_Sys_Unlock

Parameter	Description
Unit	Specifies the VDCE Unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the VDCE driver access to the specified unit for other threads.

Default Behavior

This function is empty as default. It always returns R_VDCE_ERR_OK as the return value.

Customizing Points

User need to implement the unlock process depending on R_VDCE_Sys_Lock.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.
R_VDCE_ERR_RANGE_UNIT	- The unit-number is outside the range

See Also

None

4.3.3 R_VDCE_Sys_AllLock

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_AllLock(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the VDCE driver access to the all unit for other threads.
This function is called from following functions.

- R_VDCE_ErrorCallbackSet
- R_VDCE_CapEnable
- R_VDCE_CapDisable

Default Behavior

This function executes the following processing.

- Call R_VDCE_Sys_Lock to all available unit.

Customizing Points

User need to implement the lock process by mutex or semaphore if VDCE API is called from multi-thread.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.

See Also

None

4.3.4 R_VDCE_Sys_AllUnlock

Function Prototypes

```
r_vdce_Error_t R_VDCE_Sys_AllUnlock(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the VDCE driver access to the all unit for other threads.
This function is called from following functions.

- R_VDCE_ErrorCallbackSet
- R_VDCE_CapEnable
- R_VDCE_CapDisable

Default Behavior

This function executes the following processing.

- Call R_VDCE_Sys_Unlock to all available unit.

Customizing Points

User need to implement the unlock process depending on R_VDCE_Sys_AllLock.

Return Codes

R_VDCE_ERR_OK	- No error has occurred.

See Also

None

4.4 Data types

The chapter describes all data types and defines used by driver support functions.

4.4.1 r_vdce_DeviceInfo_t

Description

The structure holding the value of device information in the function R_VDCE_Sys_DeviceInfoGet.

Definition

```
typedef struct
{
 r_dev_Device_t  Device;
 uint8_t         VIWithDataEn;
 uint8_t         SerialRGBEn;
 uint8_t         GammaCorrectEn;
 uint8_t         EnlargementEn;
} r_vdce_DeviceInfo_t
```

Table 4-25 Members of r_vdce_DeviceInfo_t structure

Name	Description
Device	Specifies the device version number that VDCE driver assumes
VIWithDataEn	Specifies the availability of video input with data enable signal. (R_TRUE / R_FALSE)
SerialRGBEn	Specifies the availability of serial RGB output. (R_TRUE / R_FALSE)
GammaCorrectEn	Specifies the availability of gamma correction. (R_TRUE / R_FALSE)
EnlargementEn	Specifies the availability of enlargement. (R_TRUE / R_FALSE)

See also

R_VDCE_Sys_DeviceInfoGet

4.4.2 r_dev_IntSel_t

Description

All possible interrupt control registers.
Only the definition related to VDCE is described here

Definition

```
typedef enum
{
    R_DEV_INT_VDCE0ERR              ,
    R_DEV_INT_VDCE0GR3VBLANK        ,
    R_DEV_INT_VDCE0S0VIVSYNC        ,
    R_DEV_INT_VDCE0S0LOVSYNC        ,
    R_DEV_INT_VDCE0GR3VLINE         ,
    R_DEV_INT_VDCE0S0VFIELD         ,
    R_DEV_INT_VDCE0S1LOVSYNC        ,
    R_DEV_INT_VDCE0OIRVIVSYNC       ,
    R_DEV_INT_VDCE0OIRLOVSYNC       ,
    R_DEV_INT_VDCE0IRVLINE          ,
    R_DEV_INT_VDCE1ERR              ,
    R_DEV_INT_VDCE1GR3VBLANK        ,
    R_DEV_INT_VDCE1S0VIVSYNC        ,
    R_DEV_INT_VDCE1S0LOVSYNC        ,
    R_DEV_INT_VDCE1GR3VLINE         ,
    R_DEV_INT_VDCE1S0VFIELD         ,
    R_DEV_INT_VDCE1S1LOVSYNC        ,
 } r_dev_IntSel_t
```

Table 4-26 Enumerator of r_dev_IntSel_t

Name	Description
R_DEV_INT_VDCE0ERR	Control register for interrupt INTVDCE0ERR
R_DEV_INT_VDCE0GR3VBLANK	Control register for interrupt INTVDCE0GR3VBLANK
R_DEV_INT_VDCE0S0VIVSYNC	Control register for interrupt INTVDCE0S0VIVSYNC
R_DEV_INT_VDCE0S0LOVSYNC	Control register for interrupt INTVDCE0S0LOVSYNC
R_DEV_INT_VDCE0GR3VLINE	Control register for interrupt INTVDCE0GR3VLINE
R_DEV_INT_VDCE0S0VFIELD	Control register for interrupt INTVDCE0S0VFIELD
R_DEV_INT_VDCE0S1LOVSYNC	Control register for interrupt INTVDCE0S1LOVSYNC
R_DEV_INT_VDCE0OIRVIVSYNC	Control register for interrupt INTVDCE0OIRVIVSYNC
R_DEV_INT_VDCE0OIRLOVSYNC	Control register for interrupt INTVDCE0OIRLOVSYNC
R_DEV_INT_VDCE0IRVLINE	Control register for interrupt INTVDCE0OIRVLINE
R_DEV_INT_VDCE1ERR	Control register for interrupt INTVDCE1ERR
R_DEV_INT_VDCE1GR3VBLANK	Control register for interrupt INTVDCE1GR3VBLANK
R_DEV_INT_VDCE1S0VIVSYNC	Control register for interrupt INTVDCE1S0VIVSYNC
R_DEV_INT_VDCE1S0LOVSYNC	Control register for interrupt INTVDCE1S0LOVSYNC
R_DEV_INT_VDCE1GR3VLINE	Control register for interrupt INTVDCE1GR3VLINE
R_DEV_INT_VDCE1S0VFIELD	Control register for interrupt INTVDCE1S0VFIELD
R_DEV_INT_VDCE1S1LOVSYNC	Control register for interrupt INTVDCE1S1LOVSYNC

See Also

R_VDCE_Sys_IntcEnable
R_VDCE_Sys_IntcDisable

4.4.3 r_vdce_State_t

Description

This type describes the state of VDCE.

Definition

```
typedef enum
{
    R_VDCE_STATE_UNINITIALIZED = 0,
    R_VDCE_STATE_INITIALIZED,
    R_VDCE_STATE_IDLE,
    R_VDCE_STATE_EXECUTING
} r_vdce_State_t
```

Table 4-27 Enumerator of r_vdce_State_t

Name	Description
R_VDCE_STATE_UNINITIALIZED	The VDCE driver is an uninitialized state.
R_VDCE_STATE_INITIALIZED	The VDCE driver is an initialization state.
R_VDCE_STATE_IDLE	The VDCE driver is an idle state.
R_VDCE_STATE_EXECUTING	The VDCE driver is an executing state.

See Also

R_VDCE_Sys_StateSet
R_VDCE_Sys_StateGet

4.5 Definition

4.5.1 Video output option

Description

This section shows the video output option.
The following options are also used in sample bsp. So if sample bsp is used, set these options globally, e.g. in a makefile.
VDCE porting layer configures the output option partly. Refer to sample bsp for remaining settings.

Table 4-28 Definition of video output option

Name	Description	Support Device
USE_VDCE_SERIALRGB	Serial RGB output configuration 0: Serial RGB is not used. 1: Serial RGB is used.	RH850/D1M1A RH850/D1M1-V2
USE_VDCE_OPENLDI	Open LDI output configuration 0: Open LDI is not used. 1: Open LDI is used.	RH850/D1M1A
USE_VDCE_VODDR	VODDR output configuration VODDR is exclusive with Serial RGB or Open LDI. 0: VODDR is not used. 1: VODDR is used.	RH850/D1M1A
USE_VDCE_SERIALRGB_SPEED	Serial RGB speed. This is valid when USE_VDCE_SERIALRGB is defined as 1. 3: Triple Speed. 4: Quadruple Speed	RH850/D1M1A RH850/D1M1-V2

4.5.2 Layer number

Description

This section shows the layer number.
These values are used with the argument of R_VDCE_Sys_StateSet and R_VDCE_Sys_StateGet.

Name	Description
R_VDCE_SYS_MAX_LAYER_NUM	The number of the layer to manage status.
R_VDCE_SYS_LAYER_OIR	OIR layer.
R_VDCE_SYS_LAYER_CAPT	Capture layer.
R_VDCE_SYS_LAYER_ALL	All layer

See Also

R_VDCE_Sys_StateSet
R_VDCE_Sys_StateGet

5.Sprite Engine (SPEA)

5.1 File list

Following table shows the file list for SPEA porting layer .

Table 5-1 File list for SPEA porting layer

File Name	Pass	Description
r_sys_spea.c	vlib/device/d1x_common/src/spea	Source file of poring layer.
r_config_spea.h	vlib/device/d1x_common/macro_cfg/spea	Header file for device configuration.
r_vdce_sys.h	vlib/macro/vo/spea/lib	Header file for porting layer interface.

5.2 Driver support functions

The SPEA driver must not rely on a specific environment. to achieve this , the SPEA driver can use driver support functions. These are not part of the driver itself but they must be provided to integrate the driver on a particular device, OS, board. Driver support functions shall never be called by the application directly, instead they are indirectly called when the application calls the generic SPEA driver API. They have to be implemented within the driver library for a concrete device. (e.g. D1L, D1M).

5.2.1 R_SPEA_SYS_HardwareInit

Function Prototypes

```
r_spea_Error_t R_SPEA_SYS_HardwareInit(const uint32_t Unit)
```

Input Parameter

Table 5-2 Input Parameter of R_SPEA_SYS_HardwareInit

Parameter	Description
Unit	Specifies the SPEA Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

Initialize the environment-dependent H/W for the SPEA.
This function is called from R_SPEA_Init.

Default Behavior

This function executes the following processing.

- Initialize the sprite engine update timing control register (SPEAUPDEN) by value 0x0F0Fu.

Customizing Points

User can modify SPEAUPDEN setting.

Return Codes

R_SPEA_ERR_OK	- No error has occurred.
R_SPEA_ERR_NG	- Error occurred.

See Also

None

5.2.2 R_SPEA_SYS_HardwareDeInit

Function Prototypes

```
r_spea_Error_t R_SPEA_SYS_HardwareDeInit(const uint32_t Unit)
```

Input Parameter

Table 5-3 Input Parameter of R_SPEA_SYS_HardwareDeInit

Parameter	Description
Unit	Specifies the SPEA Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function de-initializes environment-dependent H/W.
This function is called from R_SPEA_DeInit.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_SPEA_ERR_OK	- No error has occurred.
R_SPEA_ERR_NG	- Error occurred

See Also

None

5.2.3 R_SPEA_SYS_BaseAddr

Function Prototypes

```
uint32_t R_SPEA_SYS_BaseAddr(const uint32_t Unit)
```

Input Parameter

Table 5-4 Input Parameter of R_SPEA_SYS_BaseAddr

Parameter	Description
Unit	Specifies the SPEA Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function gives back the base address of SPEA H/W register.
This function is called from R_SPEA_Init.

Default Behavior

This function executes the following processing.

- Returns the base address of SPEA H/W register.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- SPEA H/W base address.
0	- Unit is invalid.

See Also

None

5.2.4 R_SPEA_SYS_ErrorHandler

Function Prototypes

```
void R_SPEA_SYS_ErrorHandler(const uint32_t        Unit,
                             const r_spea_Error_t  Error)
```

Input Parameter

Table 5-5 Input Parameter of R_SPEA_SYS_ErrorHandler

Parameter	Description
Unit	Specifies the SPEA Unit number or RLE unit number.
Error	Specifies the detected error.

Input – Output Parameter

None

Output Parameter

None

Description

Low level error handler, called in case there is no user error handled by R_SPEA_SetErrorCallback that assigned for this macro.
This function is called from several SPEA APIs.
The value set in the argument “Unit” varies depending on the calling API.

Default Behavior

This function is empty as default.

Customizing Points

User can modify this function freely.

Return Codes

None

See Also

None

5.2.5 R_SPEA_SYS_IsD1M1A

Function Prototypes

```
int8_t R_SPEA_SYS_IsD1M1A(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

This function returns the flag of RH850/D1M1A or otherwise.
This function is called from several SPEA APIs.

Default Behavior

This function executes the following processing.

- If target device is RH850/D1M1A this function should return R_TRUE.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_TRUE	- Target device is RH850/D1M1A.
R_FALSE	- Target device is not RH850/D1M1A.

See Also

None

5.2.6 R_SPEA_SYS_IsD1M1v2

Function Prototypes

```
int8_t R_SPEA_SYS_IsD1M1v2(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

This function returns the flag of RH850/D1M1-V2 or otherwise.
This function is called from several SPEA APIs.

Default Behavior

This function executes the following processing.

- If target device is RH850/D1M1-V2, this function should return R_TRUE.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_TRUE	- Target device is RH850/D1M1-V2.
R_FALSE	- Target device is not RH850/D1M1-V2.

See Also

None

5.2.7 R_SPEA_SYS_StateSet

Function Prototypes

```
r_spea_Error_t R_SPEA_SYS_StateSet(const uint32_t       Unit,
                                   const r_spea_State_t State
                                   const r_spea_Unit_t  SpUnit)
```

Input Parameter

Table 5-6 Input parameter of R_SPEA_SYS_StateSet

Parameter	Description
Unit	Specifies the SPEA Unit number.
State	Specifies the requested state of SPEA driver. R_SPEA_STATE_UNINITIALIZED R_SPEA_STATE_IDLE R_SPEA_STATE_UPDATING R_SPEA_STATE_EXECUTING
SpUnit	Specifies RLE unit or Sprite unit number R_SPEA_RLE0 R_SPEA_RLE1 R_SPEA_RLE2 R_SPEA_RLE3 R_SPEA_SU0 R_SPEA_SU1 R_SPEA_SU2 R_SPEA_SU3

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to change the state of Sprite unit or RLE unit.
SPEA driver requires managing 8 layer (RLE unit 0~3/Sprite unit 0~3) status.
This function is called from following APIs.

- R_SPEA_Init
- R_SPEA_DeInit
- R_SPEA_UnitEnable
- R_SPEA_SpriteEnable

Default Behavior

This function executes the following processing.

- Changes the current status of Sprite unit or RLE unit.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_SPEA_ERR_OK	- No error has occurred.
R_SPEA_ERR_NG	- Invalid argument.

See Also

r_spea_State_t

5.2.8 R_SPEA_SYS_StateGet

Function Prototypes

```
r_spea_State_t R_SPEA_Sys_StateGet(const uint32_t       Unit
                                   const r_spea_Unit_t  SpUnit)
```

Input Parameter

Table 5-7 Input parameter of R_SPEA_SYS_StateGet

Parameter	Description
Unit	Specifies the SPEA Unit number.
SpUnit	Specifies RLE unit or Sprite unit number R_SPEA_RLE0 R_SPEA_RLE1 R_SPEA_RLE2 R_SPEA_RLE3 R_SPEA_SU0 R_SPEA_SU1 R_SPEA_SU2 R_SPEA_SU3

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to get the state of SPEA RLE unit.
SPEA driver requires managing 8 layer (RLE unit 0~3/Sprite unit 0~3) status.
This function is called from several APIs.

Default Behavior

This function executes the following processing.

- Returns the current status of Sprite unit or RLE unit.
- If invalid Unit or SpUnit is specified, this function returns the R_SPEA_STATE_UNINITIALIZED.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_SPEA_STATE_UNINITIALIZED	- Sprite or RLE unit is in Uninitialized state
R_SPEA_STATE_IDLE	- Sprite or RLE unit is in Idle state.
R_SPEA_STATE_UPDATING	- Sprite or RLE unit is in Updating state.
R_SPEA_STATE_EXECUTING	- Sprite or RLE unit is in Executing state.

See Also

r_spea_State_t

5.2.9 R_SPEA_SYS_InitGlobal

Function Prototypes

```
void R_SPEA_SYS_InitGlobal(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

Initializes the global variables in SPEA porting layer.

If R_BSP_SYS_INIT_USE is defined, user must call this function before calling SPEA APIs.
This function is called from R_DEV_SysInit provided as sample code.

If R_BSP_SYS_INIT_USE is not defined, global variables are declared with initial values.
This function call is not mandatory.

Default Behavior

This function executes the following processing.

- Initialize global variables.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

5.3 Data types

5.3.1 r_spea_State_t

Description

This type describes the state of SPEA.

Definition

```
typedef enum
{
    R_SPEA_STATE_UNINITIALIZED = 0,
    R_SPEA_STATE_IDLE,
    R_SPEA_STATE_UPDATING,
    R_SPEA_STATE_EXECUTING
} r_spea_State_t
```

Table 5-8 Enumerator of r_spea_State_t

Name	Description
R_SPEA_STATE_UNINITIALIZED	The SPEA driver is an uninitialized state.
R_SPEA_STATE_IDLE	The SPEA driver is an idle state.
R_SPEA_STATE_UPDATING	The SPEA driver is an updating state.
R_SPEA_STATE_EXECUTING	The SPEA driver is an executing state.

See Also

R_SPEA_SYS_StateSet
R_SPEA_SYS_StateGet

6.Video Output Warping Engine (VOWE)

6.1 File list

Following table shows the file list for VOWE porting layer .

Table 6-1 File list for VOWE porting layer

File Name	Pass	Description
r_sys_vowe.c	vlib/device/d1mx/src/vowe	Source file of poring layer.
r_config_vowe.h	vlib/device/d1mx/macro_cfg/vowe	Header file for device configuration.
r_vowe_sys.h	vlib/macro/vo/vowe/lib	Header file for porting layer interface.

6.2 Driver support functions

It is possible to integrate the VOWE into different systems (OS, memory manager etc.). The VOWE driver must not rely on a specific environment. To achieve this, the VOWE driver can use driver support functions. These are not part of the driver itself but they must be provided to integrate the driver on a particular device, OS, board. Driver support functions shall never be called by the application directly, instead they are indirectly called when the application calls the generic VOWE driver API. They have to be implemented within the driver library for a concrete device. (Example D1L, D1M).

6.2.1 R_VOWE_Sys_Init

Function Prototypes

```
r_vowe_Error_t R_VOWE_Sys_Init(const uint32_t Unit)
```

Input Parameter

Table 6-2 Input Parameter of R_VOWE_Sys_Init

Parameter In	Description
Unit	Specifies the VOWE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function initializes environment-dependent part. This function is called from R_VOWE_Init function. This function executes the processing of adjusting display timing (calls R_VDCE_DisplayTimingAdjust function**).**

Default Behavior

This function adjusts the display timing by executing R_VDCE_DisplayTimingAdjust function.

Customizing Points

Adjust the Vlines parameter of R_VDCE_DisplayTimingAdjust function.

Return Codes

R_VOWE_ERR_OK	- No error had occurred.
R_VOWE_ERR_RANGE_UNIT	- Unit number was out of range.
R_VOWE_ERR_SYS_VDCE	- The error has occurred at the driver support function of VDCE driver.

See Also

None

6.2.2 R_VOWE_Sys_DeInit

Function Prototypes

```
r_vowe_Error_t R_VOWE_Sys_DeInit(const uint32_t Unit)
```

Input Parameter

Table 6-3 Input Parameter of R_VOWE_Sys_DeInit

Parameter In	Description
Unit	Specifies the VOWE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function de-initializes environment-dependent part. This function is called from R_VOWE_DeInit function.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.
If user want to add the de-initialization code of environment-depend (e.g. clock control), then implement to this function.

Return Codes

R_VOWE_ERR_OK	- No error had occurred.
R_VOWE_ERR_RANGE_UNIT	- Unit number was out of range.

See Also

None

6.2.3 R_VOWE_Sys_InterruptEnable

Function Prototypes

```
void R_VOWE_Sys_InterruptEnable(const uint32_t Unit)
```

Input Parameter

Table 6-4 Input Parameter of R_VOWE_Sys_InterruptEnable

Parameter In	Description
Unit	Specifies the VOWE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function enables the interrupt request. This function is called from R_VOWE_Start function.

Default Behavior

This function executes the processing for enabling the INTVOWE interrupt.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

6.2.4 R_VOWE_Sys_InterruptDisable

Function Prototypes

```
void R_VOWE_Sys_InterruptDisable(const uint32_t Unit)
```

Input Parameter

Table 6-5 Input Parameter of R_VOWE_Sys_InterruptDisable

Parameter In	Description
Unit	Specifies the VOWE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function disables the interrupt request. This function is called from the callback function of VOWE.

Default Behavior

This function executes the processing for disabling the INTVOWE interrupt.

Return Codes

None

See Also

None

6.2.5 R_VOWE_Sys_BaseAddrGet

Function Prototypes

```
uint32_t R_VOWE_Sys_BaseAddrGet(const uint32_t Unit)
```

Input Parameter

Table 6-6 Input Parameter of R_VOWE_Sys_BaseAddrGet

Parameter In	Description
Unit	Specifies the VOWE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function gives back the base address of VOWE H/W register. This function is called from several VOWE APIs.

Default Behavior

This function returns the base address of VOWE H/W register.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- VOWE H/W base address.
0	- Unit is invalid.

See Also

None

6.2.6 R_VOWE_Sys_ExtBaseAddrGet

Function Prototypes

```
uint32_t R_VOWE_Sys_ExtBaseAddrGet(const uint32_t Unit)
```

Input Parameter

Table 6-7 Input Parameter of R_VOWE_Sys_ExtBaseAddrGet

Parameter In	Description
Unit	Specifies the VOWE Unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function gives back the base address of extra register. This function is called from several VOWE APIs.

Default Behavior

This function returns the extra base address of VOWE.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- H/W base address of Ring buffer setting.
0	- Unit is invalid.

See Also

None

6.2.7 R_VOWE_Sys_VDCEnable

Function Prototypes

```
r_vowe_Error_t R_VOWE_Sys_VDCEnable(const uint32_t           Unit,
                                    const r_vowe_BufferMode_t  WorkBufferMode,
                                    const uint8_t * const      VdceBufferAddr,
                                    const uint32_t             VdceBufferSize,
                                    const uint32_t             Stride,
                                    const r_vowe_ColorFormat_t ColorFormat,
                                    const uint32_t             SourceWidth,
                                    const uint32_t             SourceHeight,
                                    const uint32_t             RingBufferDelay,
                                    const r_vowe_DestMode_t    DestMode)
```

Input Parameter

Table 6-8 Input Parameter of R_VOWE_Sys_VDCEnable

Parameter	Description
Unit	Specifies the VOWE unit number.
WorkBufferMode	Specifies the work buffer mode.
VdceBufferAddr	Specifies the top address of the work buffer to set to VDCE.
VdceBufferSize	Specifies the size of work bufffer.
Stride	Specifies the stride of output image.
ColorFormat	Specifies the color format of output image.
SourceWidth	Specifies the source image width (pixel).
SourceHeight	Specifies the source image height (pixel).
RingBufferDelay	Specifies the delay line of ring buffer.
DestMode	Specifies the destination mode.

Input – Output Parameter

None

Output Parameter

None

Description

This function enables the VDCE's feature for VOWE driver and allocates the VDCE's resources for VOWE driver. This function is called from R_VOWE_Open.

Default Behavior

This function executes the following processing:

- Sets base address by executing R_VDCE_OirBaseSet.
- Sets buffer mode by executing R_VDCE_OirRingBufferEnable or R_VDCE_OirRingBufferDisable.
- Sets delay line by executing R_VDCE_OirVSyncDelaySet in case of ring buffer mode.
- Sets stride by executing R_VDCE_OirMemGeometrySet.
- Sets image width and height (add 4 lines to the actual height) by executing R_VDCE_OirViewPortSet.
- Sets output color format by executing R_VDCE_OirFormatSet.
- Sets destination mode by executing R_VDCE_OirModeSet.
- Starts the Output Image Generator block (OIR) by executing R_VDCE_OirEnable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VOWE_ERR_OK	- No error had occurred.
R_VOWE_ERR_RANGE_UNIT	- Unit number was out of range.
R_VOWE_ERR_SYS_VDCE	- The error has occurred at the driver support function of VDCE driver.

See Also

None

6.2.8 R_VOWE_Sys_VDCDisable

Function Prototypes

```
r_vowe_Error_t R_VOWE_Sys_VDCDisable(const uint32_t Unit)
```

Input Parameter

Table 6-9 Input Parameter of R_VOWE_Sys_VDCDisable

Parameter	Description
Unit	Specifies the VOWE unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function disables the VDCE's feature for VOWE driver and releases the VDCE's resources for VOWE driver. This function is called from R_VOWE_Close.

Default Behavior

This function executes the processing for stopping the OIR by R_VDCE_OirDisable.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VOWE_ERR_OK	- No error had occurred.
R_VOWE_ERR_RANGE_UNIT	- Unit number was out of range.
R_VOWE_ERR_SYS_VDCE	- The error has occurred at the driver support function of VDCE driver.

See Also

None

6.2.9 R_VOWE_Sys_StateSet

Function Prototypes

```
r_vowe_Error_t R_VOWE_Sys_StateSet(const uint32_t        Unit,
                                   const r_vowe_Status_t State)
```

Input Parameter

Table 6-10 Input Parameter of R_VOWE_Sys_StateSet

Parameter	Description
Unit	Specifies the VOWE unit number.
State	Specifies the required state of VOWE driver. R_VOWE_STATE_DEINIT R_VOWE_STATE_INIT R_VOWE_STATE_IDLE R_VOWE_STATE_EXEC

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to change the state of VOWE driver.
VOWE driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function changes the current state of VOWE driver.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VOWE_ERR_OK	- No error has occurred.
R_VOWE_ERR_RANGE_UNIT	- Unit number was out of range.

See Also

r_vowe_Status_t

6.2.10 R_VOWE_Sys_StateGet

Function Prototypes

```
r_vowe_Status_t R_VOWE_Sys_StateGet(const uint32_t Unit)
```

Input Parameter

Table 6-11 Input Parameter of R_VOWE_Sys_StateGet

Parameter	Description
Unit	Specifies the VOWE unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to get the status of VOWE driver.
VOWE driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function executes the following processing:

- Returns the current state of VOWE driver.
- If invalid unit number is specified, this function returns R_VOWE_STATE_DEINIT.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VOWE_STATE_DEINIT	- VOWE driver is not initialized.
R_VOWE_STATE_INIT	- VOWE driver is initialized and all the internal variables are set to default values.
R_VOWE_STATE_IDLE	- All the configuration has been set and no warping have occurred yet.
R_VOWE_STATE_EXEC	- VOWE driver is performing a warping.

See Also

r_vowe_Status_t

6.2.11 R_VOWE_Sys_InitGlobal

Function Prototypes

```
void R_VOWE_Sys_InitGlobal(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

Initializes the global variables in VOWE porting layer.

If R_BSP_SYS_INIT_USE is defined, user must call this function before calling VOWE APIs.
This function is called from R_DEV_SysInit provided as sample code.

If R_BSP_SYS_INIT_USE is not defined, global variables are declared with initial values.
This function call is not mandatory.

Default Behavior

This function executes the following processing.

- Initialize global variables.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

6.3 OS interface

6.3.1 R_VOWE_Sys_Lock

Function Prototypes

```
r_vowe_Error_t R_VOWE_Sys_Lock(const uint32_t Unit)
```

Input Parameter

Table 6-12 Input Parameter of R_VOWE_Sys_Lock

Parameter	Description
Unit	Specifies the VOWE unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the VOWE driver access to the specified unit for other threads. This function is called from several VOWE APIs.

Default Behavior

This function does nothing, this function is empty as default.

Customizing Points

User needs to implement the locking process by mutex or semaphore if VOWE API is called from multi-thread.

Return Codes

R_VOWE_ERR_OK	- No error had occurred.
R_VOWE_ERR_RANGE_UNIT	- Unit number was out of range.

See Also

None

6.3.2 R_VOWE_Sys_Unlock

Function Prototypes

```
r_vowe_Error_t R_VOWE_Sys_Unlock(const uint32_t   Unit)
```

Input Parameter

Table 6-13 Input Parameter of R_VOWE_Sys_Unlock

Parameter	Description
Unit	Specifies the VOWE unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the VOWE driver access to the specified unit for other threads. This function is called from several VOWE APIs.

Default Behavior

This function does nothing, this function is empty as default.

Customizing Points

Implement the unlock process depending on R_VOWE_Sys_Lock.

Return Codes

R_VOWE_ERR_OK	- No error had occurred.
R_VOWE_ERR_RANGE_UNIT	- Unit number was out of range.

See Also

None

6.4 Data types

6.4.1 r_vowe_Status_t

Description

This type describes the VOWE driver state.

Definition

```
typedef enum
{
    R_VOWE_STATE_DEINIT = 0,
    R_VOWE_STATE_INIT,
    R_VOWE_STATE_IDLE,
    R_VOWE_STATE_EXEC
} r_vowe_Status_t
```

Table 6-14 Enumerator of r_vowe_Status_t

Name	Description
R_VOWE_STATE_DEINIT	VOWE driver in de-initialized state
R_VOWE_STATE_INIT	VOWE driver in initialized state.
R_VOWE_STATE_IDLE	VOWE driver in idle state.
R_VOWE_STATE_EXEC	VOWE driver in executing state.

See Also

None

7.JPEG Codec Unit A (JCUA)

7.1 File list

Following table shows the file list for JCUA porting layer .

Table 7-1 File list for JCUA porting layer

File Name	Pass	Description
r_sys_jcua.c	vlib/device/d1mx/src/jcua	Source file of poring layer.
r_config_jcua.h	vlib/device/d1mx/macro_cfg/jcua	Header file for device configuration.
r_jcua_sys.h	vlib/macro/vo/jcua/lib	Header file for porting layer interface.

7.2 Driver support functions

It is possible to integrate the JCUA into different systems (OS, memory manager etc.). The JCUA driver must not rely on a specific environment. To achieve this, the JCUA driver can use driver support functions. These are not part of the driver itself but they must be provided to integrate the driver on a particular device, OS, board. Driver support functions shall never be called by the application directly, instead they are indirectly called when the application calls the generic JCUA driver API. They have to be implemented within the driver library for a concrete device. (Example D1L, D1M).

7.2.1 R_JCUA_Sys_Init

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_Init(const uint32_t Unit)
```

Input Parameter

Table 7-2 Input Parameter of R_JCUA_Sys_Init

Parameter	Description
Unit	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function initializes environment-dependent (e.g. interrupt priority, power control or clock control) part. This function is called from API function R_JCUA_Init.

Default Behavior

This function executes the following processing.

- Bus reset for JCUA macro.

Since JCUA macro internal reset not sufficient, software reset is used for resetting JCUA. JPEG Codec Unit A Software reset register(JCSWRST) is used for setting software reset. Use the following software reset procedure to reset the JCUA unit:

- Wait for 30us.
- Set JCSWRST.JCUA0RES bit to 1, to generate the software reset condition.
- Read JCSWRST.JCUA0RES bit until this bit is set to 1. wait for software reset is active.
- Set JCSWRST.JCUA0RES bit to 0, to release software reset condition.

By this sequence all JCUA registers with exception of the JCSWRST register are initialized by a software reset.

- Enables the JEDI and JDTI interrupt for JCUA H/W macro.

Customizing Points

It is not necessary modify this function in general use-case. But this function is using the TICK driver or OSTM driver for wait process. If user don't want to use TICK driver or OSTM driver, modify timeout measurement process.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_BUS_TIMEOUT	- Timeout error has occurred at bus reset.
R_JCUA_ERR_TIMER_CTRL	- An error has occurred at Timer unit.

See Also

None

7.2.2 R_JCUA_Sys_DeInit

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_DeInit(const uint32_t   Unit)
```

Input Parameter

Table 7-3 Input Parameter of R_JCUA_Sys_DeInit

Parameter	Description
Unit	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function de-initializes environment-dependent part. This function is called from R_JCUA_DeInit.

Default Behavior

This function executes the following processing.

- Disables the JEDI and JDTI interrupt for JCUA H/W macro.
- Close the H/W timer for the wait process. (if necessary)

Customizing Points

It is not necessary to modify this function in general use-case. But this function is using the TICK driver or OSTM driver for wait process. If user don't want to use TICK driver or OSTM driver, modify timeout measurement process.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_TIMER_CTRL	- An error has occurred at Timer unit

See Also

None

7.2.3 R_JCUA_Sys_InterruptEnable

Function Prototypes

```
void R_JCUA_Sys_InterruptEnable(const uint32_t   Unit)
```

Input Parameter

Table 7-4 Input Parameter of R_JCUA_Sys_InterruptEnable

Parameter	Description
Unit	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function enables interrupt request. This function is called from R_JCUA_DecoderStart and R_JCUA_DecoderContinue.

Default Behavior

This function executes the following processing.

- Enables the JEDI and JDTI interrupt

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

7.2.4 R_JCUA_Sys_InterruptDisable

Function Prototypes

```
void R_JCUA_Sys_InterruptDisable(const uint32_t Unit)
```

Input Parameter

Table 7-5 Input Parameter of R_JCUA_Sys_InterruptDisable

Parameter	Description
Unit	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function disables interrupt. This function is called from several JCUA APIs.

Default Behavior

This function executes the following processing.

- Disables the JEDI and JDTI interrupt.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

7.2.5 R_JCUA_Sys_BaseAddrGet

Function Prototypes

```
uint32_t R_JCUA_Sys_BaseAddrGet(const uint32_t Unit)
```

Input Parameter

Table 7-6 Input Parameter of R_JCUA_Sys_BaseAddrGet

Parameter	Description
Unit	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function gives back the base address of JCUA H/W register. This function is called from several JCUA APIs.

Default Behavior

This function executes the following processing.

- Returns the base address of JCUA H/W register.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- Base address of JCUA H/W register.
0	- Unit is invalid

See Also

None

7.2.6 R_JCUA_Sys_TimerStart

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_TimerStart(const uint32_t Unit,
                                     const uint32_t IsHeader)
```

Input Parameter

Table 7-7 Input Parameter of R_JCUA_Sys_TimerStart

Parameter	Description
Unit	Specifies the JCUA unit number.
IsHeader	Specifies the decoding part R_TRUE : JPEG Header part. R_FALSE : JPEG Image data part.

Input – Output Parameter

None

Output Parameter

None

Description

This function starts timeout measurement. This function is called from R_JCUA_DecoderStart and R_JCUA_IsrStop. This may be called from ISR depending on the system.

Default Behavior

This function executes the following processing.

- Starts the timer for timeout measurement.
- Calls R_JCUA_IsrTimeOut callback function when timeout occurs. (This process is executed by loc_OstmCallback.)

Customizing Points

- It is not necessary to modify this function in general use-case. But this function is using the OSTM driver for timeout measurement. If user don't want to use OSTM driver, modify timeout measurement process.
- Timeout value of Header part and Image Data part is represented in micro second. If timeout is not needed, user should define both by 0.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_TIMER_CTRL	- An error has occurred at Timer unit.

See Also

None

7.2.7 R_JCUA_Sys_TimerStop

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_TimerStop(const uint32_t Unit)
```

Input Parameter

Table 7-8 Input Parameter of R_JCUA_Sys_TimerStop

Parameter	Description
Unit	Specifies the JCUA unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function stops timeout measurement. This function is called from R_JCUA_IsrFinish and R_JCUA_IsrStop. These may be called from ISR depending on the system.

Default Behavior

This function executes the following processing.

- Stops the timer for timeout measurement.

Customizing Points

It is not necessary to modify this function in general use-case.
But this function is using the OSTM driver for timeout measurement. If user don't want to use OSTM driver, modify timeout measurement process.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_TIMER_CTRL	- An error has occurred at Timer unit.

See Also

None

7.2.8 R_JCUA_Sys_TimerPause

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_TimerPause(const uint32_t Unit)
```

Input Parameter

Table 7-9 Input Parameter of R_JCUA_Sys_TimerPause

Parameter	Description
Unit	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function suspends timeout measurement (pause). This function stores the remaining counter until timeout. This function is called from R_JCUA_IsrFinish. This may be called from ISR depending on the system.

Default Behavior

This function executes the following processing.

- Pauses the timer for timeout measurement

Customizing Points

It is not necessary to modify this function in general use-case.
But this function is using the OSTM driver for timeout measurement. If user don't want to use OSTM driver, modify timeout measurement process.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_TIMER_CTRL	- An error has occurred at Timer unit.

See Also

None

7.2.9 R_JCUA_Sys_TimerResume

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_TimerResume(const uint32_t Unit)
```

Input Parameter

Table 7-10 Input Parameter of R_JCUA_Sys_TimerResume

Parameter	Description
Unit	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function resumes timeout measurement.
The timeout value is the remaining count value stored at R_JCUA_Sys_TimerPause.
This function is called from R_JCUA_DecoderContinue.

Default Behavior

This function executes the following processing.

- Resumes the timer for timeout measurement.

Customizing Points

It is not necessary to modify this function in general use-case.
But this function is using the OSTM driver for timeout measurement. If user don't want to use OSTM driver, modify timeout measurement process.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_TIMER_CTRL	- An error has occurred at Timer unit.

See Also

None

7.2.10 R_JCUA_Sys_StateSet

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_StateSet(const uint32_t       Unit,
                                   const r_jcua_State_t State)
```

Input Parameter

Table 7-11 Input Parameter of R_JCUA_Sys_StateSet

Parameter	Description
Unit	Specifies the JCUA unit number.
State	Specifies the requested state of JCUA driver R_JCUA_STATE_UNINITIALIZED R_JCUA_STATE_INITIALIZED R_JCUA_STATE_IDLE R_JCUA_STATE_EXECUTING R_JCUA_STATE_PAUSED

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to change the state of JCUA driver.
JCUA driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function changes the current status of JCUA driver.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.

See Also

r_jcua_State_t

7.2.11 R_JCUA_Sys_StateGet

Function Prototypes

```
r_jcua_State_t R_JCUA_Sys_StateGet(const uint32_t      Unit)
```

Input Parameter

Table 7-12 Input Parameter of R_JCUA_Sys_StateGet

Parameter	Description
`Unit`	Specifies the JCUA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to get the state of JCUA driver.
JCUA driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function executes the following processing.

- Returns the current status of JCUA driver.
- If invalid Unit is specified, this function returns R_JCUA_STATE_UNINITIALIZED.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_JCUA_STATE_UNINITIALIZED	- JCUA driver is Uninitialized state.
R_JCUA_STATE_INITIALIZED	- JCUA driver is Initialized state.
R_JCUA_STATE_IDLE	- JCUA driver is IDLE state
R_JCUA_STATE_EXECUTING	- JCUA driver is Executing state.
R_JCUA_STATE_PAUSED	- JCUA driver is Pause state.

See Also

r_jcua_State_t

7.2.12 R_JCUA_Sys_InitGlobal

Function Prototypes

```
void R_JCUA_Sys_InitGlobal(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

Initializes the global variables in JCUA porting layer.

If R_BSP_SYS_INIT_USE is defined, user must call this function before calling JCUA APIs.
This function is called from R_DEV_SysInit provided as sample code.

If R_BSP_SYS_INIT_USE is not defined, global variables are declared with initial values.
This function call is not mandatory.

Default Behavior

This function executes the following processing.

- Initialize global variables.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

7.3 OS interface

The JCUA driver shall support access from multiple threads in a multi-threading environment. In case of atomic data manipulation (e.g. adding events to a queue), it has to be possible to avoid concurrent access to the same data structure.

7.3.1 R_JCUA_Sys_Lock

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_Lock(const uint32_t Unit)
```

Input Parameter

Table 7-13 Input Parameter of R_JCUA_Sys_Lock

Parameter	Description
Unit	Specifies the JCUA unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the JCUA driver access to the specified unit from other threads. This function is called from several JCUA APIs.

Default Behavior

This function is empty as default.

Customizing Points

User need to implement the lock process by mutex or semaphore if JCUA API is called from multi-thread.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

See Also

None

7.3.2 R_JCUA_Sys_Unlock

Function Prototypes

```
r_jcua_Error_t R_JCUA_Sys_Unlock(const uint32_t Unit)
```

Input Parameter

Table 7-14 Input Parameter of R_JCUA_Sys_Unlock

Parameter	Description
Unit	Specifies the JCUA unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the JCUA driver access to the specified unit from other threads. This function is called from several JCUA APIs.

Default Behavior

This function is empty as default.

Customizing Points

User need to implement the unlock process depending on R_JCUA_Sys_Lock.

Return Codes

R_JCUA_ERR_OK	- No error has occurred.
R_JCUA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_JCUA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

See Also

None

7.4 Data types

7.4.1 r_jcua_State_t

Description

This enumerator indicates a main status of JCUA driver.

Definition

```
typrdef enum
{
    R_JCUA_STATE_UNINITIALIZED = 0,
    R_JCUA_STATE_INITIALIZED,
    R_JCUA_STATE_IDLE,
    R_JCUA_STATE_EXECUTING,
    R_JCUA_STATE_PAUSED
} r_jcua_State_t
```

Table 7-15 Enumerator of r_jcua_State_t

Name	Description
R_JCUA_STATE_UNINITIALIZED	This is a status that the JCUA driver is not initialized.
R_JCUA_STATE_INITIALIZED	This is a status that the operation mode of JCUA driver is not set after the JCUA driver is initialized.
R_JCUA_STATE_IDLE	This is a status that the operation mode of JCUA drive is set to a decoding mode. The decoding is not performed yet.
R_JCUA_STATE_EXECUTING	This is a status that the JCUA driver is performing a decoding.
R_JCUA_STATE_PAUSED	This is a status that a pause caused by input division or output division is occurring after JCUA driver performed a decoding. When a decoding is restarted, the status transition to a decoding status occurs.

See Also

None

8.Video Output Checker A (VOCA)

8.1 File list

Following table shows the file list for VOCA porting layer .

Table 8-1 File list for VOCA porting layer

File Name	Pass	Description
r_sys_voca.c	vlib/device/d1mx/src/voca	Source file of poring layer.
r_config_voca.h	vlib/device/d1mx/macro_cfg/voca	Header file for device configuration.
r_voca_sys.h	vlib/macro/vo/voca/lib	Header file for porting layer interface.

8.2 Driver support functions

It is possible to integrate the VOCA into different systems (OS, memory manager etc.). The VOCA driver must not rely on a specific environment. To achieve this, the VOCA driver can use driver support functions. These are not part of the driver itself but they must be provided to integrate the driver on a particular device, OS, board. Driver support functions shall never be called by the application directly, instead they are indirectly called when the application calls the generic VOCA driver API. They have to be implemented within the driver library for a concrete device. (Example D1L, D1M).

8.2.1 R_VOCA_Sys_Init

Function Prototypes

```
r_voca_Error_t R_VOCA_Sys_Init(const uint32_t Unit)
```

Input Parameter

Table 8-2 Input Parameter of R_VOCA_Sys_Init

Parameter	Description
Unit	Specifies the VOCA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function initializes environment-dependent (e.g. interrupt priority, power control or clock control) part. This function is called from API function R_VOCA_Init.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.
If user want to add the de-initialization code of environment-depend (e.g. clock control), then implement to this function.

Return Codes

R_VOCA_ERR_OK	- No error has occurred.
R_VOCA_ERR_RANGE_UNIT	- The unit number is outside the range.

See Also

None

8.2.2 R_VOCA_Sys_DeInit

Function Prototypes

```
r_voca_Error_t R_VOCA_Sys_DeInit(const uint32_t   Unit)
```

Input Parameter

Table 8-3 Input Parameter of R_VOCA_Sys_DeInit

Parameter	Description
Unit	Specifies the VOCA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function de-initializes environment-dependent part. This function is called from R_VOCA_DeInit.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.
If user want to add the de-initialization code of environment-depend (e.g. clock control), then implement to this function.

Return Codes

R_VOCA_ERR_OK	- No error has occurred.
R_VOCA_ERR_RANGE_UNIT	- The unit number is outside the range.

See Also

None

8.2.3 R_VOCA_Sys_InterruptEnable

Function Prototypes

```
void R_VOCA_Sys_InterruptEnable(const uint32_t   Unit)
```

Input Parameter

Table 8-4 Input Parameter of R_VOCA_Sys_InterruptEnable

Parameter	Description
Unit	Specifies the VOCA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function enables interrupt request. This function is called from R_VOCA_IntEnable.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

8.2.4 R_VOCA_Sys_InterruptDisable

Function Prototypes

```
void R_VOCA_Sys_InterruptDisable(const uint32_t Unit)
```

Input Parameter

Table 8-5 Input Parameter of R_VOCA_Sys_InterruptDisable

Parameter	Description
Unit	Specifies the VOCA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function disables interrupt. This function is called from R_VOCA_IntDisable.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

8.2.5 R_VOCA_Sys_BaseAddrGet

Function Prototypes

```
uint32_t R_VOCA_Sys_BaseAddrGet(const uint32_t Unit)
```

Input Parameter

Table 8-6 Input Parameter of R_VOCA_Sys_BaseAddrGet

Parameter	Description
Unit	Specifies the VOCA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function gives back the base address of VOCA H/W register. This function is called from several VOCA APIs.

Default Behavior

This function executes the following processing.

- Returns the base address of VOCA H/W register.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- Base address of VOCA H/W register.
0	- Unit is invalid

See Also

None

8.2.6 R_VOCA_Sys_MaxVideoChannelGet

Function Prototypes

```
uint32_t R_VOCA_Sys_MaxVideoChannelGet(const uint32_t Unit)
```

Input Parameter

Table 8-7 Input Parameter of R_VOCA_Sys_MaxVideoChannelGet

Parameter	Description
Unit	Specifies the VOCA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function gives back the maximum number of Video channel depending on target device. This function is called from R_VOCA_Init.

Default Behavior

This function executes the following processing.

- Stores the maximum video channel depending on the target device.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- The maximum number of Video channel depending on target device.
0	- Unit is invalid

See Also

None

8.2.7 R_VOCA_Sys_StateSet

Function Prototypes

```
r_voca_Error_t R_VOCA_Sys_StateSet(const uint32_t       Unit,
                                   const r_voca_State_t State)
```

Input Parameter

Table 8-8 Input Parameter of R_VOCA_Sys_StateSet

Parameter	Description
Unit	Specifies the VOCA unit number.
State	Specifies the requested state of VOCA driver R_VOCA_STATE_UNINITIALIZED R_VOCA_STATE_INITIALIZED R_VOCA_STATE_IDLE R_VOCA_STATE_EXECUTING

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to change the state of VOCA driver.
VOCA driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function changes the current status of VOCA driver.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VOCA_ERR_OK	- No error has occurred.
R_VOCA_ERR_RANGE_UNIT	- The unit number is outside the range.

See Also

r_voca_State_t

8.2.8 R_VOCA_Sys_StateGet

Function Prototypes

```
r_voca_State_t R_VOCA_Sys_StateGet(const uint32_t      Unit)
```

Input Parameter

Table 8-9 Input Parameter of R_VOCA_Sys_StateGet

Parameter	Description
Unit	Specifies the VOCA unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to get the state of VOCA driver.
VOCA driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function executes the following processing.

- Returns the current status of VOCA driver.
- If invalid Unit is specified, this function returns R_VOCA_STATE_UNINITIALIZED.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_VOCA_STATE_UNINITIALIZED	- VOCA driver is Uninitialized state.
R_VOCA_STATE_INITIALIZED	- VOCA driver is Initialized state.
R_VOCA_STATE_IDLE	- VOCA driver is IDLE state
R_VOCA_STATE_EXECUTING	- VOCA driver is Executing state.

See Also

r_voca_State_t

8.2.9 R_VOCA_Sys_InitGlobal

Function Prototypes

```
void R_VOCA_Sys_InitGlobal(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

Initializes the global variables in VOCA porting layer.

If R_BSP_SYS_INIT_USE is defined, user must call this function before calling VOCA APIs.
This function is called from R_DEV_SysInit provided as sample code.

If R_BSP_SYS_INIT_USE is not defined, global variables are declared with initial values.
This function call is not mandatory.

Default Behavior

This function executes the following processing.

- Initialize global variables.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

8.3 OS interface

The VOCA driver shall support access from multiple threads in a multi-threading environment. In case of atomic data manipulation (e.g. adding events to a queue), it has to be possible to avoid concurrent access to the same data structure.

8.3.1 R_VOCA_Sys_Lock

Function Prototypes

```
r_voca_Error_t R_VOCA_Sys_Lock(const uint32_t Unit)
```

Input Parameter

Table 8-10 Input Parameter of R_VOCA_Sys_Lock

Parameter	Description
Unit	Specifies the VOCA unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the VOCA driver access to the specified unit from other threads. This function is called from several VOCA APIs.

Default Behavior

This function is empty as default.

Customizing Points

User need to implement the lock process by mutex or semaphore if VOCA API is called from multi-thread.

Return Codes

R_VOCA_ERR_OK	- No error has occurred.
R_VOCA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_VOCA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

See Also

None

8.3.2 R_VOCA_Sys_Unlock

Function Prototypes

```
r_voca_Error_t R_VOCA_Sys_Unlock(const uint32_t Unit)
```

Input Parameter

Table 8-11 Input Parameter of R_VOCA_Sys_Unlock

Parameter	Description
Unit	Specifies the VOCA unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the VOCA driver access to the specified unit from other threads. This function is called from several VOCA APIs.

Default Behavior

This function is empty as default.

Customizing Points

User need to implement the unlock process depending on R_VOCA_Sys_Lock.

Return Codes

R_VOCA_ERR_OK	- No error has occurred.
R_VOCA_ERR_RANGE_UNIT	- The unit number is outside the range.
R_VOCA_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

See Also

None

8.4 Data types

8.4.1 r_voca_State_t

Description

This enumerator indicates a main status of VOCA driver.

Definition

```
typrdef enum
{
    R_VOCA_STATE_UNINITIALIZED = 0,
    R_VOCA_STATE_INITIALIZED,
    R_VOCA_STATE_IDLE,
    R_VOCA_STATE_EXECUTING
} r_voca_State_t
```

Table 8-12 Enumerator of r_voca_State_t

Name	Description
R_VOCA_STATE_UNINITIALIZED	This is a status that the VOCA driver is not initialized.
R_VOCA_STATE_INITIALIZED	This is a status that the VOCA driver is initialized.
R_VOCA_STATE_IDLE	This is a status that the VOCA driver is idled.
R_VOCA_STATE_EXECUTING	This is a status that the VOCA driver is executed.

See Also

None

9.Display Output Comparator (DISCOM)

9.1 File list

Following table shows the file list for DISCOM porting layer .

Table 9-1 File list for DISCOM porting layer

File Name	Pass	Description
r_sys_discom.c	vlib/device/d1mx/src/discom	Source file of poring layer.
r_config_discom.h	vlib/device/d1mx/macro_cfg/discom	Header file for device configuration.
r_discom_sys.h	vlib/macro/vo/discom/lib	Header file for porting layer interface.

9.2 Driver support functions

It is possible to integrate the DISCOM into different systems (OS, memory manager etc.). The DISCOM driver must not rely on a specific environment. To achieve this, the DISCOM driver can use driver support functions. These are not part of the driver itself but they must be provided to integrate the driver on a particular device, OS, board. Driver support functions shall never be called by the application directly, instead they are indirectly called when the application calls the generic DISCOM driver API. They have to be implemented within the driver library for a concrete device. (Example D1L, D1M).

9.2.1 R_DISCOM_Sys_Init

Function Prototypes

```
r_discom_Error_t R_DISCOM_Sys_Init(const uint32_t Unit)
```

Input Parameter

Table 9-2 Input Parameter of R_DISCOM_Sys_Init

Parameter	Description
Unit	Specifies the DISCOM unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function initializes environment-dependent (e.g. interrupt priority, power control or clock control) part. This function is called from API function R_DISCOM_Init.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.
If user want to add the de-initialization code of environment-depend (e.g. clock control), then implement to this function.

Return Codes

R_DISCOM_ERR_OK	- No error has occurred.
R_DISCOM_ERR_RANGE_UNIT	- The unit number is outside the range.

See Also

None

9.2.2 R_DISCOM_Sys_DeInit

Function Prototypes

```
r_discom_Error_t R_DISCOM_Sys_DeInit(const uint32_t   Unit)
```

Input Parameter

Table 9-3 Input Parameter of R_DISCOM_Sys_DeInit

Parameter	Description
Unit	Specifies the DISCOM unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function de-initializes environment-dependent part. This function is called from R_DISCOM_DeInit.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.
If user want to add the de-initialization code of environment-depend (e.g. clock control), then implement to this function.

Return Codes

R_DISCOM_ERR_OK	- No error has occurred.
R_DISCOM_ERR_RANGE_UNIT	- The unit number is outside the range.

See Also

None

9.2.3 R_DISCOM_Sys_InterruptEnable

Function Prototypes

```
void R_DISCOM_Sys_InterruptEnable(const uint32_t   Unit)
```

Input Parameter

Table 9-4 Input Parameter of R_DISCOM_Sys_InterruptEnable

Parameter	Description
Unit	Specifies the DISCOM unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function enables interrupt request. This function is called from R_DISCOM_IntEnable.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

9.2.4 R_DISCOM_Sys_InterruptDisable

Function Prototypes

```
void R_DISCOM_Sys_InterruptDisable(const uint32_t Unit)
```

Input Parameter

Table 9-5 Input Parameter of R_DISCOM_Sys_InterruptDisable

Parameter	Description
Unit	Specifies the DISCOM unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function disables interrupt. This function is called from R_DISCOM_IntDisable.

Default Behavior

This function is empty as default.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

9.2.5 R_DISCOM_Sys_BaseAddrGet

Function Prototypes

```
uint32_t R_DISCOM_Sys_BaseAddrGet(const uint32_t Unit)
```

Input Parameter

Table 9-6 Input Parameter of R_DISCOM_Sys_BaseAddrGet

Parameter	Description
Unit	Specifies the DISCOM unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function gives back the base address of DISCOM H/W register. This function is called from several DISCOM APIs.

Default Behavior

This function executes the following processing.

- Returns the base address of DISCOM H/W register.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

not 0	- Base address of DISCOM H/W register.
0	- Unit is invalid

See Also

None

9.2.6 R_DISCOM_Sys_StateSet

Function Prototypes

```
r_discom_Error_t R_DISCOM_Sys_StateSet(const uint32_t       Unit,
                                       const r_discom_State_t State)
```

Input Parameter

Table 9-7 Input Parameter of R_DISCOM_Sys_StateSet

Parameter	Description
Unit	Specifies the DISCOM unit number.
State	Specifies the requested state of DISCOM driver R_DISCOM_STATE_UNINITIALIZED R_DISCOM_STATE_INITIALIZED R_DISCOM_STATE_IDLE R_DISCOM_STATE_EXECUTING

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to change the state of DISCOM driver.
DISCOM driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function changes the current status of DISCOM driver.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_DISCOM_ERR_OK	- No error has occurred.
R_DISCOM_ERR_RANGE_UNIT	- The unit number is outside the range.

See Also

r_discom_State_t

9.2.7 R_DISCOM_Sys_StateGet

Function Prototypes

```
r_discom_State_t R_DISCOM_Sys_StateGet(const uint32_t       Unit)
```

Input Parameter

Table 9-8 Input Parameter of R_DISCOM_Sys_StateGet

Parameter	Description
`Unit`	Specifies the DISCOM unit number.

Input – Output Parameter

None

Output Parameter

None

Description

This function is used to get the state of DISCOM driver.
DISCOM driver requires managing 1status per unit.
This function is called from several APIs.

Default Behavior

This function executes the following processing.

- Returns the current status of DISCOM driver.
- If invalid Unit is specified, this function returns R_DISCOM_STATE_UNINITIALIZED.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

R_DISCOM_STATE_UNINITIALIZED	- DISCOM driver is Uninitialized state.
R_DISCOM_STATE_INITIALIZED	- DISCOM driver is Initialized state.
R_DISCOM_STATE_IDLE	- DISCOM driver is IDLE state
R_DISCOM_STATE_EXECUTING	- DISCOM driver is Executing state.

See Also

r_discom_State_t

9.2.8 R_DISCOM_Sys_InitGlobal

Function Prototypes

```
void R_DISCOM_Sys_InitGlobal(void)
```

Input Parameter

None

Input – Output Parameter

None

Output Parameter

None

Description

Initializes the global variables in DISCOM porting layer.

If R_BSP_SYS_INIT_USE is defined, user must call this function before calling DISCOM APIs.
This function is called from R_DEV_SysInit provided as sample code.

If R_BSP_SYS_INIT_USE is not defined, global variables are declared with initial values.
This function call is not mandatory.

Default Behavior

This function executes the following processing.

- Initialize global variables.

Customizing Points

It is not necessary to modify this function in general use-case.

Return Codes

None

See Also

None

9.3 OS interface

The DISCOM driver shall support access from multiple threads in a multi-threading environment. In case of atomic data manipulation (e.g. adding events to a queue), it has to be possible to avoid concurrent access to the same data structure.

9.3.1 R_DISCOM_Sys_Lock

Function Prototypes

```
r_discom_Error_t R_DISCOM_Sys_Lock(const uint32_t Unit)
```

Input Parameter

Table 9-9 Input Parameter of R_DISCOM_Sys_Lock

Parameter	Description
Unit	Specifies the DISCOM unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function locks the DISCOM driver access to the specified unit from other threads. This function is called from several DISCOM APIs.

Default Behavior

This function is empty as default.

Customizing Points

User need to implement the lock process by mutex or semaphore if DISCOM API is called from multi-thread.

Return Codes

R_DISCOM_ERR_OK	- No error has occurred.
R_DISCOM_ERR_RANGE_UNIT	- The unit number is outside the range.
R_DISCOM_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

See Also

None

9.3.2 R_DISCOM_Sys_Unlock

Function Prototypes

```
r_discom_Error_t R_DISCOM_Sys_Unlock(const uint32_t Unit)
```

Input Parameter

Table 9-10 Input Parameter of R_DISCOM_Sys_Unlock

Parameter	Description
Unit	Specifies the DISCOM unit number

Input – Output Parameter

None

Output Parameter

None

Description

This function unlocks the DISCOM driver access to the specified unit from other threads. This function is called from several DISCOM APIs.

Default Behavior

This function is empty as default.

Customizing Points

User need to implement the unlock process depending on R_DISCOM_Sys_Lock.

Return Codes

R_DISCOM_ERR_OK	- No error has occurred.
R_DISCOM_ERR_RANGE_UNIT	- The unit number is outside the range.
R_DISCOM_ERR_FATAL_OS	- Fatal error has occurred at OS interface.

See Also

None

9.4 Data types

9.4.1 r_discom_State_t

Description

This enumerator indicates a main status of DISCOM driver.

Definition

```
typrdef enum
{
    R_DISCOM_STATE_UNINITIALIZED = 0,
    R_DISCOM_STATE_INITIALIZED,
    R_DISCOM_STATE_IDLE,
    R_DISCOM_STATE_EXECUTING
} r_discom_State_t
```

Table 9-11 Enumerator of r_discom_State_t

Name	Description
R_DISCOM_STATE_UNINITIALIZED	This is a status that the DISCOM driver is not initialized.
R_DISCOM_STATE_INITIALIZED	This is a status that the DISCOM driver is initialized.
R_DISCOM_STATE_IDLE	This is a status that the DISCOM driver is idled.
R_DISCOM_STATE_EXECUTING	This is a status that the DISCOM driver is executed.

See Also

None

10.Appendix

10.1 Memory management

The memory manager handles the allocation of CPU RAM and Video Memory for the RGL.
The source code of this memory manager is available and the user can modify it if necessary. As it is used implicitly by DRW2D and WM driver for the allocation of frame buffer, contexts, textures structures, display lists, etc. It is recommended the API (function names) stay unchanged.

This chapter describes the memory handling functions and the API functions.

For information on memory mapping of the RH850/D1x Device, please refer to the hardware User Manual.

IMPORTANT: The memory manager is not part of the RGL; it is a utility provided for the sake of completion.

10.1.1 Heaps

The memory manager handles the memory allocation in the CPU local RAM and in the video memory. For this reason, the RGL requires 2 heaps:

- a CPU Heap placed in the local RAM
- a Video heap placed in the XBUS memory area

The CPU heap is a static 32-bit array with a size defined by the user at compile time.
The Video Heap is specified by the start address of the XBUS window start address and the size given by the user.

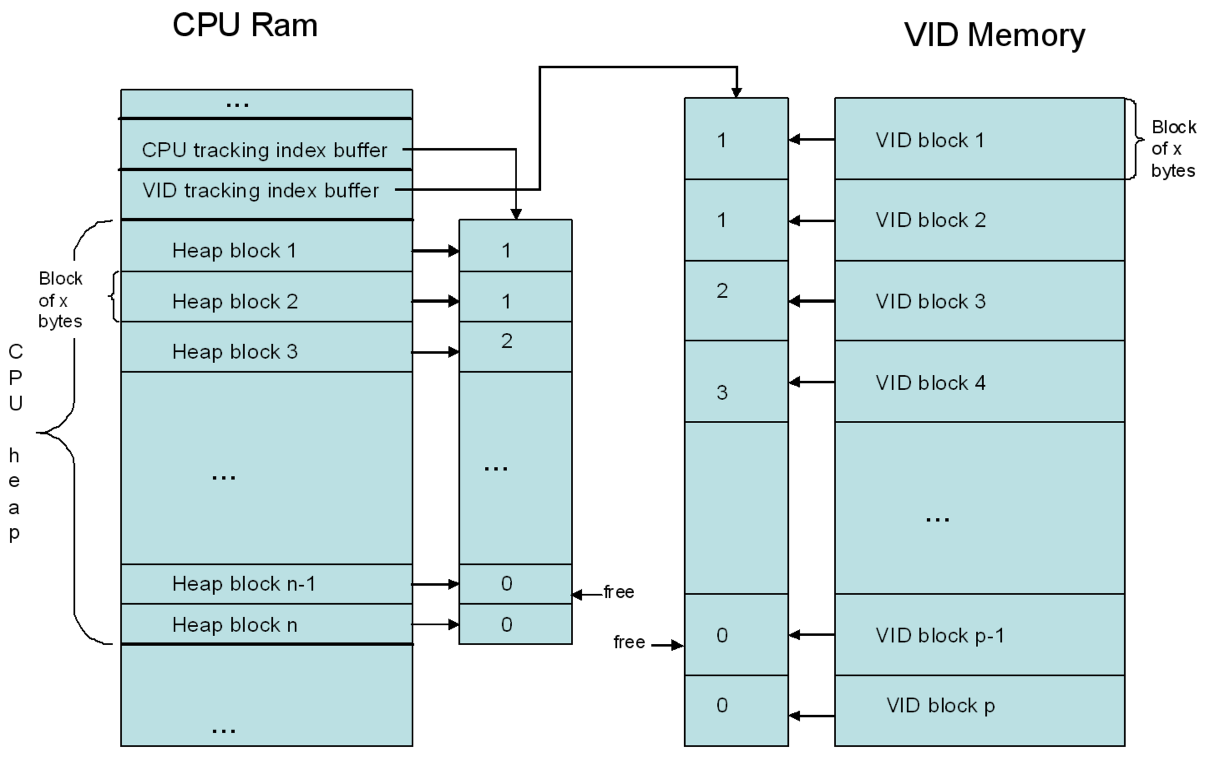
10.1.2 Memory Allocation

In both case the memory allocation happens block wise, which means that there is a minimum allocation size: the size of a block. The number of blocks as well as the size of one block is defined by the user at compile time.

Size and block number should be chosen carefully in order to avoid memory waste (not enough blocks) or performance loss (too many blocks).

10.1.3 Tracking

Each block has a corresponding tracking index of type "`r_cdi_HpBlkIndex_t`" (16-bit integer). If a block is free, its index is zero. All blocks reserved at the same time do have the same index. This index increase by one at every new allocation.
The user must declare two arrays of type "`r_cdi_HpBlkIndex_t`" (one for CPU) and one for VID heap at compile time. The size of the array is the number of blocks of the corresponding heap.



Memory handler

10.1.4 Caution of VDCE layer / VOWE image buffer memory handling

In case of devices where the memory area 0x40000000 (SDRAM) is unoccupied (like D1M1, D1M1-V2, D1L2(H)) the VDCE/WM framebuffers as well as VOWE image buffers should not be placed at the end of the VRAM (0x3xxxxxxx – 0x3fffffff).
Reason is the VDCE H/W reading several bytes implicitly over the 0x3fffffff boundary into the 0x400000xx area. This by itself does no harm (no data is written) but if 0x40000000 has no H/W access (like SDRAM) the VDCE reading can fail.
For details consult the D1x H/W User-manual (section: “Graphics Data Read control”, (6) Restriction).

10.1.5 CDI memory manager API

The function’s prototypes are defined in the header file *vlib/cdi/lib/r_cdi_api.h*. Following functions are available:

- Initialization function for each heap,
- Allocation function,
- Free functions
- Function returning the size of an allocated block row.

All functions related to the CDI are prefixed with “`R_CDI_`”.

10.1.6 Initialization of the heap manager

The function `R_CDI_InitHeapManager` creates a heap manager for a user-definable RAM area (e.g. video RAM). It must be called before the initialization of the 2D drawing engine (Drw2D) and window manager (WM). In the RGL sample software an internal CPU RAM area is initialized (lRAM) as well as the VRAM

It takes as input:

- The start address of the heap
- A control structure for the heap management
- A pointer to an index list
- The number of blocks the heap will be divided into
- The size of one block

Example:

```
/****
Definition of constants for CPU and VID heap Memory initialisation
****/
#define LOC_VID_HP_BLOCK_SIZE  (128u)
#define LOC_VID_HP_BLOCK_NB    (LOC_VIDMEM_SIZE / LOC_VID_HP_BLOCK_SIZE)
#define ALIGN(ADDR,BLOCK)      ( (ADDR+(BLOCK-1)) & ~(BLOCK-1) )
#define VRAM_HEAP_BASE         ALIGN(LOC_VID_FB_BASE, LOC_VID_HP_BLOCK_SIZE)


r_cdi_Heap_t       loc_VRAM_heap;
r_cdi_Heap_t       loc_lRAM_heap;
uint32_t           loc_lRAM_mem[LOC_CPU_HP_BLOCK_NB * LOC_CPU_HP_BLOCK_SIZE / 4];


void loc_SetupMemManager(void)
{
    uint32_t x;

    x = R_CDI_InitHeapManager((uint32_t)&loc_lRAM_mem,
                              &loc_lRAM_heap,
                              (uint32_t)&loc_lRAM_heapIdxList,
                              LOC_CPU_HP_BLOCK_NB,
                              LOC_CPU_HP_BLOCK_SIZE);
    x = R_CDI_InitHeapManager(VRAM_HEAP_BASE,
                              &loc_VRAM_heap,
                              0,
                              LOC_VID_HP_BLOCK_NB,
                              LOC_VID_HP_BLOCK_SIZE);
}
```

10.2 Calibration of SFMA device

SFMA driver needs to set the phase value as argument of R_SFMA_Open to calibrate the phase between SPBCLK, sampling point, and input / output data.
A default calibration value is defined by sample application and will be set when opening the SFMA driver. In normal cases this value is valid throughout the Mango board.
But, the phase value depends on the connected serial flash memory, board design and so on.
Sample BSP has the example code for the calibration.

The following Makefile / GHS.prj preprocessor flags are available to run the calibration

SFMA calibration options
Option (1)

```
   # allow the BSP to init the SFMA
   VLIB_DEFINE     += BSP_INIT_SFMA

   # allow the BSP to run the calibration procedure each startup
   VLIB_DEFINE     += SFMA_USE_BSP_CALIB
```

Option (2)

```
   # allow your application to run the calibration procedure individually
   VLIB_DEFINE     += SFMA_USE_BSP_CALIB
```

Note: if the calibration procedure is enabled the start-up time will increase considerably, 50 - 200ms. This is due to the data erase time and write to serial flash memory.
The determination of the calibration value should therefore be a one-time action and not started on each system startup!
The execution of the calibration routine will need to write 4kb to each flash module, so take care of your flashed data accordingly!

10.3 Calibration of OCTA device

OCTA driver requires DQS delay value as argument of R_OCTA_Open. DQS delay value depends on the connected Octa Flash / Octa RAM memory, transfer mode (SPI, OPI, DOPI), board design and so on. It might be useful to run the calibration routines to determine the best value for the calibration.

The following Makefile / GHS.prj preprocessor flags are available to run the calibration

OCTA calibration options
Option (1)

```
   # allow the BSP to init the Octa Flash or Octa RAM
   VLIB_DEFINE     += BSP_INIT_OCTA_FLASH_R
     or
   VLIB_DEFINE     += BSP_INIT_OCTA_RAM_RW

   # allow the BSP to run the calibration procedure each startup
   VLIB_DEFINE     += OCTA_USE_BSP_CALIB
```

Option (2)

```
   # allow your application to run the calibration procedure individually
   VLIB_DEFINE     += OCTA_USE_BSP_CALIB
```

Note: if the calibration procedure is enabled the start-up time will increase considerably, 50 - 200ms. This is due to the data erase time and write to Octa Flash / Octa RAM memory.
The determination of the calibration value should be a one-time action and not started on each system startup!
The execution of the calibration routine will need to erase and program 4KByte to Octa Flash / Octa RAM, so take care of your flashed data accordingly!

Revision History	Renesas Graphics Library Porting Layer Guide

Rev.	Date	Description	
		Page	Summary
0.1	Nov 29, 2018	-	First edition.
0.2	Mar 28, 2019	8-79, 108, 118-122	Added the “const” to the argument.
		8-69, 96-99, 120, 121	Changed the return code to definition value.
		8, 10, 14, 17, 18, 20, 34, 38, 40, 42, 52, 77, 87, 117, 131, 132, 147,	Changed the default behavior.
		15	Added the customizing point of MaxPitch.
		51	Added the description about the window structure update timing.
		72, 88	Deleted the functions. R_WM_Sys_TryLockMsgQueue R_VDCE_Sys_VdceVersionGet
		82, 114	Added the Definition chapter
		90, 91	Added the parameter description.
		100, 102, 122, 124, 136, 137, 152, 153	Changed the status Get/Set specification.
		105, 106, 107, 108, 118	Changed the return code.
		118	Changed the parameter description.
		83, 142	Fixed typo.
1.0	June 12, 2019	19, 108, 127, 141,	Added the new function R_WM_Sys_InitGlobal R_VDCE_Sys_InitGlobal R_SPEA_SYS_InitGlobal R_VOWE_Sys_InitGlobal R_JCUA_Sys_InitGlobal
		33	Fixed the description of R_WM_Sys_ScreenGammaSet.
		99	Changed the default behavior of R_VDCE_Sys_ClockActEdgeSet.
		121	Changed the default behavior of R_SPEA_SYS_ErrorHandler
1.1	Oct 10, 2019	6, 7	Added the description for VOCA and DISCOM
		12	Added the handling event type of R_WM_Sys_DevEventRegister.
		21, 28, 29, 39~47, 67, 68, 79, 80, 87~92,	Added the new functions to WM porting layer. R_WM_Sys_IsrVocaErr R_WM_Sys_DevRootVocaSet R_WM_Sys_DevRootDiscomSet

			R_WM_Sys_ScreenVocaInit R_WM_Sys_ScreenVocaDeInit R_WM_Sys_ScreenVocaCreate R_WM_Sys_ScreenVocaDelete R_WM_Sys_ScreenVocaEnable R_WM_Sys_ScreenVocaExpImgSet R_WM_Sys_ScreenVocaClutSet R_WM_Sys_ScreenActMonEnable R_WM_Sys_ScreenVocaUpdate R_WM_Sys_WindowScaleSet R_WM_Sys_CaptureViewPortSet R_WM_Sys_CaptureExtVsyncSet R_WM_Sys_DiscomCreate R_WM_Sys_DiscomDelete R_WM_Sys_DiscomEnable R_WM_Sys_DiscomCrcSet R_WM_Sys_DiscomCrcGet R_WM_Sys_DiscomUpdate
		27	Changed the parameter name (Unit -> CapUnit) and meaning. R_WM_Sys_DevRootCaptureSet.
		66	Fixed the description for customizing points.
		185~198	Added the chapter for VOCA porting layer.
		199~211	Added the chapter for DISCOM porting layer.
1.2	Dec 03, 2019	8	Added the news file for DISCOM/VOCA/ECM control.
		9, 21, 29, 88~94	Added the DISCOM control to default behavior.
		21, 22, 28, 32, 38~48	Added the VOCA control to default behavior.
		9, 11	Added ECM sample driver control to default behavior.
		9, 11	Added ECM sample driver to the customize point.
		12, 13. 30, 32, 76, 78	Added the new VDCE interrupt control. Changed the callback set timing.
		50	Changed the definition name of virtual frame buffer address.
		70, 73	Added the parameter check of PosY and Height to default behavior.
		76	Added the new capture control to default behavior. Deleted the pin setting from default behavior.
1.3	Dec 25, 2019	30	Changed the default behavior of scan line setting.
		69, 107	Added the missing structure type. r_wm_ScaleChg_t
2.0	May 13, 2020	11, 90	Changed the default behavior of DISCOM interrupt disable timing. R_WM_Sys_DevDeinit R_WM_Sys_DiscomEnable
		21	Change the default behavior of control order. R_WM_Sys_IsrVocaErr
		23	Added the pending buffer update to default behavior. R_WM_Sys_DevFrameFinished
		32, 33	Changed the default behavior of interrupt control. R_WM_Sys_ScreenEnable

		38	Added the front porch check to default behavior. R_WM_Sys_ScreenVocaInit
		49, 53, 57, 59, 68	Added the layer lock to default behavior. R_WM_Sys_WindowSetFb R_WM_Sys_WindowCreate R_WM_Sys_WindowPosSet R_WM_Sys_WindowGeomSet R_WM_Sys_WindowFlagsUpdate R_WM_Sys_WindowScaleSet
		59	Added the scaling parameter setting to default behavior. R_WM_Sys_WindowGeomSet
		76	Added the SYNC_ONLY option control to default behavior R_WM_Sys_CaptureCreate
		101, 102	Changed the specification of control unit and caller functions. R_WM_Sys_LockDevice R_WM_Sys_UnlockDevice
		107	Added the enum value R_WM_SCALE_CHANGE_WIN_SIZE
		137, 138	Changed the caller functions and default behavior. R_VDCE_Sys_AllLock R_VDCE_Sys_AllUnlock
		215-218	Added the Appendix chapter.

Renesas Graphics Library
Porting Layer Guide

Publication Date:	Rev.0.1	Nov 29, 2018
	Rev.2.0	May 13, 2020

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/

Renesas Graphics Library
Porting Layer Guide

Renesas Electronics Corporation