User's Manual

RENESAS

CONFIDENTIAL

RH850/D1x Device Family

Renesas Graphics Library Window Manager (WM) Driver

User's Manual: Software

Renesas Electronics
www.renesas.com

Rev. 2.0 May 2020

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

How to Use This Manual

1. Purpose and Target Readers

This manual is designed to provide the user with an understanding the functions of WM. This manual is written for engineers who use WM.

> 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 WM 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 (This manual)
	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

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
bpp	bit per pixel
CLUT	Color Look Up table
CPU	Central Processing Unit. The microprocessor core of the LSI.
Device	S/W abstraction of the H/W.
DISCOM	Display Output Comparator. This is H/W, which calculates CRC.
ECM	Error Control Module. This is H/W, which handles error interrupt.
FB	Frame buffer
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 rendering activities.
GPU	Graphics Processing Unit. This is H/W which controls 2D graphics rendering.
H/W	Hardware
Layer	H/W concept of the stackable visual area on the display
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 frame buffer 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, which is abstraction of the attached physical display
SPEA	Sprite Engine A. This is H/W, which controls Sprite data and RLE decoding.
Sprite	A graphical entity which can be moved on the screen independently.
Surface	A concrete (i.e. physical) implementation of the window's area
TARGA	Truevision Graphics Adapter
VBLANK	Vertical blanking signal.
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.
Window	A S/W 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 Feature and Scope

The window manager is a driver stack, which enables an abstract access to the device's video input and video output hardware. The abstraction serves the purpose of relieving an application developer from using low-level drivers, as well as enabling code portability because the same API can be used on different hardware.
The window manager itself is a kind of middleware driver. That means that the driver itself is not closely linked to a specific hardware macro. It defines functionality, which is achieved by implementation of hardware independent parts in the generic code and by calling other hardware drivers (e.g. VIN/VOUT).

1.2 Component Structure

The component structure of WM is shown in *Figure 1-1.*

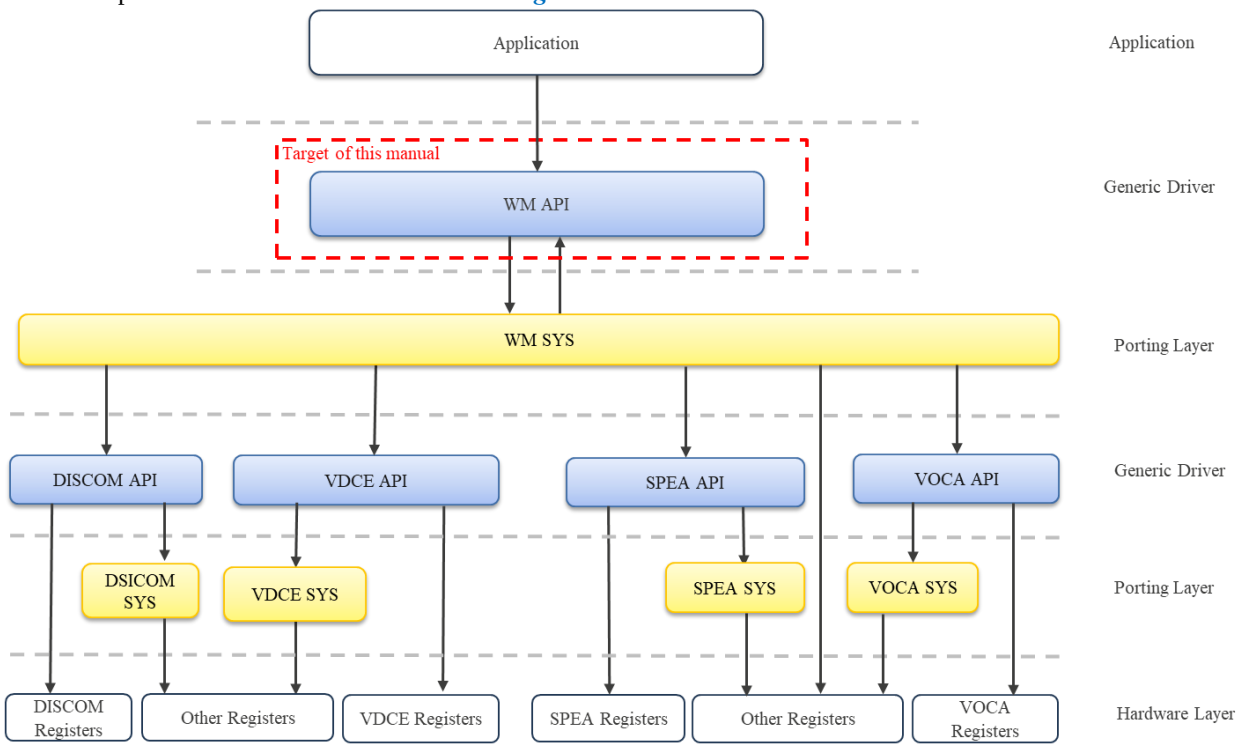


Figure 1-1 Component Structure

For the details of the API, please refer to *Chapter 4.*

2.Basic Specification

2.1 Summary Specification

The summary of specification is described in *Table 2-1.*

Table 2-1 Summary Specification

Items	Description
Target LSI	RH850/D1L2(H), RH850/D1M1(H), RH850/D1M1-V2, RH850/D1M1A, RH850/D1M2(H)
Main Feature	• Handle abstract video-output “windows” (creation/deletion, enable/disable, swap buffer, etc.) • Handle abstract video-input “capture windows” (creation/deletion, enable/disable, pause, etc.) • Handle Sprites (buffer setting, enable/disable, animate, etc.) • Control and setup a display with the display database (DDB) • Control the V-sync interrupt handling and buffer switching (double/triple buffering) • Creating Window (Max. 4 windows) and Window configuration (Frame Buffer/Sprite, Color format, etc.) • Moving Window and Moving Sprite inside Window • Management of Frame buffer (Single buffer, double buffer, etc...) • Control of synchronous update by queue mechanism • Handle DSICOM/VOCA (creation/deletion, enable/disable, set expected value, etc.)
Semaphore / Mutex	N/A. This can be implemented with the porting layer.
Interrupts	Used in the WM driver. For more details please see *section 2.3.*

2.2 Reserved Word

WM uses the following prefixes for avoiding confusion from other software. Prefixes of WM is described in *Table 2-2.*

Table 2-2 Prefixes

Prefix	Description
R_WM_*	Prefix for WM Module
r_wm_*	

2.3 Interrupt Handler List

WM uses the VDCE interrupt. See VDCE User's manual for the detail.

Table 2-3 Interrupt Handler List

No.	Interrupt Name	Interrupt Handler Name	Description
VDCE Unit 0			
(1)	INTVDCE0S0LOVSYNC	R_VDCE_Isr	V-sync output at Layer 0 interrupt.
(2)	INTVDCE0GR3VLINE	R_VDCE_Isr	Scanline detection of designated line at Graphics 3 interrupt.
(3)	INTVDCE0S0VIVSYNC	R_VDCE_Isr	V-sync input at Layer 0 interrupt.
(4)	INTVDCE0ERR	R_VDCE_IsrError	Error interrupt.
(5)	INTVDCE0S1LOVSYNC	R_VDCE_Isr	V-sync output at Layer 1 interrupt.
(6)	INTVDCE0OIRLOVSYNC	R_VDCE_Isr	V-sync output at Output Image Render interrupt.
(7)	INTVDCE0OIRVLINE	R_VDCE_Isr	Scan Line detection of designated line at Output Image Render Interrupt.
VDCE Unit 1			
(8)	INTVDCE1S0LOVSYNC	R_VDCE_Isr	V-sync output at Layer 0 interrupt.
(9)	INTVDCE1GR3VLINE	R_VDCE_Isr	Scanline detection of designated line at Graphics 3 interrupt.
(10)	INTVDCE1S0VIVSYNC	R_VDCE_Isr	V-sync input at Layer 0 interrupt.
(11)	INTVDCE1ERR	R_VDCE_IsrError	Error interrupt.
(12)	INTVDCE1S1LOVSYNC	R_VDCE_Isr	V-sync output at Layer 1 interrupt.

WM uses the interrupt handling via ECM.
Interrupt handler exists in WM Porting layer. See Porting Layer Guide for the detail.

Table 2-4 Interrupt Handler List via ECM

No.	Interrupt Name	Interrupt Handler Name	Description
(1)	INTVOCAERR	R_WM_Sys_IsrVocaErr	Logical OR combination of VOCA and DISCOM error signals.

2.4 Error Handling

2.4.1 Return code

WM driver has 5 types of error codes.
The error code generated by each API synchronously is notified by both return code and error callback.
The error code generated asynchronously is notified by only error callback.
The error callback is installed by R_WM_ErrorCallbackSet.

2.4.1.1 Parameter level

Following errors occur by a cause such as abnormality of parameter. In this case, please set valid parameter again.

- R_WM_ERR_INVALID_WM_UNIT
- R_WM_ERR_PARAM_INCORRECT
- R_WM_ERR_NULL_PTR_ARGUMENT
- R_WM_ERR_RANGE_WM
- R_WM_ERR_COLORFMT
- R_WM_ERR_NOT_CLUT_WIN_FMT
- R_WM_ERR_NOT_FB_WINDOW
- R_WM_ERR_NOT_SPRITE_WINDOW
- R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW

2.4.1.2 Timing level

Following errors occur by a cause such as abnormality of execution timing. In this case, please call again after changing to valid state or timing.

- R_WM_ERR_NOT_UNINITIALIZED
- R_WM_ERR_NOT_INITIALIZED
- R_WM_ERR_NOT_DELETED
- R_WM_ERR_NOT_DISABLED
- R_WM_ERR_WIN_SWAP_FAILED
- R_WM_ERR_SCREEN_TIMING_NOT_SET
- R_WM_ERR_NO_PHYS_WINDOW
- R_WM_ERR_SPRITE_NOT_FOUND
- R_WM_ERR_CAPTURE_UNIT_COUNT_EXCEEDED

2.4.1.3 Porting layer level

Following errors occur when unexpected error occurs in porting layer. In this case, please reset the RH850/D1x device.

- R_WM_ERR_NG
- R_WM_ERR_SYS_LAYER_INIT_FAILURE
- R_WM_ERR_DEV_DEINIT_FAILED
- R_WM_ERR_DISPLAY_TIMING_SET
- R_WM_ERR_DISPLAY_OUTPUT_FORMAT_SET
- R_WM_ERR_COULD_NOT_SET_SCREEN_BG_COLOR
- R_WM_ERR_COULD_NOT_SET_SCREEN_COLOR_CURVE
- R_WM_ERR_COULD_NOT_SET_SCREEN_GAMMA
- R_WM_ERR_COULD_NOT_ENABLE_SCREEN
- R_WM_ERR_COULD_NOT_DISABLE_SCREEN
- R_WM_ERR_SYS_WIN_CREATE_FAILED
- R_WM_ERR_SYS_WIN_DELETE_FAILED
- R_WM_ERR_SYS_WIN_MOVE_FAILED
- R_WM_ERR_SYS_WIN_GEOMETRY_SET_FAILED
- R_WM_ERR_SYS_WIN_ALPHA_SET_FAILED
- R_WM_ERR_SYS_WIN_FLAG_UPDATE_FAILED
- R_WM_ERR_SYS_WIN_SWAP_FAILED
- R_WM_ERR_SYS_WIN_EXTERNAL_BUF_SET_FAILED
- R_WM_ERR_SYS_WIN_DELETE_ALL_SPRITES_FAILED
- R_WM_ERR_SYS_CAPTURE_CREATE_FAILED

- R_WM_ERR_SYS_CAPTURE_DELETE_FAILED
- R_WM_ERR_SYS_CAPTURE_ENABLE_FAILED
- R_WM_ERR_SPEA_INTERNAL
- R_WM_ERR_VOUT_INTERNAL
- R_WM_ERR_SYS_WIN_SCALED_SET_FAILED
- R_WM_ERR_SYS_CAPTURE_MOVE_FAILED
- R_WM_ERR_SYS_CAPTURE_RESIZE_FAILED
- R_WM_ERR_SYS_CAPTURE_SET_VSYNC_FAILED
- R_WM_ERR_SYS_VOCA_CREATE_FAILED
- R_WM_ERR_SYS_VOCA_DELETE_FAILED
- R_WM_ERR_SYS_VOCA_ENABLE_FAILED
- R_WM_ERR_SYS_VOCA_SET_FAILED
- R_WM_ERR_SYS_ACT_MON_FAILED
- R_WM_ERR_SYS_DISCOM_CREATE_FAILED
- R_WM_ERR_SYS_DISCOM_DELETE_FAILED
- R_WM_ERR_SYS_DISCOM_ENABLE_FAILED
- R_WM_ERR_SYS_DISCOM_SET_FAILED
- R_WM_ERR_SYS_DISCOM_GET_FAILED
- R_WM_ERR_VOCA_NOT_FOUND
- R_WM_ERR_DISCOM_NOT_FOUND
- R_WM_ERR_VOCA_INTERNAL
- R_WM_ERR_DISCOM_INTERNAL

2.4.1.4 System level

Following errors occur by a cause such as OS dependent error (e.g. system call error, resource shortage). In this case, please do recovery processing from a system layer, because this status cannot be restored only in this library.

- R_WM_ERR_MALLOC_FAILED
- R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE
- R_WM_ERR_EVENT_FAILED
- R_WM_ERR_FREE_FAILED

2.4.1.5 Device level

Following errors occur when the function is not supported with target device. In this case, please skip the function call.

- R_WM_ERR_NOT_SUPPORTED

2.4.2 Callback

H/W error event is notified by event callback installed with R_WM_DevInit.

2.4.2.1 VDCE error event

These events occur when H/W processing is not in time (e.g. Input or Output pixel clock is too high.).

- 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

Even under normal conditions, these events may occur immediately after R_WM_WindowEnable, R_WM_WindowDisable, R_WM_CaptureEnable and R_WM_CaptureDisable.
If it does not occur regularly during normal operation, there is a high possibility that there is no problem. In this case, please ignore the event.
If it occurs regularly during normal operation, video output signal will be noisy. In this case, please reset the RH850/D1x device.

2.4.2.2 VOCA/DISCOM error event

These events occur when output image or timing is not match with expected value. Error handling is left to the user's discretion.

- R_WM_EVENT_DISCOM_MISMACTH
- R_WM_EVENT_VOCA_MISMACTH
- R_WM_EVENT_ACT_MON_ERROR

R_WM_EVENT_DISCOM_MISMACTH and R_WM_EVENT_VOCA_MISMACTH may occur due to deviation of the update timing of the expected value.
Anomaly judgment might be triggered by being notified several times continuously at Vsync intervals.

2.5 State Transition

2.5.1 WM Unit State Transition

Each WM unit has following status.

Table 2-5 WM unit State Details

No.	State Name	Description
(1)	Uninitialized	WM unit is not initialized.
(2)	Initialized	WM unit is initialized.
(3)	Display Initialized	Display is initialized.
(4)	Display Active	Display is activated

The image describes state transition.

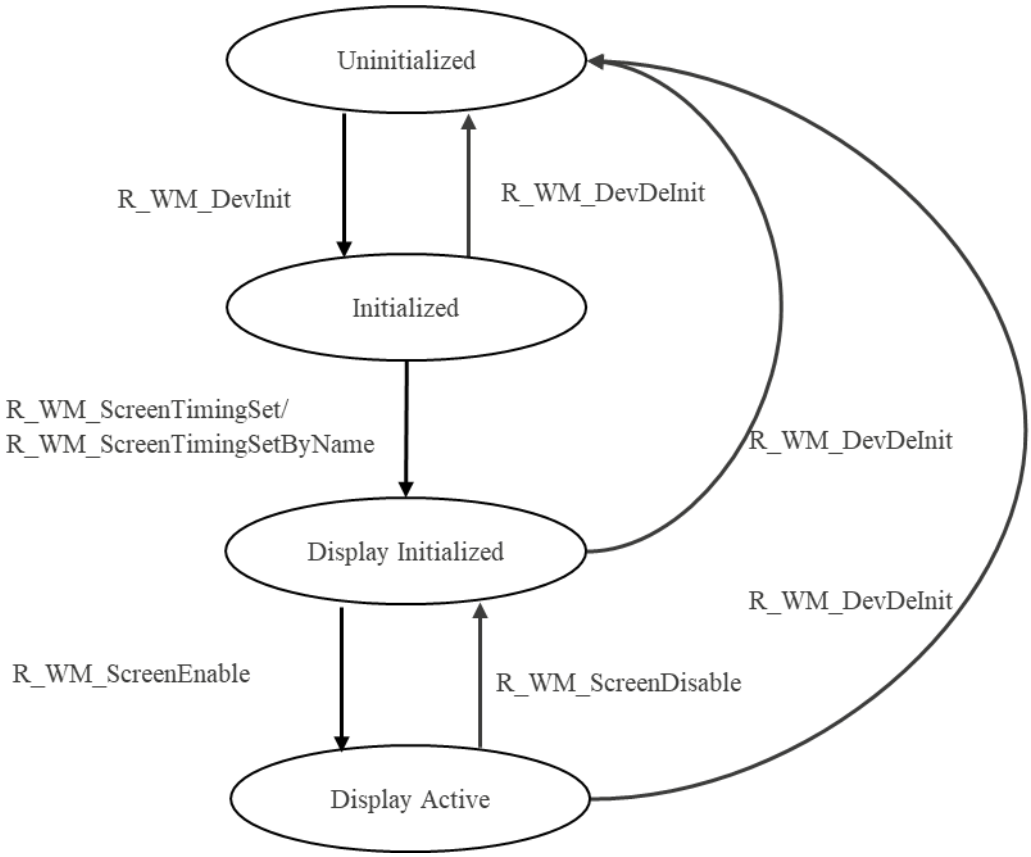


Figure 2-1 State Transition Diagram of WM driver

Table 2-6 State Transition Table of WM unit

Function Name	State			
	Uninitialized	Initialized	Display Initialized	Display Active
R_WM_DevInit	OK	NG	NG	NG
R_WM_DevEventRegister	NG	OK	OK	OK
R_WM_DevDeinit	OK	OK	OK	OK
R_WM_DevInfoGet	OK	OK	OK	OK
R_WM_GetVersionString	OK	OK	OK	OK
R_WM_ScreenTimingSet	NG	OK	OK	NG
R_WM_ScreenTimingSetByName	NG	OK	OK	NG
R_WM_ScreenColorFormatSet	NG	OK	OK	NG
R_WM_ScreenBgColorSet	NG	OK	OK	OK
R_WM_ScreenColorCurveSet	NG	OK	OK	OK
R_WM_ScreenGammaSet	NG	OK	OK	OK
R_WM_ScreenEnable	NG	NG	OK	OK
R_WM_ScreenDisable	NG	OK	OK	OK
R_WM_ScreenVocaInit	NG	NG	NG	OK
R_WM_ScreenVocaDeInit	NG	NG	NG	OK
R_WM_ScreenVocaCreate	NG	NG	NG	OK
R_WM_ScreenVocaDelete	NG	NG	NG	OK
R_WM_ScreenVocaEnable	NG	NG	NG	OK
R_WM_ScreenVocaDisable	NG	NG	NG	OK
R_WM_ScreenVocaExpImgSet	NG	NG	NG	OK
R_WM_ScreenVocaClutSet	NG	NG	NG	OK
R_WM_ScreenActivityMonEnable	NG	NG	NG	OK
R_WM_ScreenActivityMonDisable	NG	NG	NG	OK
R_WM_WindowCapabilitiesSet	OK *1	NG	NG	NG
R_WM_WindowCreate	NG	OK	OK	OK

R_WM_WindowDelete	NG	OK	OK	OK
R_WM_WindowEnable	NG	NG	NG	OK
R_WM_WindowDisable	NG	OK	OK	OK
R_WM_WindowMove	NG	OK	OK	OK
R_WM_WindowResize	NG	OK	OK	OK
R_WM_WindowColorFmtSet	NG	OK	OK	OK
R_WM_WindowAlphaSet	NG	OK	OK	OK
R_WM_WindowPremultipliedAlphaEnable	NG	OK	OK	OK
R_WM_WindowPremultipliedAlphaDisable	NG	OK	OK	OK
R_WM_WindowVerticalMirrorEnable	NG	OK	OK	OK
R_WM_WindowVerticalMirrorDisable	NG	OK	OK	OK
R_WM_WindowSwap	NG	OK	OK	OK
R_WM_WindowNewDrawBufGet	NG	OK	OK	OK
R_WM_WindowVisibleBufGet	OK	OK	OK	OK
R_WM_WindowCurrentDrawBufGet	OK	OK	OK	OK
R_WM_WindowExternalBufSet	NG	OK	OK	OK
R_WM_WindowColorKeyEnable	NG	OK	OK	OK
R_WM_WindowColorKeyDisable	NG	OK	OK	OK
R_WM_WindowClutSet	NG	OK	OK	OK
R_WM_WindowDeleteAllSprites	NG	OK	OK	OK
R_WM_WindowScaledSizeSet	NG	OK	OK	OK
R_WM_FrameEndMark	NG	OK	OK	OK
R_WM_FrameWait	NG	OK	OK	OK
R_WM_FrameExecuteNext	NG	OK	OK	OK
R_WM_FrameExecuteVoca	NG	OK	OK	OK
R_WM_FrameExecuteDiscom	NG	OK	OK	OK
R_WM_CaptureCreate	NG	OK	OK	OK
R_WM_CaptureDelete	NG	OK	OK	OK
R_WM_CaptureEnable	NG	NG	NG	OK

R_WM_CaptureDisable	NG	OK	OK	OK
R_WM_Cap_CapBufGet	OK	OK	OK	OK
R_WM_Cap_DispBufGet	OK	OK	OK	OK
R_WM_CaptureMove	NG	OK	OK	OK
R_WM_CaptureResize	NG	OK	OK	OK
R_WM_CaptureScaledSizeSet	NG	OK	OK	OK
R_WM_CaptureExtVsyncSet	NG	OK	OK	OK
R_WM_SpriteCreate	NG	OK	OK	OK
R_WM_SpriteEnable	NG	OK	OK	OK
R_WM_SpriteDisable	NG	OK	OK	OK
R_WM_SpriteMove	NG	OK	OK	OK
R_WM_SpriteBufSet	NG	OK	OK	OK
R_WM_SpriteDelete	NG	OK	OK	OK
R_WM_ErrorCallbackSet	NG	OK	OK	OK
R_WM_ErrorHandler	OK	OK	OK	OK
R_WM_ColorFmtBitsPerPixGet	OK	OK	OK	OK
R_WM_DiscomCreate	NG	OK	OK	OK
R_WM_DiscomDelete	NG	NG	NG	OK
R_WM_DiscomEnable	NG	NG	NG	OK
R_WM_DiscomDisable	NG	NG	NG	OK
R_WM_DiscomCrcSet	NG	OK	OK	OK
R_WM_DiscomCrcGet	NG	OK	OK	OK

*1: It can execute when all WM units are uninitialized status.

2.5.2 Window State Transition

Each Window has following status.

Table 2-7 Window State Details

No.	State Name	Description
(1)	Not Initialized	Window is not initialized.
(2)	Disabled	Window is disabled.
(3)	Enabled	Window is enabled.

The image describes state transition.

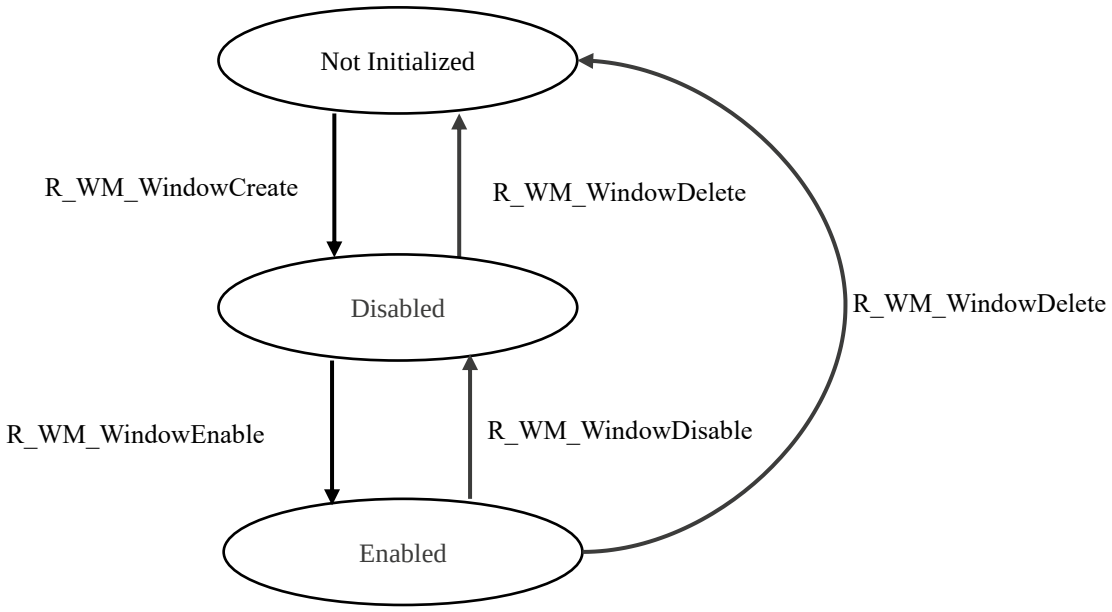


Figure 2-2 State Transition Diagram of WM driver

Table 2-8 State Transition Table of Window functions

Function Name	State		
	Not Initialized	Disabled	Enabled
R_WM_WindowCreate	OK	NG	NG
R_WM_WindowDelete	NG	OK	OK
R_WM_WindowEnable	NG	OK	OK
R_WM_WindowDisable	NG	OK	OK
R_WM_WindowMove	NG	OK	OK
R_WM_WindowResize	NG	OK	OK
R_WM_WindowColorFmtSet	NG	OK	OK
R_WM_WindowAlphaSet	NG	OK	OK
R_WM_WindowPremultipliedAlphaEnable	NG	OK	OK
R_WM_WindowPremultipliedAlphaDisable	NG	OK	OK
R_WM_WindowVerticalMirrorEnable	NG	OK	OK
R_WM_WindowVerticalMirrorDisable	NG	OK	OK
R_WM_WindowSwap	NG	OK	OK
R_WM_WindowNewDrawBufGet	OK	OK	OK
R_WM_WindowVisibleBufGet	OK	OK	OK
R_WM_WindowCurrentDrawBufGet	OK	OK	OK
R_WM_WindowExternalBufSet	NG	OK	OK
R_WM_WindowColorKeyEnable	NG	OK	OK
R_WM_WindowColorKeyDisable	NG	OK	OK
R_WM_WindowClutSet	NG	OK	OK
R_WM_WindowDeleteAllSprites	NG	OK	OK
R_WM_WindowScaledSizeSet	NG	OK	OK

2.5.3 Sprite data State Transition

Each Sprite data has following status.

Table 2-9 Sprite data State Details

No.	State Name	Description
(1)	Not Initialized	Sprite data is not initialized.
(2)	Disabled	Sprite data is disabled.
(3)	Enabled	Sprite data is enabled.

The image describes state transition.

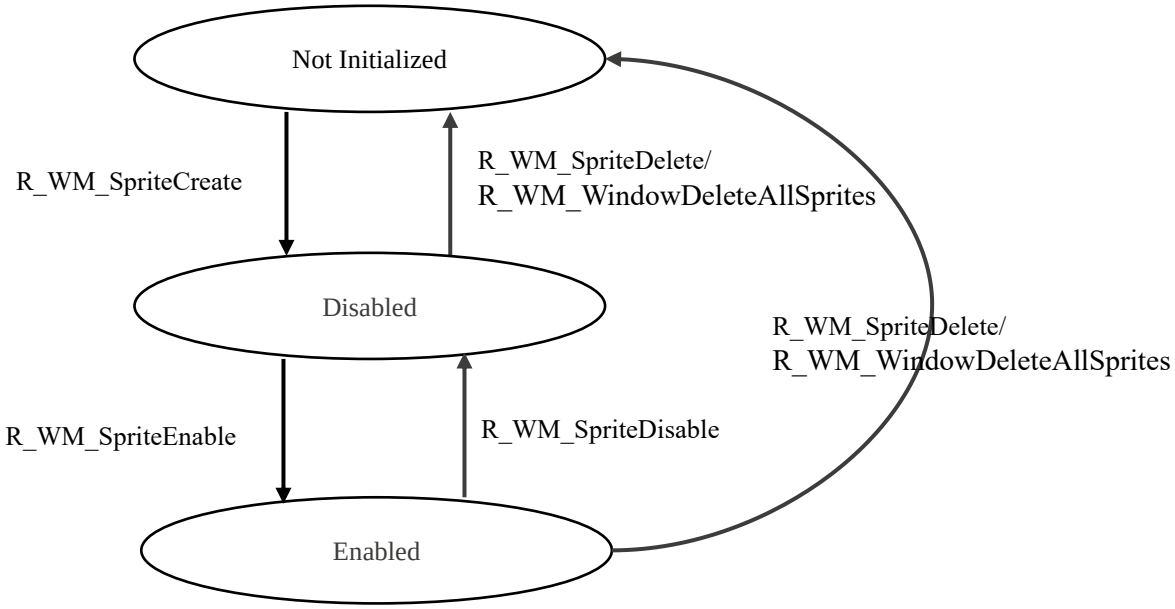


Figure 2-3 State Transition Diagram of Sprite function

Table 2-10 State Transition Table of Sprite functions

Function Name	State		
	Not Initialized	Disabled	Enabled
R_WM_SpriteCreate	OK	NG	NG
R_WM_SpriteEnable	NG	OK	OK
R_WM_SpriteDisable	NG	OK	OK
R_WM_SpriteMove	NG	OK	OK
R_WM_SpriteBufSet	NG	OK	OK
R_WM_SpriteDelete	NG	OK	OK
R_WM_WindowDeleteAllSprites	OK	OK	OK

2.5.4 Capture surface State Transition

Each Capture surface has following status.

Table 2-11 Capture surface State Details

No.	State Name	Description
(1)	Not Initialized	Capture surface is not initialized.
(2)	Disabled	Capture surface is disabled.
(3)	Enabled	Capture surface is enabled.

The image describes state transition.

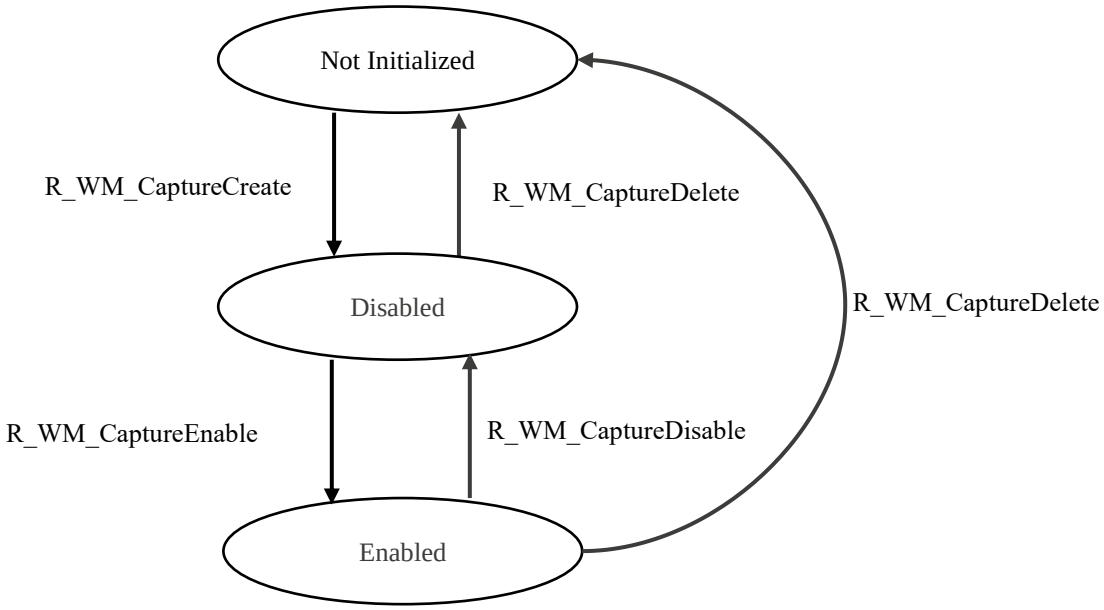


Figure 2-4 State Transition Diagram of Capture function

Table 2-12 State Transition Table of Capture functions

Function Name	State		
	Not Initialized	Disabled	Enabled
R_WM_CaptureCreate	OK	NG	NG
R_WM_CaptureDelete	NG	OK	OK
R_WM_CaptureEnable	NG	OK	OK
R_WM_CaptureDisable	NG	OK	OK
R_WM_Cap_CapBufGet	OK	OK	OK
R_WM_Cap_DispBufGet	OK	OK	OK
R_WM_CaptureMove	NG	OK	OK
R_WM_CaptureResize	NG	OK	OK
R_WM_CaptureScaledSizeSet	NG	OK	OK
R_WM_CaptureExtVsyncSet	NG	OK	NG

2.5.5 VOCA monitor area State Transition

Each VOCA monitor area has following status.

Table 2-13 VOCA monitor area State Details

No.	State Name	Description
(1)	Not Initialized	VOCA H/W is not initialized.
(2)	Initialized	VOCA H/W is initialized. VOCA monitor area is not created.
(3)	Disabled	VOCA monitor area is disabled.
(4)	Enabled	VOCA monitor area is enabled.

The image describes state transition.

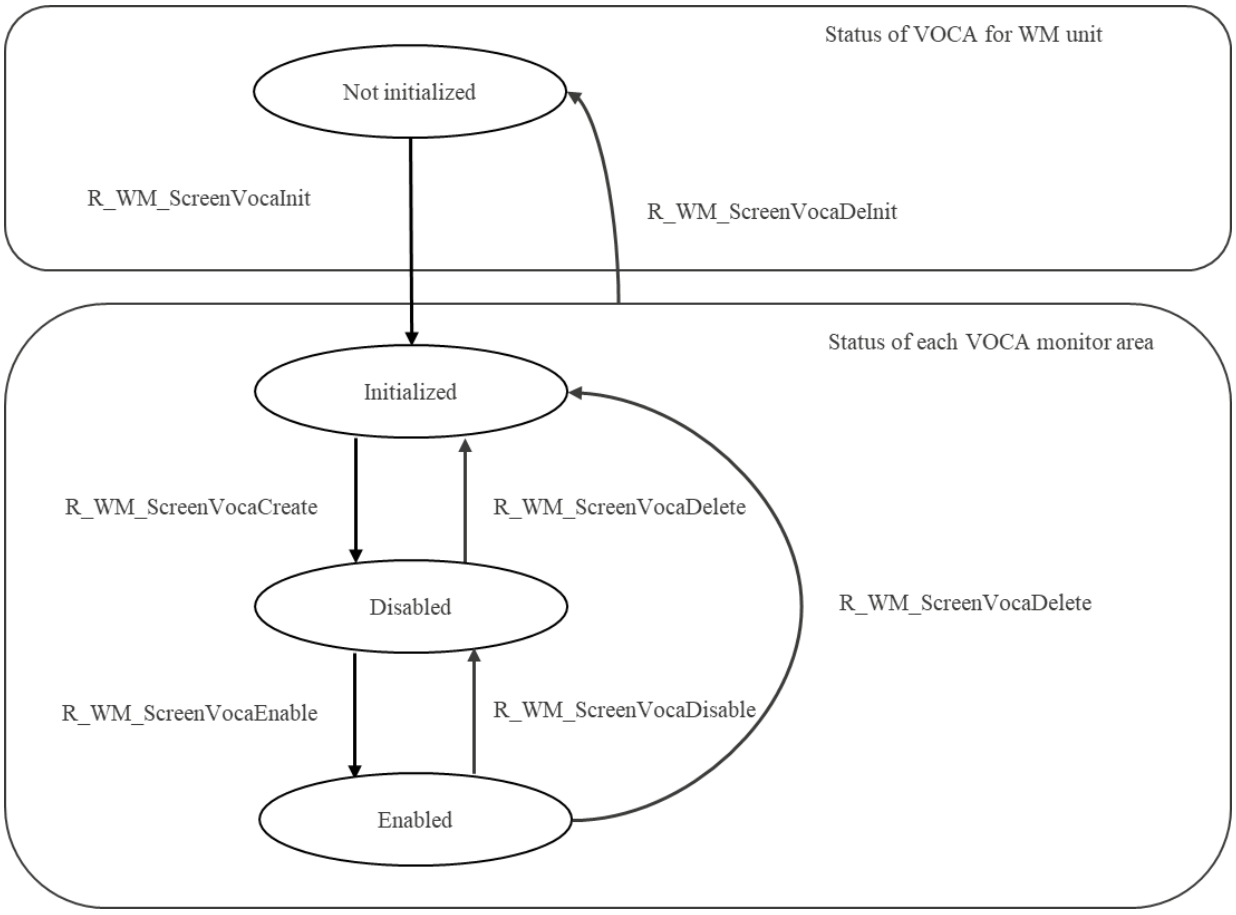


Figure 2-5 State Transition Diagram of VOCA function

Table 2-14 State Transition Table of VOCA functions

Function Name	State			
	Not Initialized	Initialized	Disabled	Enabled
R_WM_ScreenVocaInit	OK	NG	NG	NG
R_WM_ScreenVocaDeInit	OK	OK	OK	OK
R_WM_ScreenVocaCreate	NG	OK	NG	NG
R_WM_ScreenVocaDelete	NG	NG	OK	OK
R_WM_ScreenVocaEnable	NG	NG	OK	OK
R_WM_ScreenVocaDisable	NG	NG	OK	OK
R_WM_ScreenVocaExpImgSet	NG	NG	OK	OK
R_WM_ScreenVocaClutSet	NG	NG	OK	OK
R_WM_ScreenActivityMonEnable	NG	OK	OK	OK
R_WM_ScreenActivityMonDisable	NG	OK	OK	OK

2.5.6 DISCOM device State Transition

Each Discom device has following status.

Table 2-15 Discom device State Details

No.	State Name	Description
(1)	Not Initialized	Discom device is not initialized.
(2)	Disabled	Discom device is disabled.
(3)	Enabled	Discom device is enabled.

The image describes state transition.

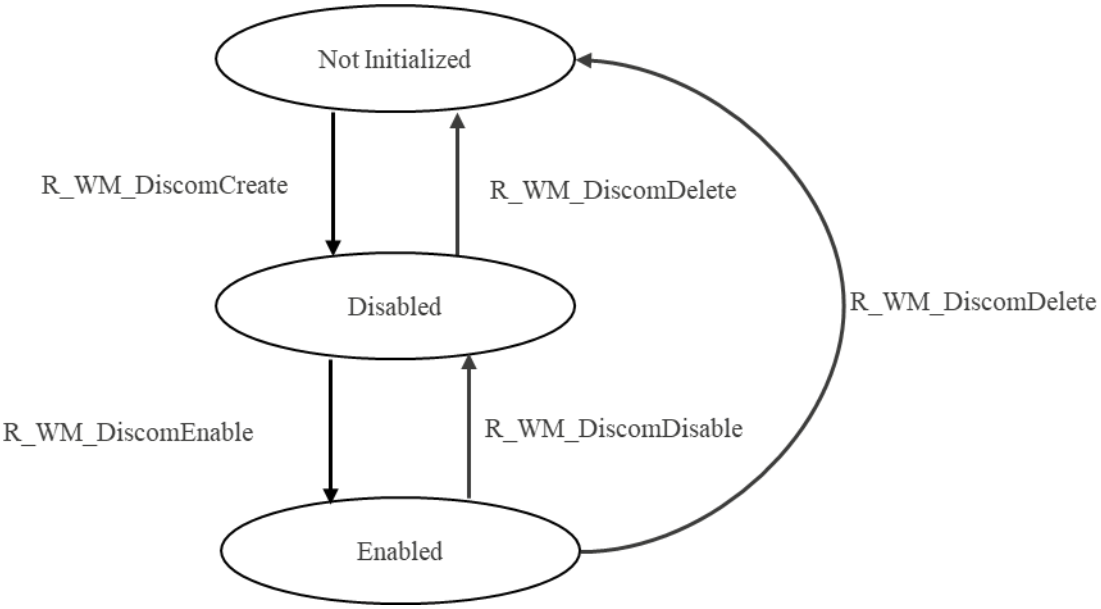


Figure 2-6 State Transition Diagram of Discom function

Table 2-16 State Transition Table of Discom functions

Function Name	State		
	Not Initialized	Disabled	Enabled
R_WM_DiscomCreate	OK	NG	NG
R_WM_DiscomDelete	NG	OK	OK
R_WM_DiscomEnable	NG	OK	OK
R_WM_DiscomDisable	NG	OK	OK
R_WM_DiscomCrcSet	NG	OK	OK
R_WM_DiscomCrcGet	OK	OK	OK

3.Function Description

3.1 Fundamental Concepts

3.1.1 Device

A device encapsulates the framework’s internal state regarding one WM unit. In case of more than one WM unit present, there should exist a separate device for each of these units intended to be used. The application layer can address one or more devices by the Unit parameter, which is available for all device dependent API calls.

WM unit is same as VDCE unit. It can take values 0 or 1, depending on the number of VDCE units available on the RH850/D1x device.

3.1.2 Window

A window is a designated, independently managed rectangular portion of the screen. It is a way of managing access to a shared global resource – the physical display. Drawing to a physical display can only be realized through drawing to a window, which means filling its framebuffer with meaningful data, normally by the help of GPU. The framework takes care of windows through the user-maintained r_wm_Window_t structure instances.

A window can be mapped to a hardware layer. There are 4 layers in WM Unit.

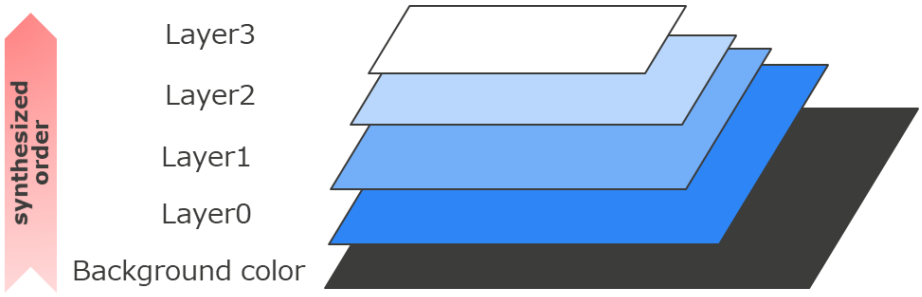


Figure 3-1 Layer order

There are three distinct types of windows.

- Frame buffer window
  Framebuffer window's content is defined by the content of its currently visible framebuffer.

- RLE window
  RLE window uses the Run Length Encoding feature of the SPEA H/W. This means that the FB pixels are specified according to TARGA RLE format description. Only the image definition part of the format is supported, the information found in the header of the TARGA file is anyway specified with the other window parameters. The RLE format is a great choice for images with larger flat-colored sections.

- Sprite window
  Sprite window's content is defined by the content of the Sprites attached to it, clipped by the window's geometry.

Following figure is example of window assignment and relationship with VDCE and SPEA blocks.
For the detail of VDCE and SPEA, see User's manual of each driver.

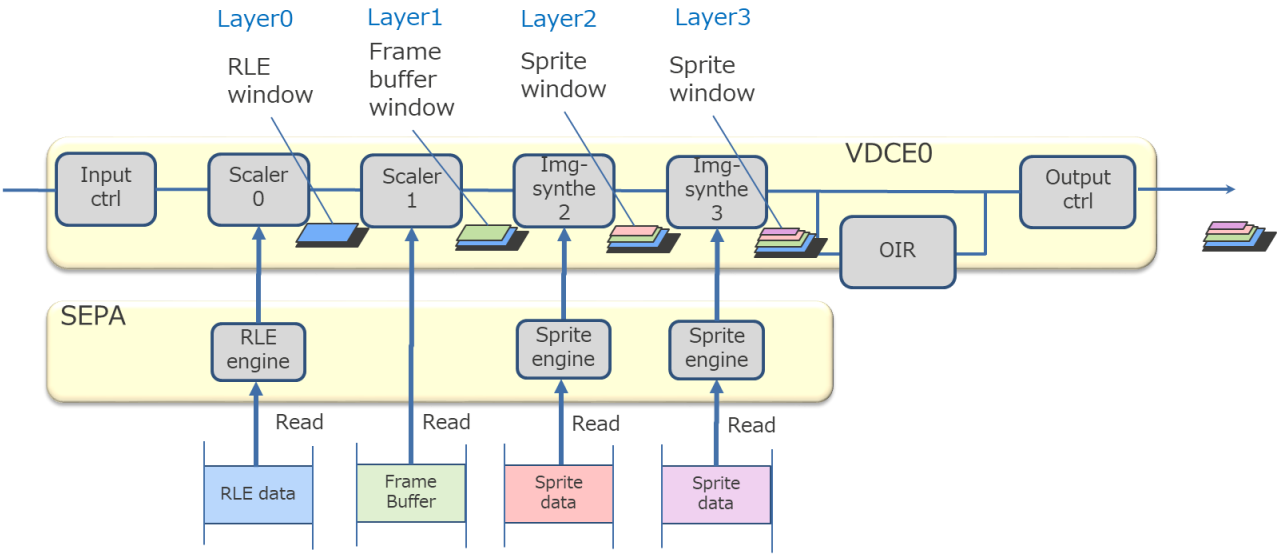


Figure 3-2 Window allocation example

3.1.3 RLE data

When user prepares the RLE compressed data, it needs an additional dummy border. Dummy border requires more than 128 bytes. And, total width of display area and dummy border requires 128 byte-aligned.

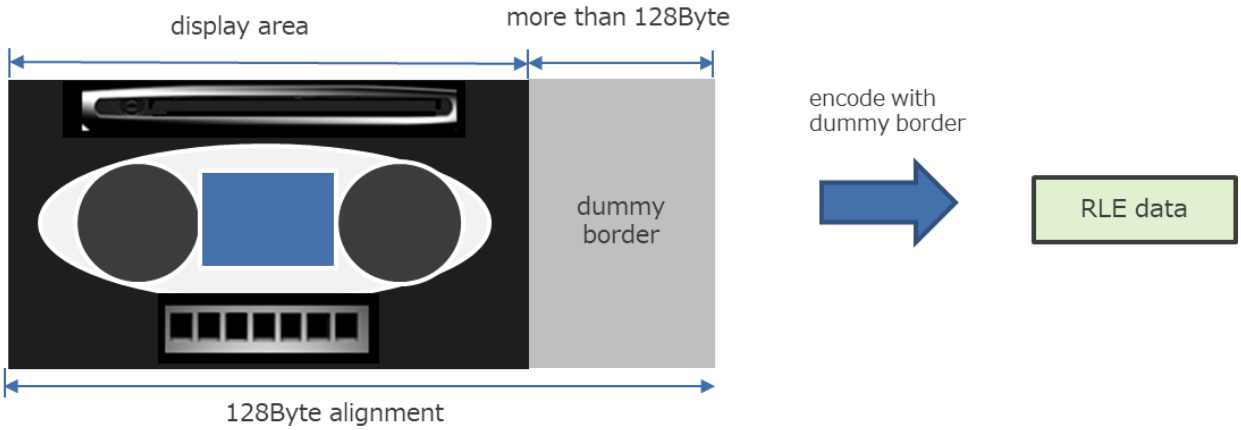


Figure 3-3 Dummy border

For example, width of display area is 720 [pixel] and decoded image is 32 [bpp] then;
total_width = ((((720 * 4) + 128 + (128 - 1)) / 128) * 128) / 4 = 768 [pixel]
In this case, 48 pixels dummy boarder should be added to image data before RLE data is created.

3.1.4 Sprite data

A Sprite is an image in the memory registered with the H/W (Sprite engine).
16 Sprite data can be displayed to Sprites-hosting windows.

When two or more Sprite data areas overlap, the one with the larger PosZ takes precedence. Note that alpha blending between sprite data is not effective within one sprite window.

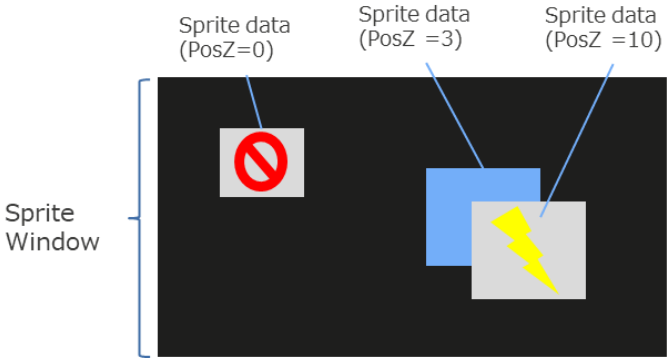


Figure 3-4 Sprite data

3.1.5 Capture surface

A capture surface can be created for one video input source.

VDCE has Input controller for receiving external video input. Scaler 0 has the feature to write the received video data to the capture buffer. The capture buffer can be synthesized by inputting it to Scaler 0 or Scaler 1 as a frame buffer window.
Video input captured data with VDCE0 can be synthesized by VDCE0 Scaler 0 or VDCE1 Scaler1.
Video input captured data with VDCE1 can be synthesized by VDCE1 Scaler 0 or VDCE0 Scaler1.

VDCE units that support the capture function are device dependent. See *Table 3-42* for the detail.
Following figure is RH850/D1M2H data flow that supports the most units.

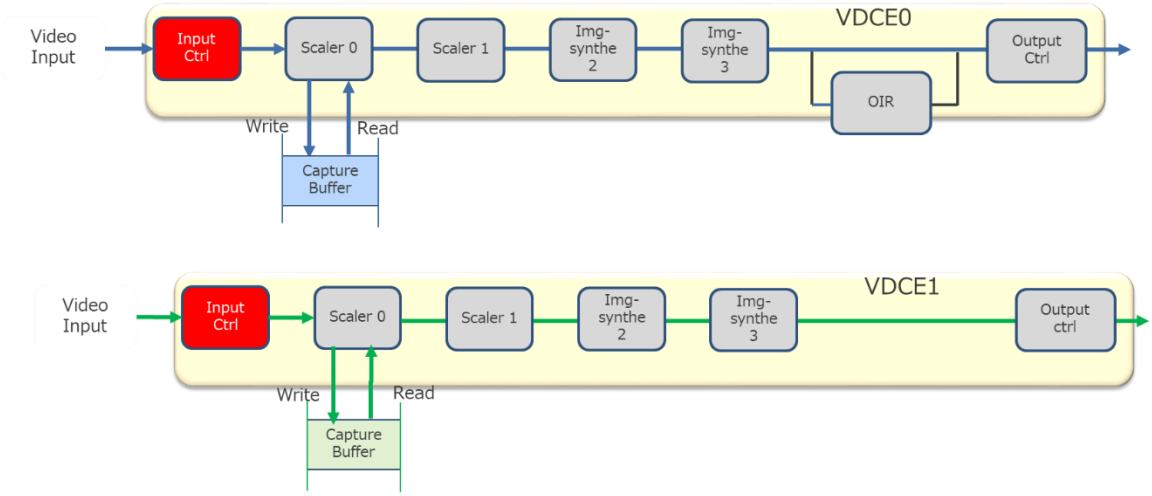


Figure 3-5 Capture data flow (1) for RH850/D1M2H

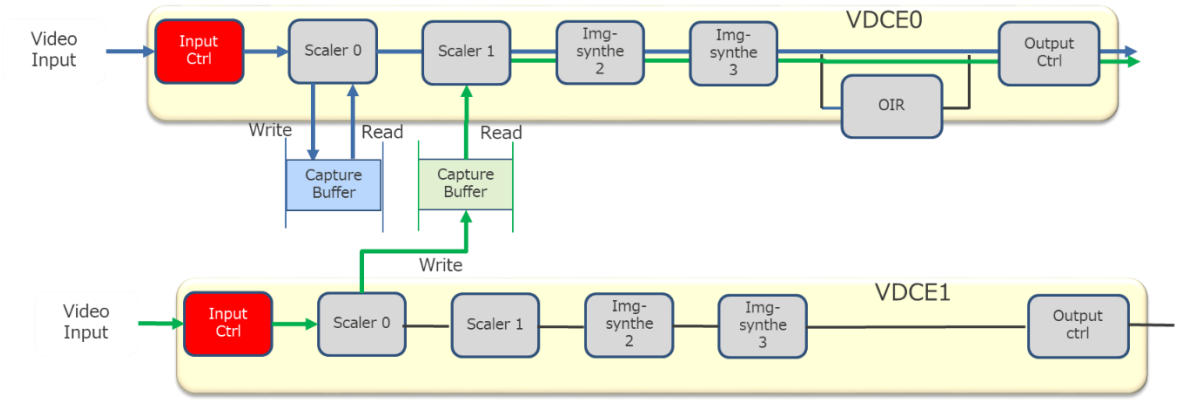


Figure 3-6 Capture data flow (2) for RH850/D1M2H

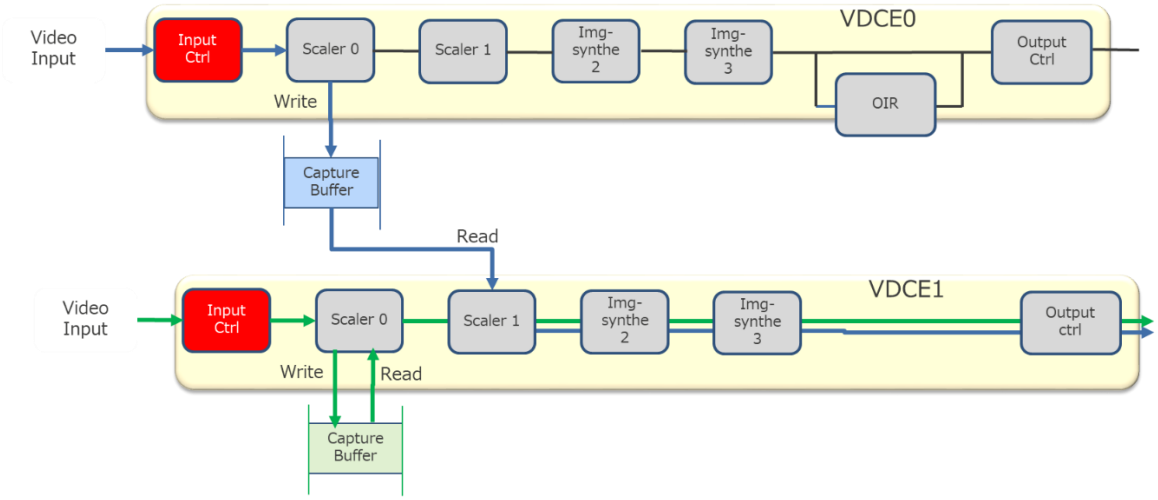


Figure 3-7 Capture data flow (3) for RH850/D1M2H

3.1.6 Message Queue

Most of the WM functions do not have an immediate effect (or have just part of it), they are first queued in a message queue and then later executed.
One message is enqueued when one API (that requires message queue) is called. The message is dequeued when actual process is executed in R_WM_FrameWait.
The required size is determined by the maximum number of APIs to executed in R_WM_FrameWait.

To delimit the groups of requests needed to be executed within a single screen update, the R_WM_FrameEndMark call should be used. The Id can be used in a subsequent call to R_WM_FrameWait and R_WM_FrameExecuteNext in order to execute all requests waiting in the queue until the delimiter with Id is found.

e.g. 4 Message queues are required in following case.

```
R_WM_FrameWait();

R_WM_WindowMove(); /* Message queue is required */
R_WM_WindowSwap(); /* Message queue is required */
R_WM_WindowSwap(); /* Message queue is required */
 :
R_WM_FrameEndMark(); /* Message queue is required */
R_WM_FrameWait();
```

Following table shows whether each function uses the message queue.

Table 3-1 Function type

Function name	Message Queue
R_WM_DevInit	-
R_WM_DevEventRegister	-
R_WM_DevDeinit	-
R_WM_DevInfoGet	-
R_WM_GetVersionString	-
R_WM_ScreenTimingSet	-
R_WM_ScreentimingSetByName	-
R_WM_ScreenColorFormatSet	-
R_WM_ScreenBgColorSet	✔
R_WM_ScreenColorCurveSet	✔
R_WM_ScreenGammaSet	✔
R_WM_ScreenEnable	✔
R_WM_ScreenDisable	✔
R_WM_ScreenVocaInit	✔

R_WM_ScreenVocaDeInit	✔
R_WM_ScreenVocaCreate	✔
R_WM_ScreenVocaDelete	✔
R_WM_ScreenVocaEnable	✔
R_WM_ScreenVocaDisable	✔
R_WM_ScreenVocaExpImgSet	✔
R_WM_ScreenVocaClutSet	✔
R_WM_ScreenActivityMonEnable	✔
R_WM_ScreenActivityMonDisable	✔
R_WM_WindowCapabilitiesSet	-
R_WM_WindowCreate	✔
R_WM_WindowDelete	✔
R_WM_WindowEnable	✔
R_WM_WindowDisable	✔
R_WM_WindowMove	✔
R_WM_WindowResize	✔
R_WM_WindowColorFmtSet	✔
R_WM_WindowAlphaSet	✔
R_WM_WindowPremultipliedAlphaEnable	✔
R_WM_WindowPremultipliedAlphaDisable	✔
R_WM_WindowVerticalMirrorEnable	✔
R_WM_WindowVerticalMirrorDisable	✔
R_WM_WindowSwap	✔
R_WM_WindowNewDrawBufGet	-
R_WM_WindowVisibleBufGet	-
R_WM_WindowCurrentDrawBufGet	-
R_WM_WindowExternalBufSet	✔
R_WM_WindowColorKeyEnable	✔

R_WM_WindowColorKeyDisable	✔
R_WM_WindowClutSet	✔
R_WM_WindowDeleteAllSprites	✔
R_WM_WindowScaledSizeSet	✔
R_WM_FrameEndMark	✔
R_WM_FrameWait	-
R_WM_FrameExecuteNext	-
R_WM_FrameExecuteVoca	-
R_WM_FrameExecuteDiscom	-
R_WM_CaptureCreate	✔
R_WM_CaptureDelete	✔
R_WM_CaptureEnable	✔
R_WM_CaptureDisable	✔
R_WM_Cap_CapBufGet	-
R_WM_Cap_DispBufGet	-
R_WM_CaptureMove	✔
R_WM_CaptureResize	✔
R_WM_CaptureScaledSizeSet	✔
R_WM_CaptureExtVsyncSet	✔
R_WM_SpriteCreate	✔
R_WM_SpriteEnable	✔
R_WM_SpriteDisable	✔
R_WM_SpriteMove	✔
R_WM_SpriteBufSet	✔
R_WM_SpriteDelete	✔
R_WM_ErrorCallbackSet	-
R_WM_ErrorHandler	-
R_WM_ColorFmtBitsPerPixGet	-

R_WM_DiscomCreate	✔
R_WM_DiscomDelete	✔
R_WM_DiscomEnable	✔
R_WM_DiscomDisable	✔
R_WM_DiscomCrcSet	✔
R_WM_DiscomCrcGet	-

✔: Message queue is used. - : Message queue is not used.

3.1.7 Discom unit

DISCOM unit can calculate CRC32 from VDCE internal synthesized signal.
The characteristics of the data handled by DISCOM unit are as follows.

- After the completion of 4Layer synthesis in VDCE.
- Color format is ARGB8888 with fixed alpha (0xFF). Original alpha value is already multiplied by RGB data.
- Before the distortion correction by VOWE and gamma correction.

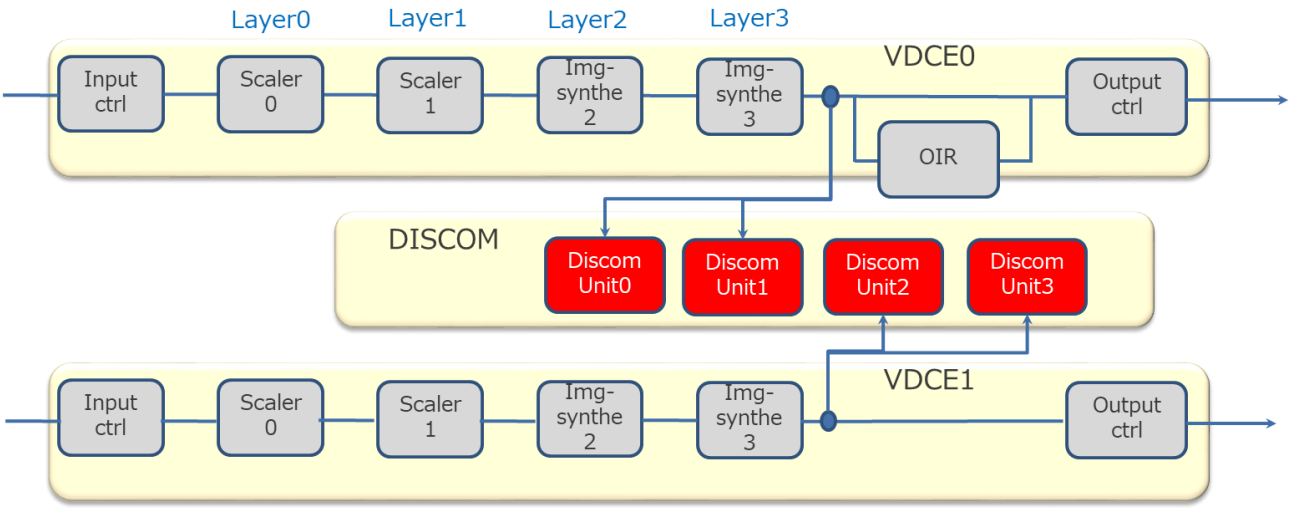


Figure 3-8 DISCOM Unit

Initial value of the CRC code at the start of calculation is fixed as 0xFFFFFFFF.
Refer to the H/W User's manual for the CRC calculation algorithm.

3.1.8 VOCA monitor area

VOCA has two features.

- Video output monitor
  VOCA can monitor whether the display content is correctly output by the Video Output signal.
  Total 16 VOCA monitor areas can be created to compare the expected image.
  VOCA monitor areas are shared by all WM units.

- Activity monitor
  Activity monitor can monitor the Vsync signal interval.
  Activity monitor has 2 channels to monitor each WM units.

VOCA block is existing after VDCE output.
The monitor signal is after the distortion correction by VOWE and gamma correction.

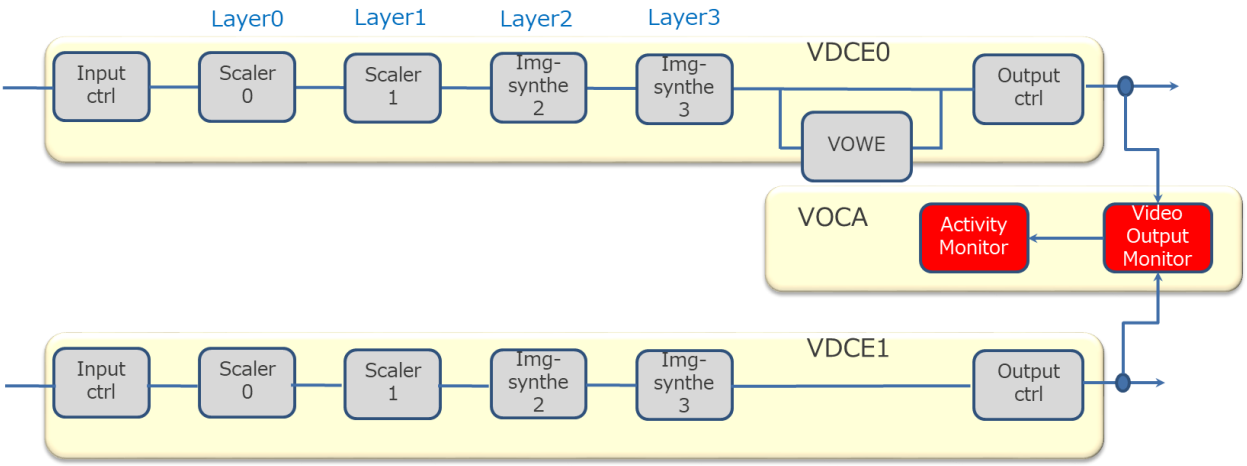


Figure 3-9 VOCA monitor

3.2 Using the API

3.2.1 Framework initialization

The very first framework function that should be called is R_WM_DevInit which initializes the framework.
In the background, it takes care about initializing all the H/W macros used by the WM, as well as framework internals.

R_WM_DevInit initialize the following.

- Message queue
  User allocates buffer for message queue and sets the start address and the number of queues.
  User should prepare more queues than the maximum value to use. See 3.1.6 for the detail.

- Event callback
  Event callback can be installed optionally. User can obtain some timing and error occurrence by event callback.

- Heap memory
  If frame buffer is allocated with internal allocate mode, heap memory is required.
  See 3.2.4.4 for the detail.
  Note: Heap memory is common resource for WM unit 0 and WM unit1. User should set the same value in WM unit0 and WM unit1 when using two WM units.

  Example of using Heap memory:

```
#define WM_MSG_QUEUE_LEN (32)

static r_wm_Msg_t       wm_msg_queue[WM_MSG_QUEUE_LEN];
static r_cdi_Heap_t     cpu_heap, vid_heap;

/* Allocate Heap memory */
R_CDI_InitHeapManager(&cpu_heap); /* &cpu_heap is 2nd argument */
R_CDI_InitHeapManager(&vid_heap); /* &vid_heap is 2nd argument */

/* Initialize */
R_WM_DevInit(UNIT_0, wm_msg_queue, WM_MSG_QUEUE_LEN, R_NULL,
             (void*)&cpu_heap, (void*)&vid_heap);
```

  The argument “CpuHeap” and “VidHeap” of R_WM_DevInit are not directly referenced in RGL but are passed directly to R_WS_Sys_Heap_Set and controlled by the porting layer.
  The default implementation is to specify a pointer of type r_cdi_Heap_t. However, user can control different types by changing the memory control implementation of R_WS_Sys_Heap_Set.
  See Porting Layer guide for the detail.

  When internal allocate mode is not used, WM does not need heap memory.
  User can set R_NULL to the arguments.

  Example without using heap memory:

```
#define WM_MSG_QUEUE_LEN (32)

static r_wm_Msg_t     wm_msg_queue[WM_MSG_QUEUE_LEN];

/* Initialize */
R_WM_DevInit(UNIT_0, wm_msg_queue, WM_MSG_QUEUE_LEN, R_NULL, R_NULL, R_NULL);
```

It is highly recommended to register an error callback handler via R_WM_ErrorCallbackSet.

3.2.2 Screen

3.2.2.1 Display area

The next step is setting up the physical display. This can be done by manually specifying the display properties via R_WM_ScreenTimingSet or referencing the display by its name from the provided display timings database via R_WM_ScreenTimingSetByName.

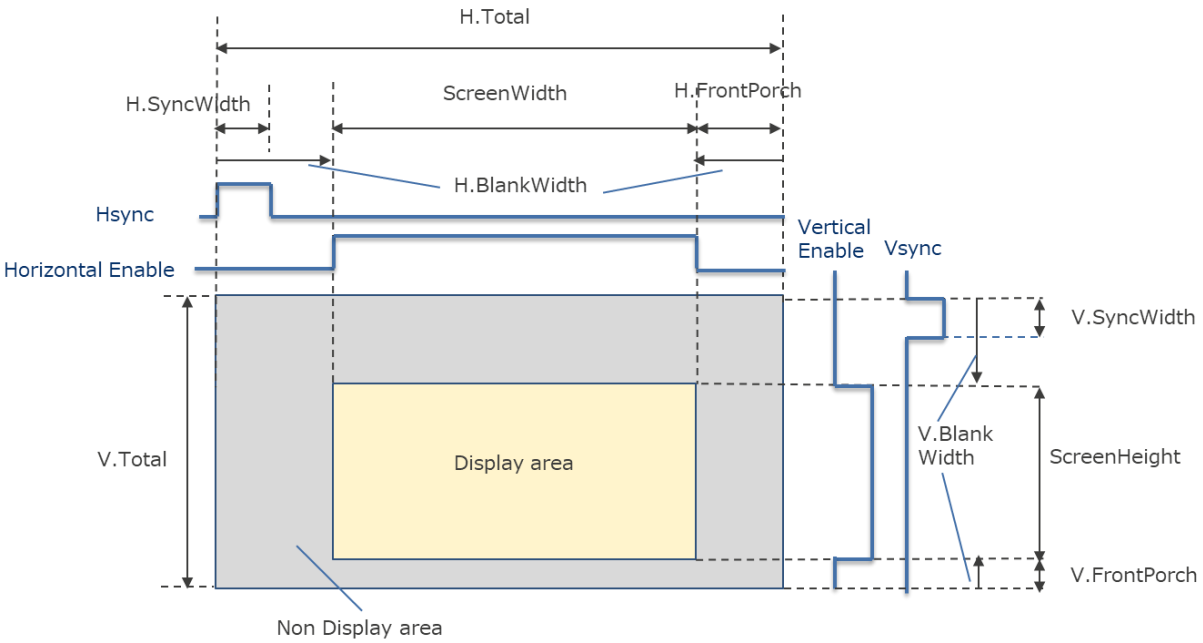


Figure 3-10 Screen signal image

Table 3-2 Parameter range

Timing parameter	Range	
	Min	Max
Timing->ScreenWidth	16	1280 / 1024 / 480 (*2)
Timing->ScreenHeight	16	1024 / 320 (*2)
Timing->H.Total	32	2048
Timing->H.BlankWidth	16	2032
Timing->H.SyncWidth	1	Timing->H.BlankWidth-1.
Timing->H.FrontPorch	0 (*4)	Timing->H.BlankWidth-2.
Horizontal back porch (*1)	1	Timing->H.BlankWidth-1
Timing->V.Total	21	2048
Timing->V.BlankWidth	5	2032
Timing->V.SyncWidth	1	Timing->V.BlankWidth-2
Timing->V.FrontPorch	1	Timing->V.BlankWidth-2
Vertical back porch (*1)	1	Timing->V.BlankWidth-2
Timing->PixelClock	1	- (*3)

(*1): back porch = BlankWidth – SyncWidth – FrontPorch
(*2): maximum value is depending on RH850/D1x device.
(*3): maximum pixel clock is depending on RH850/D1x device and output format (LVTTL, Serial RGB etc).
(*4): Timing->H.FrontPorch = 0 is allowed, but in this case VOCA monitor feature cannot be worked.

3.2.2.2 Output color data

R_WM_ScreenColorFormatSet sets the output signal format. VDCE outputs 24 color data signals (LCD_DATA23..00). Data signal assignment is depending on following settings.

- Endian is set by R_WM_OUTCOLORFMT_FLAG_ENDIAN flag.
- Swap BR is set by R_WM_OUTCOLORFMT_FLAG_SWAP_BR flag

Table 3-3 Output color data

Parameter			LCD_DATA23 <---> LCD_DATA00
Format	Endian	Swap BR	
RGB888	Off	Off	$R_7R_6R_5R_4R_3R_2R_1R_0$ $G_7G_6G_5G_4G_3G_2G_1G_0$ $B_7B_6B_5B_4B_3B_2B_1B_0$
		On	$B_7B_6B_5B_4B_3B_2B_1B_0$ $G_7G_6G_5G_4G_3G_2G_1G_0$ $R_7R_6R_5R_4R_3R_2R_1R_0$
	On	Off	$R_0R_1R_2R_3R_4R_5R_6R_7$ $G_0G_1G_2G_3G_4G_5G_6G_7$ $B_0B_1B_2B_3B_4B_5B_6B_7$
		On	$B_0B_1B_2B_3B_4B_5B_6B_7$ $G_0G_1G_2G_3G_4G_5G_6G_7$ $R_0R_1R_2R_3R_4R_5R_6R_7$
RGB666	Off	Off	0 0 0 0 0 0 R_7R_6 $R_5R_4R_3R_2G_7G_6G_5G_4$ $G_3G_2B_7B_6B_5B_4B_3B_2$
		On	0 0 0 0 0 0 B_7B_6 $B_5B_4B_3B_2G_7G_6G_5G_4$ $G_3G_2R_7R_6R_5R_4R_3R_2$
	On	Off	0 0 0 0 0 0 R_2R_3 $R_4R_5R_6R_7G_2G_3G_4G_5$ $G_6G_7B_2B_3B_4B_5B_6B_7$
		On	0 0 0 0 0 0 B_2B_3 $R_4R_5R_6R_7G_2G_3G_4G_5$ $G_6G_7B_2B_3B_4B_5B_6B_7$
RGB565	Off	Off	0 0 0 0 0 0 0 0 $R_7R_6R_5R_4R_3G_7G_6G_5$ $G_4G_3G_2B_7B_6B_5B_4B_3$
		On	0 0 0 0 0 0 0 0 $B_7B_6B_5B_4B_3G_7G_6G_5$ $G_4G_3G_2R_7R_6R_5R_4R_3$
	On	Off	0 0 0 0 0 0 0 0 $R_3R_4R_5R_6R_7G_2G_3G_4$ $G_5G_6G_7B_3B_4B_5B_6B_7$
		On	0 0 0 0 0 0 0 0 $B_3B_4B_5B_6B_7G_2G_3G_4$ $G_5G_6G_7R_3R_4R_5R_6R_7$

R_n, G_n and B_n (n=0~7) are RGB888 data in VDCE internal. For example, R=128 (0x80): R_7=1b, R_6 = 0b, ... R_0 =0b.

3.2.2.3 Gamma Correction (1)

WM driver provides two functions for Gamma correction.

- R_WM_ScreenColorCurveSet
- R_WM_ScreenGammaSet

R_WM_ScreenColorCurveSet sets the 33 reference points for each correction color component.
These are the values corresponding to 0, 8, 16, ... ,252, 256 of the values before correction.
Regardless of the output color data, consider RGB888. The range of each component is 0 to 255.

For the correction in VDCE, the incoming color information of each channel is split into 32 equally sized segments each covering a range 8 color value. For each of these 8 values in a segment, the same gain factor applies.
For each segment of each color, the gain factor must be in range [x0.00 … x2.0], thus the values between two reference points may have a difference in range of [0 … +16].

The origin is always at point zero: (R, G, B) = (0, 0, 0). The slope is always positive. If user setting value exceeds above limitation, the curve will be approximated as close as possible while staying within the H/W limitations.

Example

```
                                                /*  B    G    R    A */
static r_wm_ClutEntry_t ColorCurv[33] =        {{   0,   0,   0,   0}, /*  0 */
                                                {   0,   8,  25,   0}, /*  1 */
                                                {   1,  16,  40,   0}, /*  2 */
                                                {   2,  24,  53,   0}, /*  3 */
                                                {   4,  32,  64,   0}, /*  4 */
                                                {   6,  40,  74,   0}, /*  5 */
                                                {   9,  48,  84,   0}, /*  6 */
                                                {  12,  56,  93,   0}, /*  7 */
                                                {  16,  64, 101,   0}, /*  8 */
                                                {  20,  72, 110,   0}, /*  9 */
                                                {  25,  80, 118,   0}, /* 10 */
                                                {  30,  88, 125,   0}, /* 11 */
                                                {  36,  96, 133,   0}, /* 12 */
                                                {  42, 104, 140,   0}, /* 13 */
                                                {  49, 112, 147,   0}, /* 14 */
                                                {  56, 120, 154,   0}, /* 15 */
                                                {  64, 128, 161,   0}, /* 16 */
                                                {  73, 136, 168,   0}, /* 17 */
                                                {  81, 144, 174,   0}, /* 18 */
                                                {  91, 152, 181,   0}, /* 19 */
                                                { 100, 160, 187,   0}, /* 20 */
                                                { 111, 168, 193,   0}, /* 21 */
                                                { 121, 176, 199,   0}, /* 22 */
                                                { 133, 184, 205,   0}, /* 23 */
                                                { 145, 192, 211,   0}, /* 24 */
                                                { 157, 200, 217,   0}, /* 25 */
                                                { 170, 208, 223,   0}, /* 26 */
                                                { 183, 216, 228,   0}, /* 27 */
                                                { 197, 224, 234,   0}, /* 28 */
                                                { 211, 232, 239,   0}, /* 29 */
                                                { 226, 240, 245,   0}, /* 30 */
                                                { 241, 248, 250,   0}, /* 31 */
                                                { 255, 255, 255,   0}  /* 32 */
};

R_WM_ScreenColorCurveSet(LOC_WM_UNIT, 33, ColorCurv);
```

Following figure is the correction result of above R_WM_ScreenColorCurveSet example.
Because ColorCurve[1].R, ColorCurve[2].R and ColorCurve[3].R exceed the limitation of slope, correction value is round down to the maximum slope .

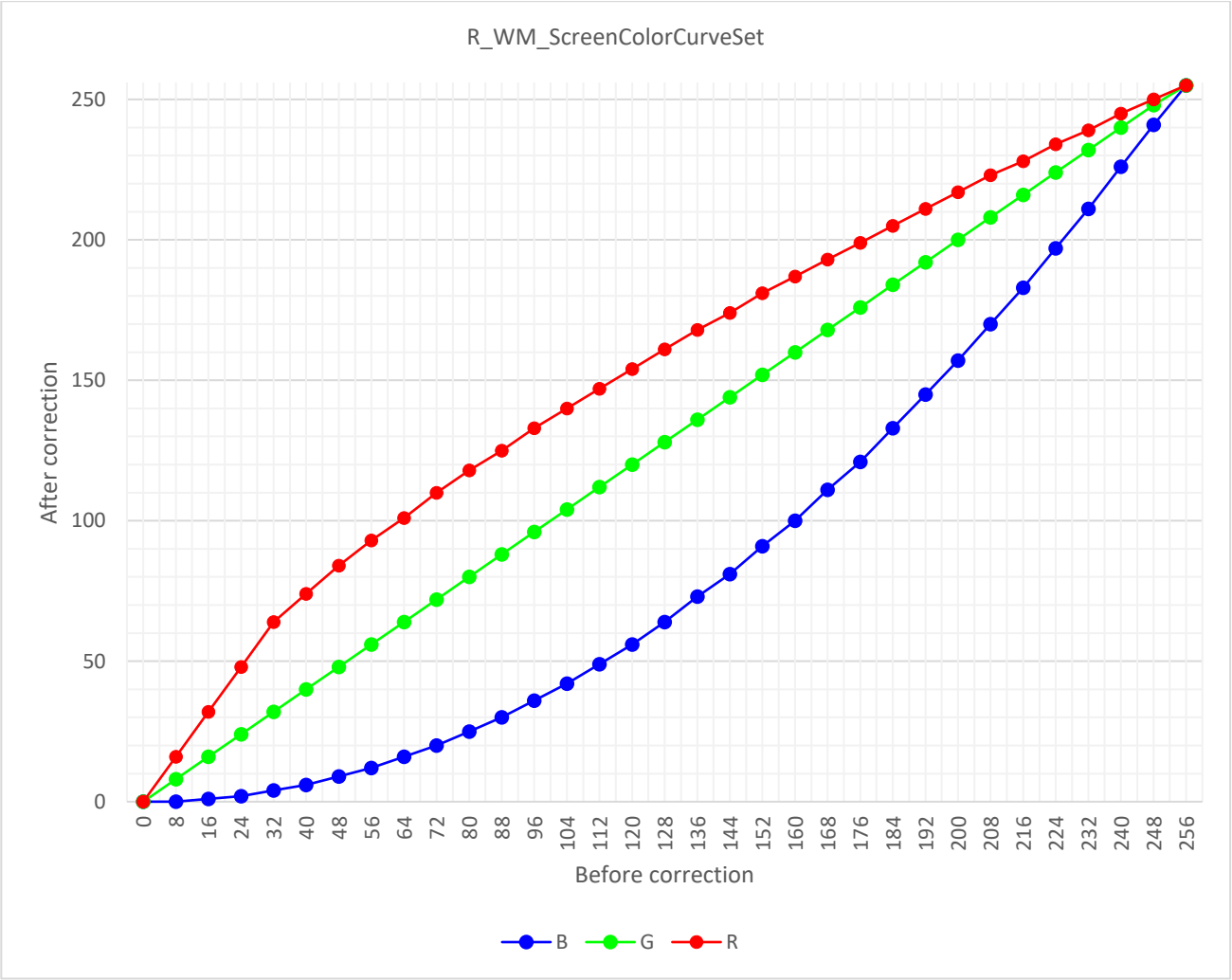


Figure 3-11 R_WM_ScreenColorCurveSet

3.2.2.4 Gamma Correction (2)

R_WM_ScreenGammaSet sets the gamma value for each color components.
WM driver calculates from following formula and makes the 33 reference points to execute the same processing as R_WM_ScreenColorCurveSet.

Formula: $y = x^{1/\gamma}$
y: after correction color component assumed range [0.0-1.0]
x: before correction color component assumed range [0.0-1.0]
γ = (specified gamma value) / 128

* Here, the division is fixed point arithmetic.

The curve will be approximated as close as possible while staying within the H/W limitations.
Following figure is the correction result depending on specified gamma value. RGB range is 0 to 255.

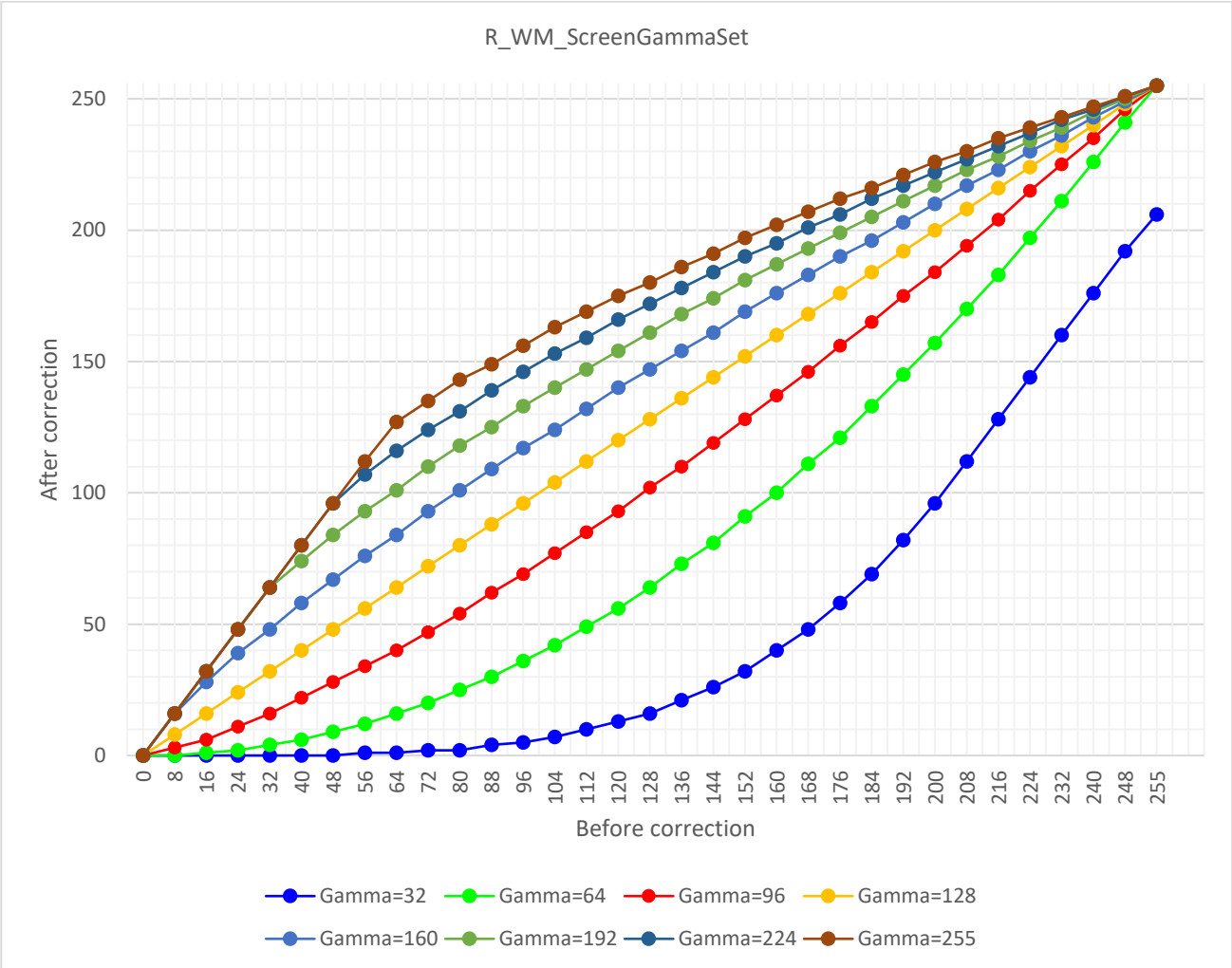


Figure 3-12 R_WM_ScreenGammaSet

3.2.3 Window

Window is managed by the structure r_wm_Window_t

3.2.3.1 Types

There are three distinct types of windows. The window type is specified with parameter “Mode” and “ColorFmt”.

Table 3-4 Window type

Window type	Parameter	
	Mode	ColorFmt
Frame buffer window	R_WM_WINMODE_FB	Not RLE format
RLE window	R_WM_WINMODE_FB	RLE format
Sprite windows	R_WM_WINMODE_SPRITES	Not RLE format

3.2.3.2 Color formats

There are 3 general color format types of Frame buffer window and sprite window.

- RGB
- YCbCr
- CLUT

RGB and YCbCr are direct color formats have a predefined number of bytes defining a color for every pixel in the FB. This would be the logical choice for a FB whose content requires a significant number of colors, e.g. displaying digital images.
CLUT formats store a color lookup table index for every pixel. The lookup table can be 256, 16 or 2 color entries large, yielding 8, 4 or 1 bits per pixel in the FB. The obvious downside of this type is the number of colors available. On the up side, a lot of storage can be saved for FBs that don't need more than a few colors (e.g. 2 in case of some text), plus the nice color effects can be obtained just by manipulating the lookup table, without touching the single pixel-indices in the FB at all.

Table 3-5 Cloro Format for Frame Buffer and Sprite window

Type	Format	bpp	Memory image (MSB <--> LSB)	Remarks
RGB	R_WM_COLORFMT_RGB565	16	RRRRRGGG GGGBBBBB	
	R_WM_COLORFMT_ARGB1555	16	ARRRRRGG GGGBBBBB	
	R_WM_COLORFMT_ARGB4444	16	AAAARRRR GGGGBBBB	
	R_WM_COLORFMT_RGB0888	32	xxxxxxxx RRRRRRRR GGGGGGGG BBBBBBBB	
	R_WM_COLORFMT_ARGB8888	32	AAAAAAAA RRRRRRRR GGGGGGGG BBBBBBBB	
	R_WM_COLORFMT_RGBA5551	16	RRRRRGGG GGBBBBBA	
	R_WM_COLORFMT_RGBA8888	32	RRRRRRRR GGGGGGGG BBBBBBBB AAAAAAAA	
CLUT	R_WM_COLORFMT_CLUT8	8	TTTTTTTT	
	R_WM_COLORFMT_CLUT4	4	TTTT	
	R_WM_COLORFMT_CLUT1	1	T	
YCbCr	R_WM_COLORFMT_YCBCR_422	16	UUUUUUUU YYYYYYYY VVVVVVVV YYYYYYYY	32bits / 2Pixels
	R_WM_COLORFMT_YCBCR_444	32	xxxxxxxx VVVVVVVV YYYYYYYY UUUUUUUU	
	R_WM_COLORFMT_YUV_YUYV	16	VVVVVVVV YYYYYYYY UUUUUUUU YYYYYYYY	32bits / 2Pixels
	R_WM_COLORFMT_YUV_UYVY	16	YYYYYYYY VVVVVVVV YYYYYYYY UUUUUUUU	32bits / 2Pixels
	R_WM_COLORFMT_YUV_YVYU	16	UUUUUUUU YYYYYYYY VVVVVVVV YYYYYYYY	32bits / 2Pixels
	R_WM_COLORFMT_YUV_VYUY	16	YYYYYYYY UUUUUUUU YYYYYYYY VVVVVVVV	32bits / 2Pixels

A/R/G/B: each component of ARGB, x: unused, T: table index,
Y: Y component, U: Cb or U component, V: Cr or V component

RLE color formats use the Run Length Encoding feature of the SPEA Sprite Engine H/W macro. This means that the FB pixels are specified according to TARGA RLE format description. Only the image definition part of the format is supported, the information found in the header of the TARGA file is anyway specified with the other window parameters. The RLE format is a great choice for images with larger flat-colored sections.

Table 3-6 Cloro Format for RLE window

Type	Format	RLE data (before decoding)	Window (after decoding)
RLE	R_WM_COLORFMT_RLE24ARGB8888	RGB888	ARGB8888 (A=0xFF fixed)
	R_WM_COLORFMT_RLE24RGB0888	RGB888	RGB0888

3.2.3.3 Stacking (Z-order)

Windows are internally sorted by the framework using their PosZ property. The PosZ property is not used for direct mapping to a specific layer number but to maintain the order of the windows only. Layer of the window is assigned in ascending order of PosZ.
The assignment of windows to layers also depends on the type of windows. As each of the H/W layers supports a subset of the features, the WM driver will select an appropriate layer based on the features requested (e.g. Sprite windows; Capture windows; RLE color format; etc.) during window creation.

The following functions change the layer assignment of the Window.

Table 3-7 Create functions

Function	Condition
R_WM_WindowCreate	Always.
R_WM_WindowMove	PosZ is changed from the previous setting.
R_WM_WindowClutSet	Clut[n].A (!= 0xFF) is set to the window assigned to Layer0.
R_WM_CaptureCreate	Always.

Following table shows the available feature of each layer.

Table 3-8 Layer feature

Feature	How to Specify		Available layer			
	Function	Parameter	Layer0	Layer1	Layer2	Layer3
RLE window	R_WM_WIndowCreate	ColorFmt (= RLE type)	✔	- (*2)	- (*2)	- (*2)
YUV, YCbCr	R_WM_WindowCreate	ColorFmt (= YCbCr type)	✔	✔	-	-
Video input	R_WM_CaptureCreate	-	✔ (*1)	✔ (*1)	-	-
Sprite window	R_WM_WindowCreate	Mode (= Sprite mode)	- (*2)	✔	✔	✔
Alpha blending of whole window	R_WM_WindowCreate	Alpha (!= 0xFF)	-	✔	✔	✔
CLUT with Alpha blending	R_WM_WindowCreate	ColorFmt (= CLUT type)	-	✔	✔	✔
		Clut[n].A (!= 0xFF)				
	R_WM_WindowClutSet	Clut[n].A (!= 0xFF)	-	✔	✔	✔
ColorKey	R_WM_WindowCreate	ColorKey.Enabled (= 1)	-	✔	✔	✔

(*1) Depending on device. See 3.2.11 for the detail.
(*2) Default is not available. Configurable by R_WM_WindowCapabilitiesSet depending on the R850/D1x device.

In case there is a conflict regarding the requested order of layers versus the requested set of features for a layer, the automatic mapping will fail. The queue execution from above functions will return an error.

This is an example of window creation in the default configuration.

Window A : Frame buffer window (which can be assigned to Layer0)
Window B : Sprite window
Window C : Frame buffer window
Window D : RLE window

Order1: Create the Window of Window A with PosZ = 30
WM step1: Window A is assigned to Layer0.

Order2: Create the Window of Window B with PosZ = 20
(WM consider form ascending order of PosZ.)
WM step1: Layer0 is skipped because Window B cannot be assigned to Layer0 as default configuration.
WM step2: Window B is assigned to Layer1.
WM step3: Window A is assigned to Layer2. (Move from Layer0 to Layer2.)

Order3: Create the Window of Window C with PosZ = 40
WM step1: Layer0 is skipped because Window B cannot be assigned to Layer0 as default configuration.
WM step2: Window B is assigned to Layer1.
WM step3: Window A is assigned to Layer2.
WM step4: Window C is assigned to Layer3.

Order4: Create the Window of Window D with PosZ = 50
WM step1: Layer0 is skipped because Window B cannot be assigned to Layer0 as default configuration.
WM step2: Window B is assigned to Layer1.
WM step3: Window A is assigned to Layer2.
WM step4: Window C is assigned to Layer3.
WM step5: No upper layer is remained for Window D. WM notifies an error to user.

Order5: Create the Window of Window B with PosZ = 10
WM step1: Window D is assigned to Layer0.
WM step2: Window B is assigned to Layer1.
WM step3: Window A is assigned to Layer2.
WM step4: Window C is assigned to Layer3.

Table 3-9 Window assignment

Order	Layer0	Layer1	Layer2	Layer3
Order 1	Window A	-	-	-
Order 2	-	Window B	Window A	
Order 3	-	Window B	Window A	Window C
Order 4	-	Window B	Window A	Window C
Order 5	Window D	Window B	Window A	Window C

3.2.4 Frame buffer window

3.2.4.1 Creation

The function R_WM_WindowCreate creates the window on the desired WM Unit, with the parameters specified in the user-maintained (in terms of storage persistency) Window structure. The Window structure is a reference parameter, which is used driver internally until R_WM_WindowDelete is called. Make sure that Window is not allocated as local variable that may be discarded while the Window is still in use.

Example for creating Window for Frame buffer window size is W: 240 x H: 200

Example

```
static r_wm_Window_t Window;

memset (&Window, 0, sizeof (r_wm_Window_t));
Window.Mode                   = R_WM_WINMODE_FB;
Window.Status                 = R_WM_WINSTATUS_NOT_INITIALIZED;
Window.ColorFmt               = R_WM_COLORFMT_ARGB8888;
Window.Alpha                  = 0xFF;
Window.PosX                   = 10;
Window.PosY                   = 20;
Window.PosZ                   = 0;
Window.Width                  = 240;
Window.Height                 = 200;
Window.Pitch                  = 256;
Window.ScaledWidth            = 0;
Window.ScaledHeight           = 0;
Window.Surface.Fb.BufMode     = R_WM_WINBUF_ALLOC_INTERNAL;
Window.Surface.Fb.BufNum      = 2;
Window.Surface.Fb.Buffer      = 0;
R_WM_WindowCreate(UNIT_0, &Window);
```

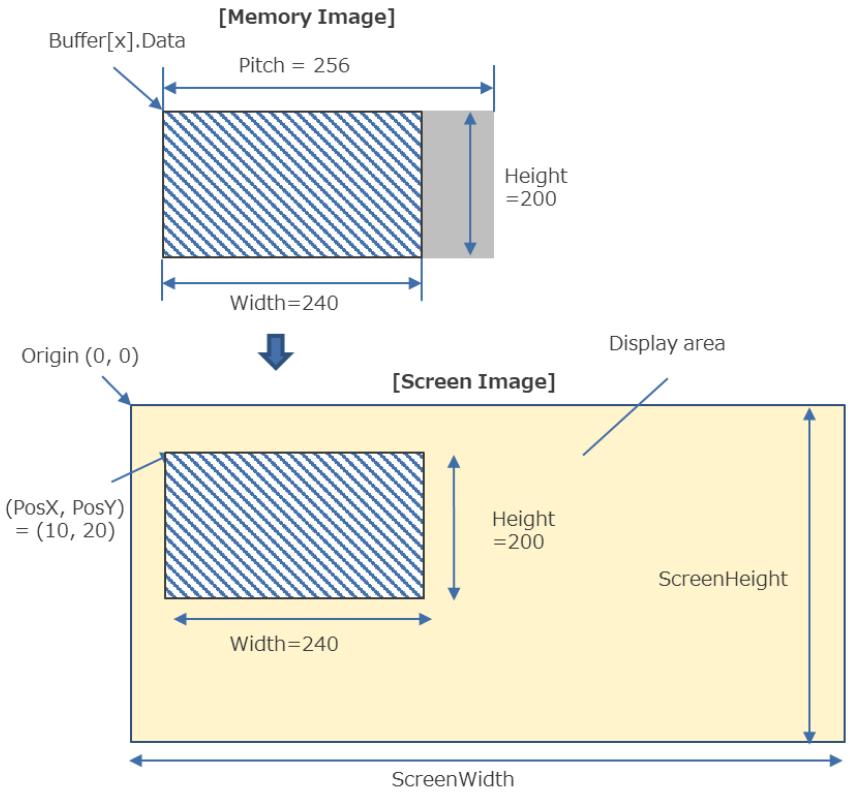


Figure 3-13 Creating Window for Frame Buffer

Regarding with the frame buffer start address, see Frame buffer chapter.
Pitch should be the multiple of 128 bytes. Thus, if color format is 32bpp, Pitch should be 32 pixel-aligned.
Following table shows the range information.

Table 3-10 Parameter range w/o scaling-up

Parameter	Setting unit	Range		Alignment
		Min	Max	
PosX	Pixel	-1280	1279	-
PosY	Pixel	-1024	1023	-
Width	Pixel	3	1280	-
Height	Pixel	1	1024	-
Pitch	Pixel	Width	8192 and (261120 / bpp)	128 Byte

If Window position is out of screen, WM disable the layer internally.

3.2.4.2 Frame buffer size

Frame buffer size is calculated by following formula.

without scaling-up case:
Frame buffer size [Byte] = Pitch * Height * N [Byte/pixel]

with scaling-up case:
Frame buffer size [Byte] = Pitch * ScaledHeight * N [Byte/pixel]

3.2.4.3 Frame buffer allocate mode

There are two types of allocate mode of frame buffer, internal mode and external mode.

In case of internal mode, R_WM_WindowCreate allocates the frame buffer from Heap memory.
Example for internal mode:

```
static r_wm_Window_t Window;

Window.Mode                   = R_WM_WINMODE_FB;
:
Window.Surface.Fb.BufMode     = R_WM_WINBUF_ALLOC_INTERNAL;
Window.Surface.Fb.BufNum      = 2;
Window.Surface.Fb.Buffer      = R_NULL;

R_WM_WindowCreate(UNIT_0, &Window);
```

In case of external mode, the user should allocate the buffers and specifies the address with R_WM_WindowCreate. The framebuffer start address must be 128 bytes aligned.

Example for external mode:

```
static r_wm_Window_t Window;
static r_wm_WinBuffer_t ExternalBuffer[2];

ExternalBuffer[0].Status = R_WM_WINBUF_FREE;
ExternalBuffer[1].Status = R_WM_WINBUF_FREE;
ExternalBuffer[0].Data = (void *)0x40000000; /* allocating buffer address */
ExternalBuffer[1].Data = (void *)0x40100000; /* allocating buffer address */

Window.Mode                   = R_WM_WINMODE_FB;
:
Window.Surface.Fb.BufMode     = R_WM_WINBUF_ALLOC_EXTERNAL;
Window.Surface.Fb.BufNum      = 2;
Window.Surface.Fb.Buffer      = ExternalBuffer;

R_WM_WindowCreate(UNIT_0, &Window);
```

User can control easily by using the internal mode.
However, in the following cases, the external mode is recommended.

- Improving the efficiency of bus access
  (e.g One Frame buffer is allocated in VRAM0, one frame buffer is allocated in VRAM1 to avoid conflict.)
- Repeating R_WM_WindowCreate and R_WM_WindowDelete of multiple Windows

3.2.4.4 Heap memory

Heap memory for internal allocate mode is prepared with R_WM_DevInit. Two types of heap memory are required.

Table 3-11 Heap memory type

Argument	Purpose	Required Size	Access H/W	Recommended location
CpuHeap	Management of Frame buffer	8[Byte] * BufNum	CPU	LRAM
VidHeap	Frame buffer	Frame buffer size * BunNum	VDCE, 2DGPU, etc...	VRAM, SDRAM

3.2.4.5 Status transition

Frame buffer has the status as following.

Table 3-12 Frame buffer State

No.	State Name	Description
(1)	Uninitialized	Buffer is not initialized.
(2)	Free	Buffer is free.
(3)	Render Started	Rendering to the buffer is started.
(4)	Render Finished	Rendering to the buffer is finished.
(5)	On Screen	Buffer is on screen.

User manages these transitions by obtaining the free frame buffer from the framework via R_WM_WindowNewDrawBufGet and R_WM_WindowSwap call.

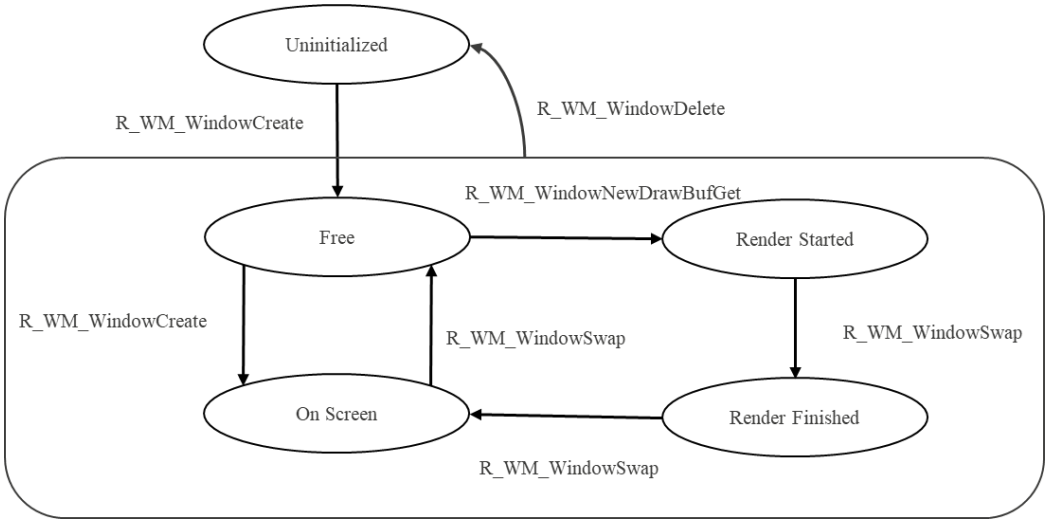


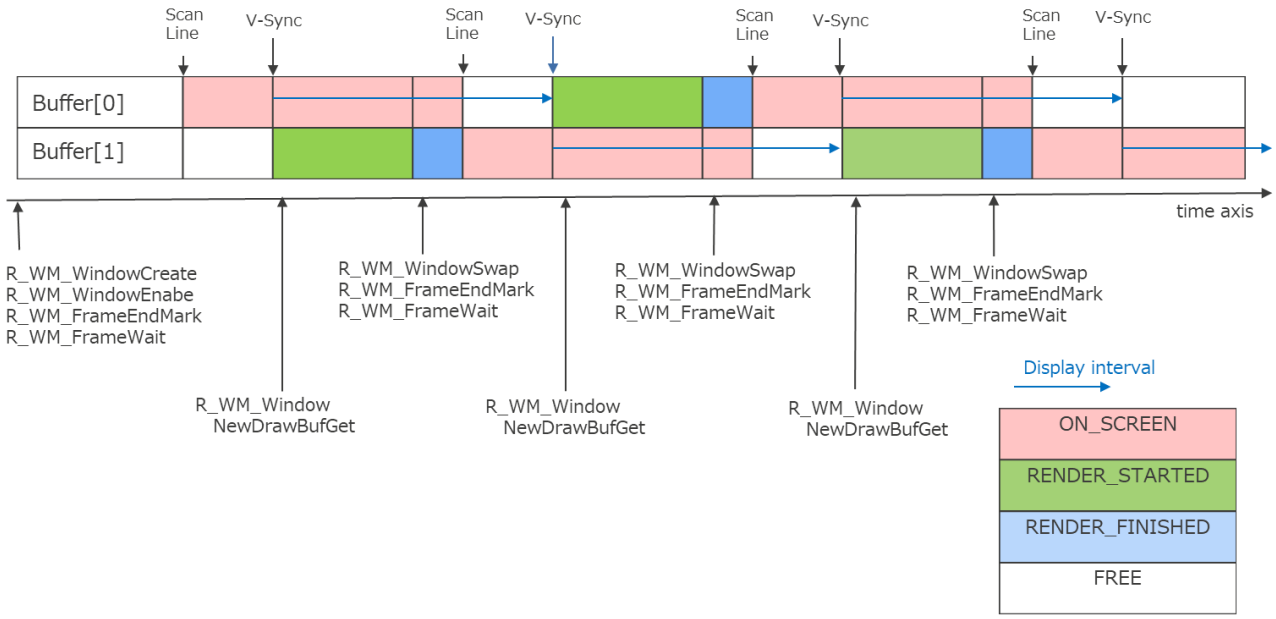
Figure 3-14 Buffer State Transition

Table 3-13 State Transition detail

Timing (*1)		Status		Transition	
Function	**Sync**	**Before**	**After**	**Target**	**How to search**
R_WM_Window Create	Sync	Uninitialized	Free	All buffers	WM transitions all buffers to Free state in case of internal allocate mode. (*2)
	Async	Free	On Screen	1 buffer	Search from the first buffer, and WM transitions the first found Free buffer.
R_WM_Window NewDrawBufGet	Sync	Free	Render Started	1 buffer	Search from the On-Screen buffer, and WM transitions the first found Free buffer.
R_WM_Window Swap	Sync	Render Started	Render Finished	1 buffer	There is only one buffer with Render Started status.
	Async	On Screen	Free	1 buffer	There is only one buffer with On Screen status.
		Render Finished	On Screen	1 buffer	There is only one buffer with Render Finished status.
R_WM_Window Delete	Async	any	Uninitialized	All buffers	WM freed management memory in case of internal allocate mode.

(*1) The meanings of Sync and Async are as follows.
Sync: Synchronous execution with function call.
Async: Asynchronous execution within R_WM_FrameWait process.
(*2) In case of external allocate mode, user should initialize the buffer status as Free.

Following figure shows the image of buffer state transition.



After changing the content of the frame buffer (not part of the WM framework, see Drw2D for example), user should instruct the framework that drawing to this buffer has finished, and the frame buffer should become visible at some point in the future. This is done via R_WM_WindowSwap call. Although the word “swap” resembles the double-buffered scenarios, user can specify any number of frame buffers per window, where the “swap” should then be understood as putting the frame buffer in the queue of frame buffers waiting to become visible one at the time on the screen.

Example:

```
R_WM_WindowCreate(UNIT_0, &Window);
R_WM_WindowEnable(UNIT_0, &Window);
R_WM_FrameEndMark(LOC_WM_UNIT, 0);
R_WM_FrameWait(LOC_WM_UNIT, 0);

while(1)
{
    /* Get the next drawing buffer */
    address = R_WM_WindowNewBufGet(LOC_WM_UNIT, &Window);

    ~ Drawing Process ~

    /* Swap window */
    R_WM_WindowSwap(LOC_WM_UNIT, &Window);
    R_WM_FrameEndMark(LOC_WM_UNIT, 0);
    R_WM_FrameWait(LOC_WM_UNIT, 0);
}
```

3.2.5 Window feature

3.2.5.1 Feature table

Following table shows the features supported by each layer.

Table 3-14 Supported Feature with Layer

Layer	Features						
	Scaling-up	Vertical Rotation	Alpha blending per one pixel	Constant Alpha	Pre-multiplied alpha	Chroma Key	Color Look up table
Layer 0	✔	✔	-	-	-	-	✔
Layer 1	✔	✔	✔	✔	✔	✔	✔
Layer 2	-	✔	✔	✔	✔	✔	✔
Layer 3	-	✔	✔	✔	✔	✔	✔

✔: Supported -: Not supported

Following tables show the supported features depending on the window color format.

Table 3-15 Supported Feature with Color Format for Frame buffer and Sprite window

Layer color format	Features						
	Scaling-up	Vertical Rotation	Alpha blending per one pixel	Constant Alpha	Pre-multiplied alpha	Chroma Key	Color Look up table
R_WM_COLORFMT_RGB565 R_WM_COLORFMT_RGB0888	✔	✔	-	✔	-	✔	-
R_WM_COLORFMT_ARGB1555 R_WM_COLORFMT_ARGB4444 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_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	✔	✔	-	-	-	-	-

✔: Supported -: Not supported

Table 3-16 Supported Feature with Color Format for RLE window

Layer color format	Features						
	Scaling-up	Vertical Rotation	Alpha blending per one pixel	Constant Alpha	Pre-multiplied alpha	Chroma Key	Color Look up table
R_WM_COLORFMT_RLE24ARGB8888 R_WM_COLORFMT_RLE24RGB0888	-	-	-	✔	-	✔	-

✔: Supported -: Not supported

When both layer and color format conditions are supported, the feature can be enabled.

3.2.5.2 Scaling-up

Scaling-up feature can be available depending on device.
ScaledWidth and ScaledHeight are specified the target memory size to be enlarged.
Example for creating Window when 240x200 frame buffer is enlarged to 480x300.

Example

```
static r_wm_Window_t Window;

memset (&Window, 0, sizeof (r_wm_Window_t));
Window.Mode                 = R_WM_WINMODE_FB;
Window.Status               = R_WM_WINSTATUS_NOT_INITIALIZED;
Window.ColorFmt             = R_WM_COLORFMT_ARGB8888;
Window.Alpha                = 0xFF;
Window.PosX                 = 10;
Window.PosY                 = 20;
Window.PosZ                 = 0;
Window.Width                = 480;
Window.Height               = 300;
Window.Pitch                = 256;
Window.ScaledWidth          = 240;
Window.ScaledHeight         = 200;
Window.Surface.Fb.BufMode   = R_WM_WINBUF_ALLOC_INTERNAL;
Window.Surface.Fb.BufNum    = 2;
Window.Surface.Fb.Buffer    = 0;
R_WM_WindowCreate(UNIT_0, &Window);
```

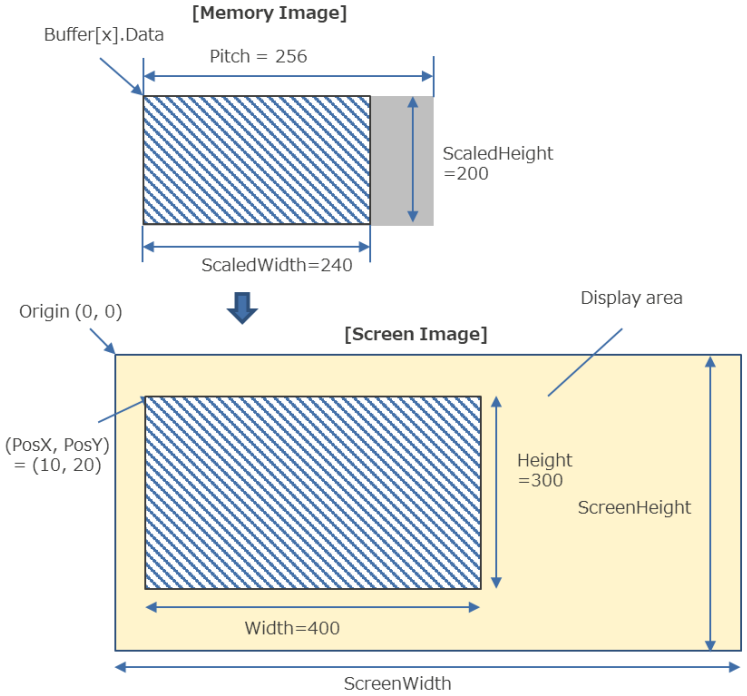


Figure 3-15 Creating Window with scale-up

Following table shows the range information.

Table 3-17 Parameter range with scaling-up

Parameter	Setting unit	Range		Alignment
		Min	Max	
ScaledWidth	Pixel	4	Width - 1	-
ScaledHeight	Pixel	4	Height - 1	-
Pitch	Pixel	ScaledWidth	8192 and (261120 / bpp)	128 Byte

The range of PosX, PosY Width and Height is same as Table 3-10.

3.2.5.3 Vertical Rotation

Vertical rotation is set by following functions depending on input type.
If vertical rotation is enabled, rotation image is synthesized after the image is read from frame buffer.

Table 3-18 Vertical rotation setting

Input Type	Function	Argument	Setting value	rotation
Graphic	R_WM_WindowCreate	Window ->Flags	R_WM_WINFLAG_V_MIRROR flag is on	Enable
			R_WM_WINFLAG_V_MIRROR flag is off	Disable
	R_WM_WindowVerticalMirrorEnable	-	-	Enable
	R_WM_WindowVerticalMirrorDisable	-	-	Disable
Video	R_WM_CaptureCreate	Capture ->Mode	R_WM_CAPMODE_V_MIRRORING flag is on	Enable
			R_WM_CAPMODE_V_MIRRORING flag is off	Disable

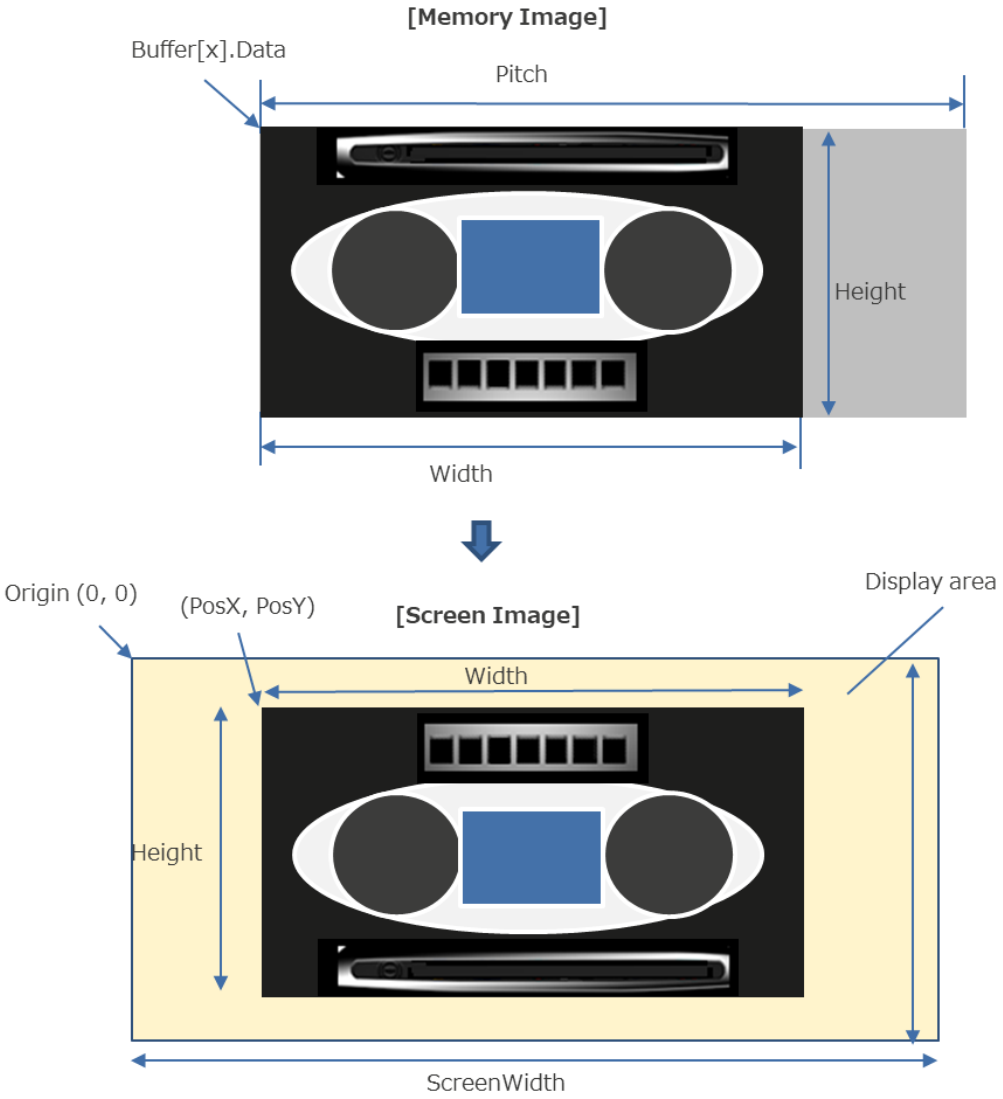


Figure 3-16 Vertical rotation image

3.2.5.4 Alpha Blending

Layer1, Layer2 and Layer3 support alpha blending features.
Layer0 doesn't support alpha blending feature.

- Alpha blending per one pixel
  When color format has alpha channel (i.e. ARGBxxxx, RGBAxxxx, CLUTx), alpha value of each pixel data is enabled.

- Constant Alpha
  The constant alpha value can be set by following functions. Specified vale is multiplied to all pixel data of the layer.

Table 3-19 Constant alpha

Function	Argument	Setting Range
R_WM_WindowCreate	Window->Alpha	0 - 255
R_WM_WindowAlphaSet	Alpha	0 - 255

- Pre-multiplied alpha
  Pre-multiplied alpha feature can be set by following functions. If Pre-multiplied alpha is enabled, VDCE skips the multiplication of input RGB data and alpha value. This feature should be enabled when input RGB data has already pre-multiplied.

Table 3-20 Pre-multiplied alpha

Function	Argument	Setting value	Operation
R_WM_WindowCreate	Window-> UsePremultipliedAlpha	1	Enable
		0	Disable
R_WM_WindowPremultipliedAlphaEnable	-	-	Enable
R_WM_WindowPremultipliedAlphaDisable	-	-	Disable

VDCE calculates the alpha value with following formula.

Table 3-21 Alpha value

Alpha blending per one pixel	calculation
Enable	alpha value = A[in] * AlphaConst / 255
Disable	alpha value = AlphaConst

A[in] is alpha value of the pixel data that is converted from 0 to 255.
AlphaConst is setting value as Constant Alpha.

VDCE calculates each pixel data with following formula.

Table 3-22 Layer color data

Pre-multiplied alpha	calculation
Disable	G[out] = ((G[in1] * alpha value) + (G[in0] * (255 − alpha value))) / 255 B[out] = ((B[in1] * alpha value) + (B[in0] * (255 − alpha value))) / 255 R[out] = ((R[in1] * alpha value) + (R[in0] * (255 − alpha value))) / 255
Enable	G[out] = G[in1] + (G[in0] * (255 − alpha value)) / 255 B[out] = B[in1] + (B[in0] * (255 − alpha value)) / 255 R[out] = R[in1] + (R[in0] * (255 − alpha value)) / 255

G[in0], B[in0], R[in0] are G/B/R value of the lower-layer.
G[in1], B[in1], R[in1] are G/B/R value of the input graphics data.

3.2.5.5 Chromakey

Following functions can set the control of chromakey.

Table 3-23 Chromakey control

Function	Argument	Setting value	Operation
R_WM_WindowCreate	Window->ColorKey.Enable	1	Enable
		0	Disable
R_WM_WindowColorKeyEnable	-	-	Enable
R_WM_WindowColorKeyDisable	-	-	Disable

R_WM_WindowCreate can set one set of chromakey for a layer to convert color data.
The color before conversion is specified by Window->ColorKey.In.RgbKey, and the color after conversion is specified by Window->ColorKey.Out.
In case of RGB format, R/G/B data is set to ColorKey.In.RgbKey. Regardless of alpha value, colors with specified RGB value is converted.
ColorKey.Out is set with RGB format with alpha value. R/G/B values are specified by values in the range that depends on the color format, but alpha value is specified by 8 bits without depending on the color format.

Following table shows the range of each member depending on color format.

Table 3-24 Range of argument

Layer color format	ColorKey.In.RgbKey			ColorKey.Out			
	Blue	Green	Red	Blue	Green	Red	Alpha
R_WM_COLORFMT_RGB565	0-31	0-63	0-31	0-31	0-63	0-31	0-255
R_WM_COLORFMT_RGB0888 R_WM_COLORFMT_ARGB8888 R_WM_COLORFMT_RGBA8888 R_WM_COLORFMT_RLE24ARGB8888 R_WM_COLORFMT_RLE24RGB0888	0-255	0-255	0-255	0-255	0-255	0-255	0-255
R_WM_COLORFMT_ARGB1555 R_WM_COLORFMT_RGBA5551	0-31	0-31	0-31	0-31	0-31	0-31	0-255
R_WM_COLORFMT_ARGB4444	0-15	0-15	0-15	0-15	0-15	0-15	0-255

Example of ARGB8888 format:
Example 1: Convert (B, G, R) = (255, 0, 0) to (B, G, R, A) = (0, 255, 0, 255)

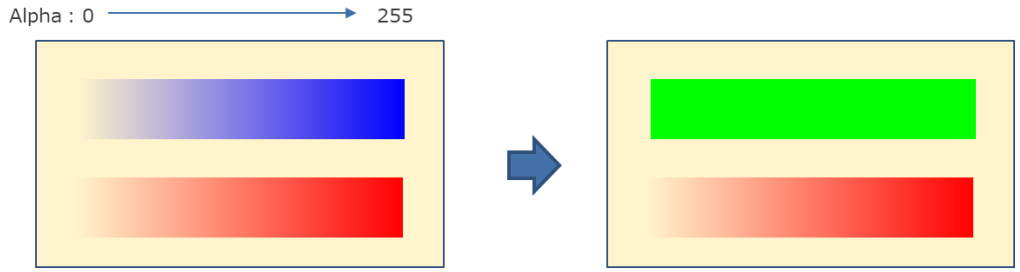


Figure 3-17 Chromakey example 1

Example 2: Convert (B, G, R) = (0, 0, 255) to (B, G, R, A) = (0, 255, 0, 63)

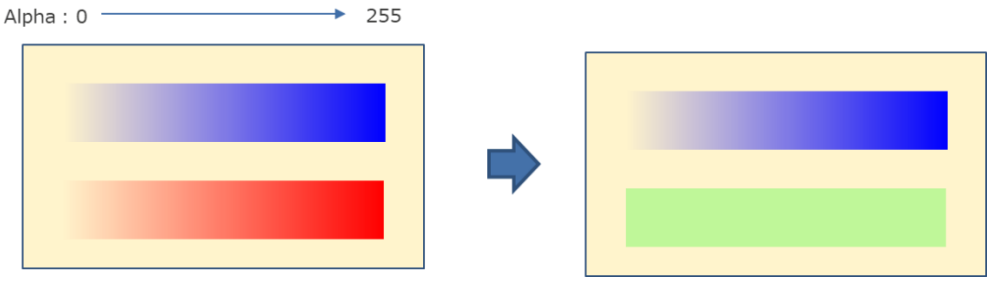


Figure 3-18 Chromakey example 2

3.2.5.6 Color Look up table

CLUT (Color look up table) mode can be selected by layer color format. Each pixel data of frame buffer should consist of the index of the table.

Table 3-25 Range of table index

Layer color format	Range of table index
R_WM_COLORFMT_CLUT8	0 - 255
R_WM_COLORFMT_CLUT4	0 - 15
R_WM_COLORFMT_CLUT1	0 - 1

There are two CLUTs for each layer to be able to update while displaying. WM driver uses CULT #0 and CLUT #1 alternately.

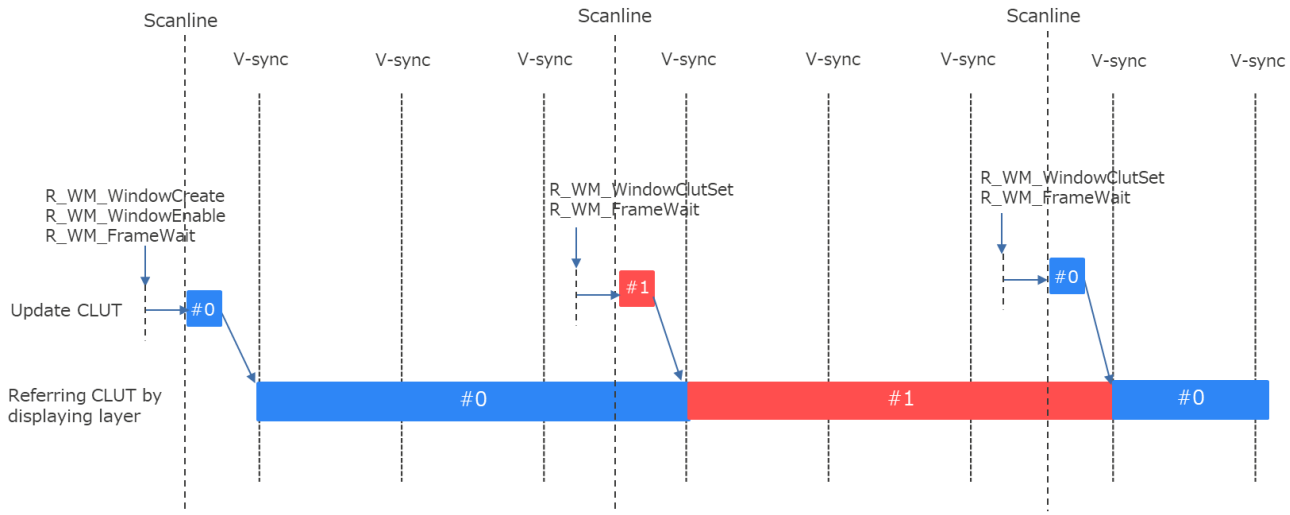


Figure 3-19 CULT update timing

By configuring the frame buffer as shown below, user can switch the display image only by updating CLUT.

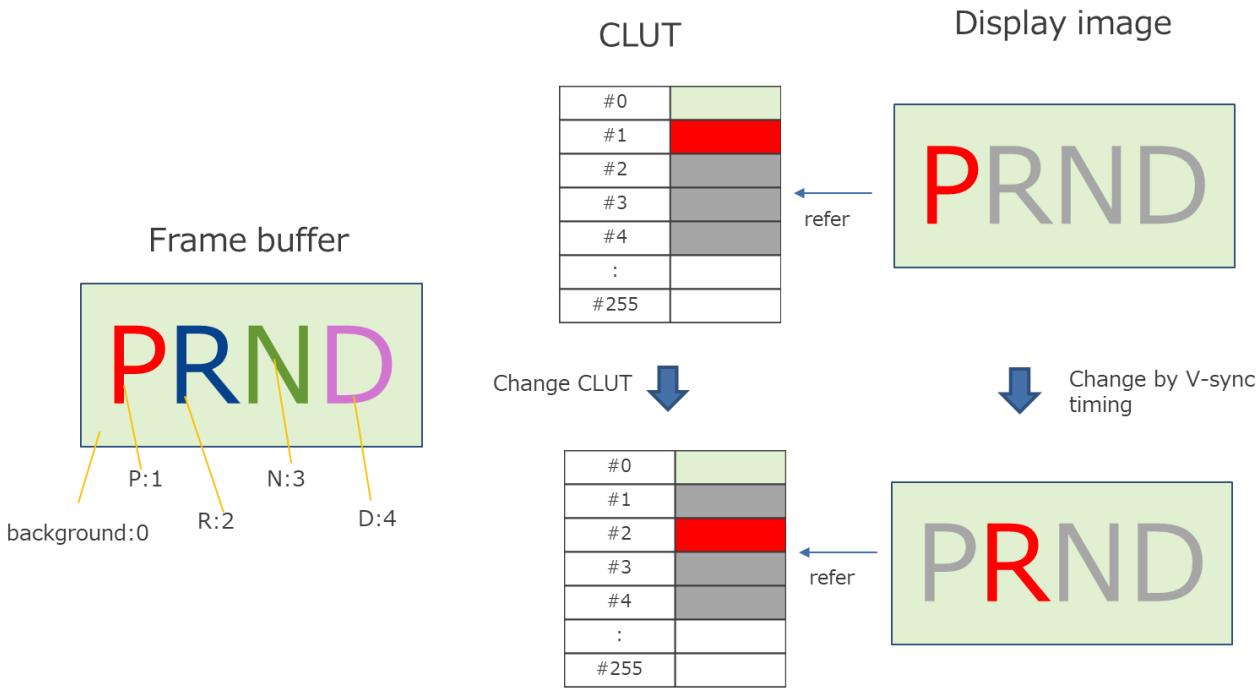


Figure 3-20 CULT use case

Example

```
static r_wm_Window_t Window;
                                      /*  B    G    R    A */
static r_wm_ClutEntry_t ColorTbl[6] = {{ 200, 255, 200, 255}, /*  0 */
                                       {   0,   0, 255, 255}, /*  1 */
                                       { 128, 128, 128, 255}, /*  2 */
                                       { 128, 128, 128, 255}, /*  3 */
                                       { 128, 128, 128, 255}, /*  4 */
                                       {   0,   0,   0,   0}  /*  5 */
};

/* Set default CLUT */
memset (&Window, 0, sizeof (r_wm_Window_t));
Window.Mode                   = R_WM_WINMODE_FB;
Window.Status                 = R_WM_WINSTATUS_NOT_INITIALIZED;
Window.ColorFmt               = R_WM_COLORFMT_CLUT8;
 :
Window.ClutNumEntries         = 6;
Window.Clut                   = ColorTbl;
R_WM_WindowCreate(UNIT_0, &Window);

 :

/* Update CLUT */
ColorTbl[1].B = 128;
ColorTbl[1].G = 128;
ColorTbl[1].R = 128;
ColorTbl[1].A = 255;
ColorTbl[2].B =   0;
ColorTbl[2].G =   0;
ColorTbl[2].R = 255;
ColorTbl[2].A = 255;
R_WM_WindowClutSet(UNIT_0, &Window, 6, ColorTbl);
```

3.2.6 RLE window

3.2.6.1 Creation

WM enables user to fully utilize the RLE feature within frame buffer windows. To do this, one of the available RLE color formats should be selected for a window. The RLE enabled window is like a normal frame buffer window, so:

- There can be any number of RLE data specified within a window as frame buffer.
- They conform to the same swapping rules as the normal frame buffer windows

The buffers for RLE window should be allocated by external mode.

Example

```
static r_wm_Window_t Window;
static r_wm_WinBuffer_t ExBuffer[1];

ExBuffer[0].Status = R_WM_WINBUF_FREE;
ExBuffer[0].Data = rle_image_data; /* Address of RLE data */

memset (&Window, 0, sizeof (r_wm_Window_t));
Window.Mode                  = R_WM_WINMODE_FB;
Window.Status                = R_WM_WINSTATUS_NOT_INITIALIZED;
Window.ColorFmt              = R_WM_COLORFMT_RLE24ARGB8888;
Window.Alpha                 = 0xFF;
Window.PosX                  = 20;
Window.PosY                  = 10;
Window.PosZ                  = 0;
Window.Width                 = 720;
Window.Height                = 480;
Window.Pitch                 = 768;
Window.ScaledWidth           = 0;
Window.ScaledHeight          = 0;
Window.Surface.Fb.BufMode    = R_WM_WINBUF_ALLOC_EXTERNAL;
Window.Surface.Fb.BufNum     = 1;
Window.Surface.Fb.Buffer     = ExBuffer;
R_WM_WindowCreate(UNIT_0, &Window);
```

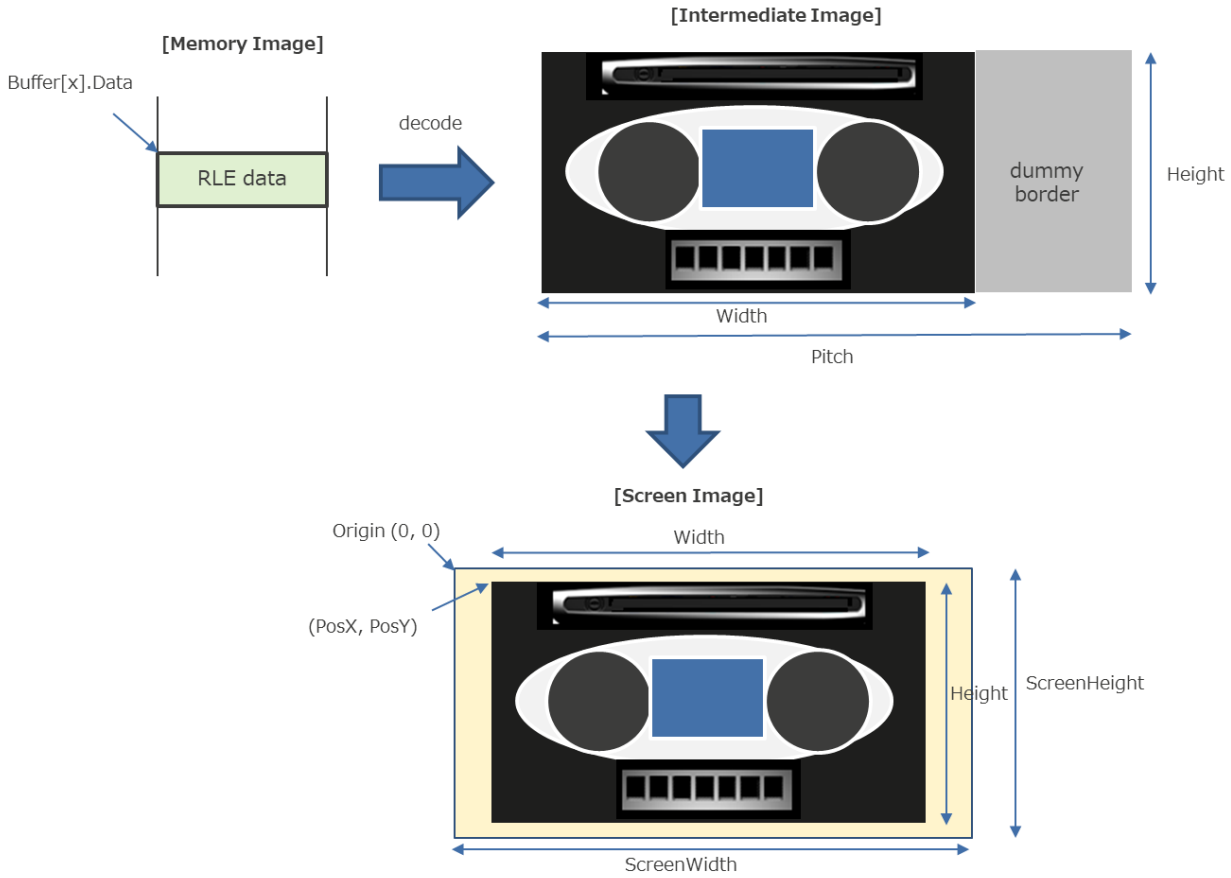


Figure 3-21 Creating window for RLE data

RLE data contains the pitch size which includes the dummy boarder.
Decode buffer for intermediate image is unnecessary.

PosX and PosY should be set to a value greater than or equal to 0 in case of RLE window.
(PosX + Width) should be set within the screen area.
Following table shows the range information for RLE window.

Table 3-26 Parameter range

Parameter	Setting unit	Range		Alignment
		Min	Max	
PosX	Pixel	0	ScreenWidth - Width	-
PosY	Pixel	0	1023	-
Width	Pixel	3	ScreenWidth	-
Height	Pixel	1	1024	-
Pitch	Pixel	Width	8192 and (261120 / bpp)	128 Byte

3.2.7 Sprite window

3.2.7.1 Window Creation

Sprite window is a window hosting Sprite data.

Example

```
static r_wm_Window_t Window;

memset (&Window, 0, sizeof (r_wm_Window_t));
Window.Mode                  = R_WM_WINMODE_SPRITES;
Window.Status                = R_WM_WINSTATUS_NOT_INITIALIZED;
Window.ColorFmt              = R_WM_COLORFMT_ARGB8888;
Window.Alpha                 = 0xFF;
Window.PosX                  = 40;
Window.PosY                  = 10;
Window.PosZ                  = 10;
Window.Width                 = 240;
Window.Height                = 200;
Window.Pitch                 = 2048; /* 8192(Byte) / 4 (Byte per Pixel) */
Window.ScaledWidth           = 0;
Window.ScaledHeight          = 0;
Window.Surface.SpritesRoot   = R_NULL;
R_WM_WindowCreate(UNIT_0, &Window);
```

Following table shows the range information for Sprite window.
Pitch for Sprite window is fixed at 8192 Bytes. WM calculates automatically from bpp of window color format. User can set any Pitch value that meets the following conditions in R_WM_WindowCreate.

Table 3-27 Parameter range

Parameter	Setting unit	Range		Alignment
		Min	Max	
PosX	Pixel	-1280	1279	-
PosY	Pixel	-1024	1023	-
Width	Pixel	3	1280	-
Height	Pixel	1	1024	-
Pitch	Pixel	Width	8192	128 Byte

Sprite window are shared between the layers of the two WM units as following table.
This should be kept in mind when using two displays simultaneously, because some unintentional Sprite occurrences can happen.
It is recommended to use the Sprite window exclusively with the shared layer.
e.g. If Sprite window is assigned to Layer 1 of WM Unit 0, then Frame buffer window is assigned to Layer 3 of WM Unit 1.

Table 3-28 Layer sharing

Sprite window	WM Unit 0	WM Unit 1
1st Sprite window	Layer 1	Layer 3
2nd Sprite window	Layer 2	Layer 2
3rd Sprite window	Layer 3	Layer 1
4th Sprite window	Layer 0	Layer 0

3.2.7.2 Sprite data Creation

The function R_WM_SpriteCreate creates the sprite data on the Sprite window, with the parameters specified in the user-maintained (in terms of storage persistency) Sprite structure. The Sprite structure is a reference parameter, which is used driver internally until R_WM_SpriteDelete or R_WM_WindowDeleteAllSprites is called. Make sure that Sprite structure is not allocated as local variable that may be discarded while the Sprite data is still in use.

Example

```
static r_wm_Sprite_t Sprite;

memset (&Sprite, 0, sizeof (r_wm_Sprite_t));
sprite.Data   = (void*)sprite_data; /* Address of Sprite data */
sprite.PosX   = 160;
sprite.PosY   = 80;
sprite.PosZ   = 0;
sprite.Width  = 50;
sprite.Height = 30;
sprite.Window = &Window; /* Created sprite window */
R_WM_SpriteCreate(UNIT_0, &Sprite);
R_WM_SpriteEnable(UNIT_0, &Sprite);
```

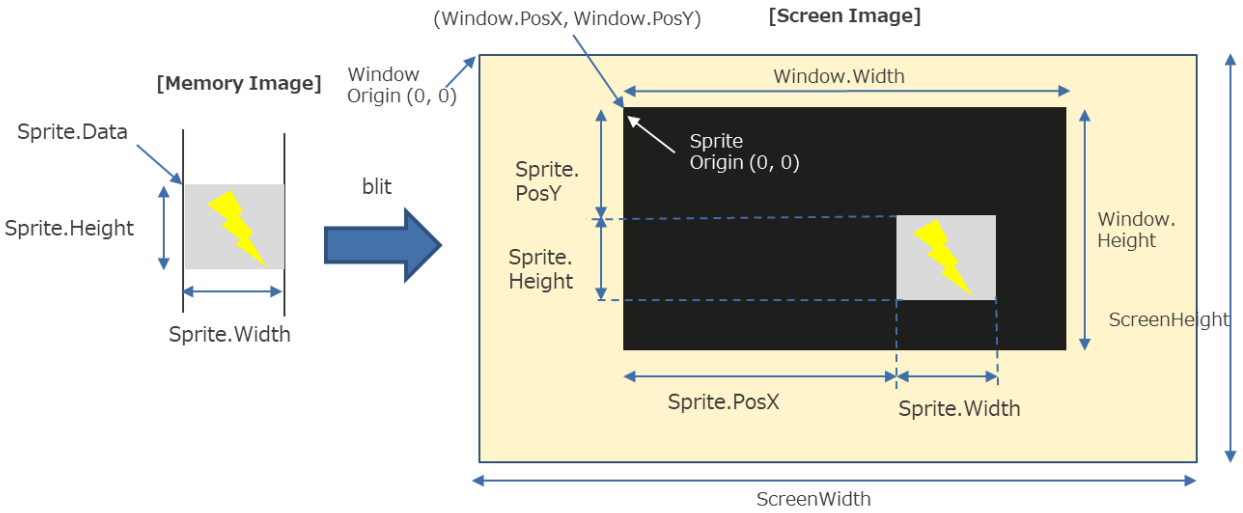


Figure 3-22 Sprite window and Sprite data

Following table shows the range information for Sprite data.

Table 3-29 Parameter range

Parameter	Setting unit	Range		Alignment
		Min	Max	
Data	Byte	-	-	8 Byte
PosX	Pixel	0	(8184 * 8) / bpp	(8 * 8) / bpp
Width	Pixel	0	(8184 * 8) / bpp	(8 * 8) / bpp
(PosX + Width)	Pixel	0	(8192 * 8) / bpp	(8 * 8) / bpp
PosY	Pixel	0	4095	-
Height	Pixel	0	2047	-
(PosY + Height)	Pixel	0	4096	-

3.2.7.3 Sprite data visualization

Sprite data is visualized by R_WM_SpriteEnable.
Up to 32 Sprite data can created per window and up to 16 Sprite data can be visualized per sprite window. If more than 16 sprite data are enabled, Sprite data with larger PosZ takes precedence.
When sharing one Sprite window with two WM units, WM unit 0 takes precedence over PosZ value.

For example:
20 sprite data (PosZ=1~20) are created in Unit 0 Layer1.

Layer	WM Unit 0 Layer1																			
PosZ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Enable Control	Invalid				Valid															

10 sprite data (PosZ=11~20) are created in Unit 0 Layer1 and 10 sprite data are (PosZ=1~10) created in Unit 1 Layer3.

Layer	WM Unit 1 Layer3										WM Unit 0 Layer1									
PosZ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Enable Control	Invalid				Valid						Valid									

10 sprite data (PosZ=1~10) are created in Unit 0 Layer1 and 10 sprite data are (PosZ=11~20) created in Unit 1 Layer3.

Layer	WM Unit 0 Layer1										WM Unit 1 Layer3									
PosZ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Enable Control	Valid										Invalid				Valid					

3.2.7.4 Sprite color format

RGB type is recommended for the color format of Sprite window. VDCE Image synthesizer read data with value 0 if areas of the sprite window without enabled sprite data. So, these data are processed as fully transparent color to the Image synthesizer if color format is RGB format with alpha value (ARGB8888 / ARGB4444 / ARGB1555 / RGBA5551 / RGBA8888).

Note that Layer0 does not have alpha blending feature. Thus, areas of the sprite window without enabled sprite data looks black if Layer0 is used and RGB format is selected.

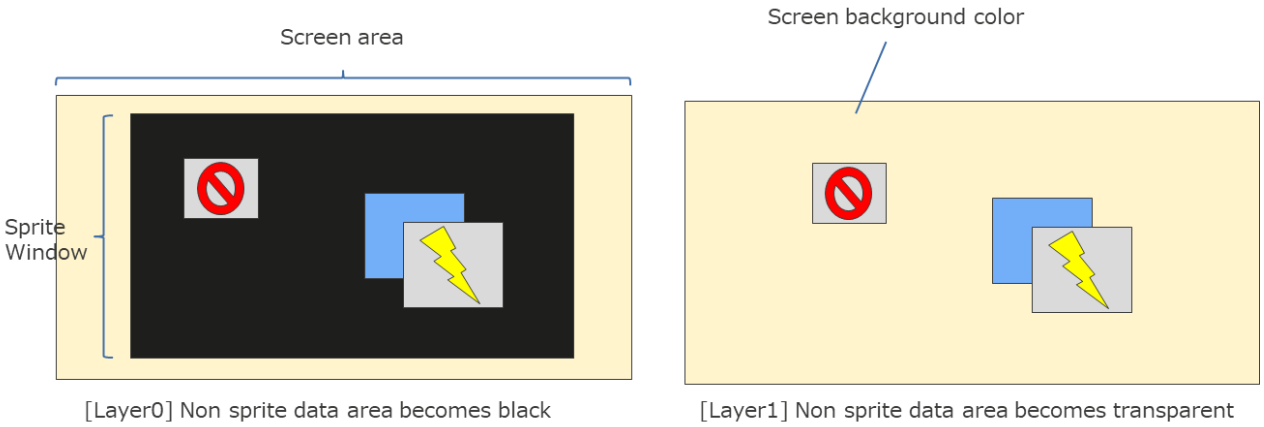


Figure 3-23 No sprite data area

3.2.8 Video input capture

3.2.8.1 Capture control

Capturing video input works always with an existing window. The window needs to fulfill the prerequisites listed in the r_wm_Capture_t structure during the call to R_WM_CaptureCreate. The video input will not be visible on the window surface until enabling it with R_WM_CaptureEnable. The R_WM_CaptureDisable call stops (“freezes”) the video input with the last frame received displayed on the window, until eventually enabled again with R_WM_CaptureEnable. The visibility control of the input stream is delegated to the associated window (R_WM_WindowEnable/Disable).

3.2.8.2 Input video format

The input video format is selected by R_WM_CaptureCreate.

Table 3-30 Input video format

Input video format	Set flag
BT656	R_WM_CAPMODE_YUV_ITU656
BT601	R_WM_CAPMODE_YUV_8BIT
YCbCr422	R_WM_CAPMODE_YUV_16BIT
RGB565	R_WM_CAPMODE_RGB_16BPP
RGB666	R_WM_CAPMODE_RGB_18BPP
RGB888	R_WM_CAPMODE_RGB_24BPP

24 color data signals (DV_DATA23..00) are input to VDCE. Data signal assignment is depending on above input video format and endian.
The endian is set by R_WM_CAPMODE_BIG_ENDIAN flag in R_WM_CaptureCreate.

Table 3-31 Input color data

Setting		DV_DATA23 <---> DV_DATA00
Input video format	Endian	
RGB888	Little	$R_7R_6R_5R_4R_3R_2R_1R_0$ $G_7G_6G_5G_4G_3G_2G_1G_0$ $B_7B_6B_5B_4B_3B_2B_1B_0$
	Big	$R_0R_1R_2R_3R_4R_5R_6R_7$ $G_0G_1G_2G_3G_4G_5G_6G_7$ $B_0B_1B_2B_3B_4B_5B_6B_7$
RGB666	Little	0 0 0 0 0 0 R_7R_6 $R_5R_4R_3R_2G_7G_6G_5G_4$ $G_3G_2B_7B_6B_5B_4B_3B_2$
	Big	0 0 0 0 0 0 R_2R_3 $R_4R_5R_6R_7G_2G_3G_4G_5$ $G_6G_7B_2B_3B_4B_5B_6B_7$
RGB565	Little	0 0 0 0 0 0 0 0 $R_7R_6R_5R_4R_3G_7G_6G_5$ $G_4G_3G_2B_7B_6B_5B_4B_3$
	Big	0 0 0 0 0 0 0 0 $R_3R_4R_5R_6R_7G_2G_3G_4$ $G_5G_6G_7B_3B_4B_5B_6B_7$
YCbCr422	Little	0 0 0 0 0 0 0 0 $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$ $C_7C_6C_5C_4C_3C_2C_1C_0$
	Big	0 0 0 0 0 0 0 0 $Y_0Y_1Y_2Y_3Y_4Y_5Y_6Y_7$ $C_0C_1C_2C_3C_4C_5C_6C_7$
BT656 / BT601	Little	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $T_7T_6T_5T_4T_3T_2T_1T_0$
	Big	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 $T_0T_1T_2T_3T_4T_5T_6T_7$

R_n, G_n and B_n (n=0~7): RGB data.
Y_n: Y component
C_n: Cb or Cr component is input alternately.
T_n: BT656 / BT601 input data.

3.2.8.3 Capture buffer color format

The color format of the capture buffer is set by ColorFmt of associated window. The following three formats can be set. Because VDCE has color matrix for input data, any input video format can convert to RGB format.

- R_WM_COLORFMT_RGB0888
- R_WM_COLORFMT_ARGB8888 (Alpha value is fixed at 0xFF.)
- R_WM_COLORFMT_RGB565

3.2.8.4 Capture surface creation

The capture surface is created by R_WM_CaptureCreate.
Example of creating capture surface and capture window with same magnification is shown below.

Example

```
static r_wm_Window_t   Window;
static r_wm_Bufffer_t  Buffer[3] = {0};
static r_wm_Capture_t  Capt;

memset(&Window, 0, sizeof(r_wm_Window_t));
Window.Mode              = R_WM_WINMODE_FB;
Window.Status            = R_WM_WINSTATUS_NOT_INITIALIZED;
Window.ColorFmt          = R_WM_COLORFMT_ARGB8888;
Window.Alpha             = 0xFF;
Window.PosX              = 60;
Window.PosY              = 40;
Window.PosZ              = 0;
Window.Width             = 720;
Window.Height            = 480;
Window.Pitch             = 736;
Window.ScaledWidth       = 0;
Window.ScaledHeight      = 0;
Window.Surface.Fb.BufMode= R_WM_WINBUF_ALOC_EXTERNAL;
Window.Surface.Fb.BufNum = 3;
Window.Surface.Fb.Buffer = Buffer;
Buffer[0].Data      = (void*)0x40000000;
Buffer[1].Data      = (void*)0x40200000;
Buffer[2].Data      = (void*)0x40400000;
R_WM_WindowCreate(UNIT_0, &Window);
R_WM_WindowEnable(UNIT_0, &Window);

memset(&Capt, 0, sizeof(r_wm_Capture_t));
Capt.Window              = &Window;
Capt.Mode                = R_WM_CAPMODE_RGB_18BPP|R_WM_CAPMODE_VSYNC_INVERT|
                           R_WM_CAPMODE_HSYNC_INVERT;
Capt.StartX              = 16;
Capt.StrideX             = Window.Pitch; /* It should be same as Window Pitch */
Capt.StartY1             = 24;
Capt.StartY2             = 0;
Capt.Width               = 720;
Capt.Height              = 480;
Capt.ScaledWidth         = 0;
Capt.ScaledHeight        = 0;
Capt.Delay               = 0;
Capt.CapUnit             = UNIT_0;
Capt.Next                = R_NULL;
R_WM_CaptureCreate(UNIT_0, &Capt);
R_WM_CaptureEnable(UNIT_0, &Capt);
```

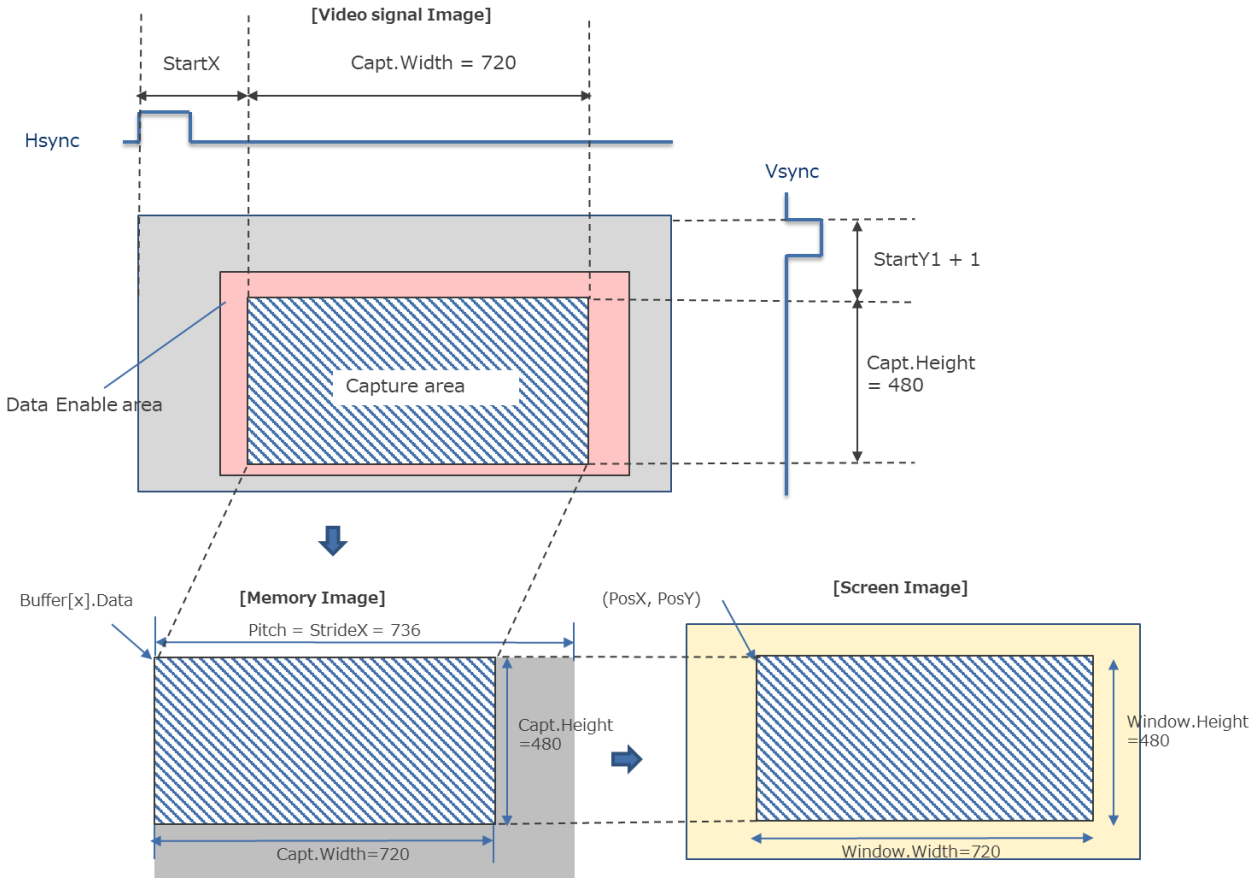


Figure 3-24 Capture with Hsync

Following table shows the range information.

Table 3-32 Parameter range of R_WM_CaptureCreate (same magnification)

Parameter	Setting unit	Range		Alignment
		Min	Max	
StrideX	Pixel	1024 / bpp	261120 / bpp	128 Byte
StartX	Pixel	16	2011	-
StartY1	Pixel	4	2035	-
Width	Pixel	4	1024	4 Pixel
Height	Pixel	4	1024	4 Pixel
StartX + Width	Pixel	20	2015	-
StartY1 + Height	Pixel	8	2039	-
Delay	Line	0	255	-

* bpp is depending on color format of capture buffer.

3.2.8.5 Scaling-down

Scaling-down feature can be available for capture data.
ScaledWidth and ScaledHeight in r_wm_Capture_t is specified the target memory size to be reduced.
Example for creating capture surface and window when 720x480 capture signal is reduced to 360x320.

Example

```
static r_wm_Window_t   Window;
static r_wm_Capture_t  Capt;
:
Window.Width              = 360;
Window.Height             = 320;
Window.Pitch              = 384;
Window.ScaledWidth        = 0;
Window.ScaledHeight       = 0;
:
R_WM_WindowCreate(UNIT_0, &Window);
R_WM_WindowEnable(UNIT_0, &Window);

:
Capt.StrideX              = Window.Pitch; /* It should be same as Window Pitch */
Capt.Width                = 720;
Capt.Height               = 480;
Capt.ScaledWidth          = 360;
Capt.ScaledHeight         = 320;
:
R_WM_CaptureCreate(UNIT_0, &Capt);
R_WM_CaptureEnable(UNIT_0, &Capt);
```

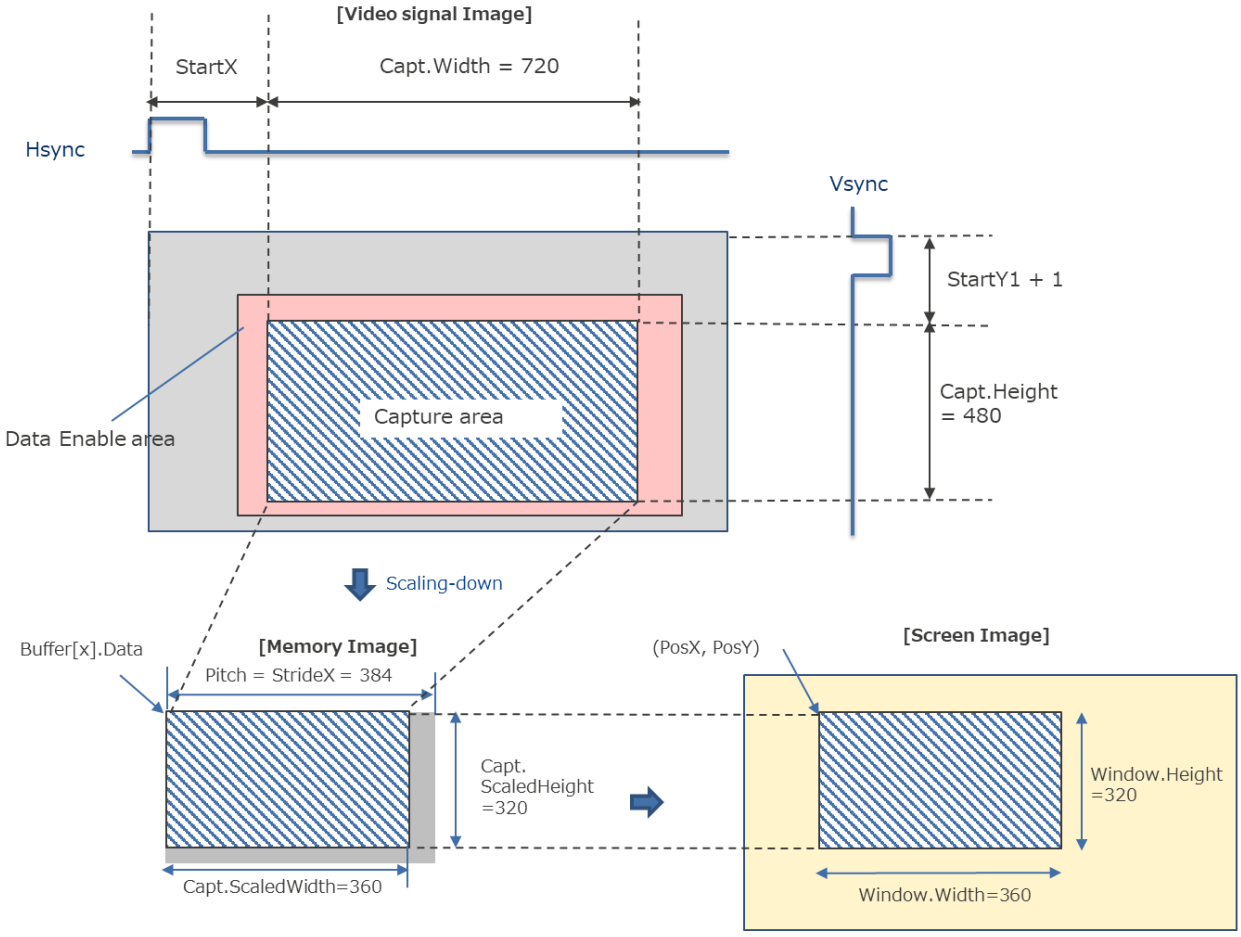


Figure 3-25 Capture with scaling-down

Following table shows the range information. The range of other parameters are same as Table 3-32.

Table 3-33 Parameter range of R_WM_CaptureCreate (scaling-down)

Parameter	Setting unit	Range		Alignment
		Min	Max	
Width	Pixel	8	2008	4 Pixel
Height	Pixel	8	1024	4 Pixel
StartX + Width	Pixel	24	2015	-
StartY1 + Height	Pixel	12	2039	-
ScaledWidth	Pixel	4	1024	4 Pixel
ScaledHeight	Pixel	4	1020	4 Pixel

Note:
When horizontal scaling-down is enabled, horizontal scaling-up cannot work.
When vertical scaling-down is enabled, vertical scaling-up cannot work.

3.2.8.6 Scaling-up

Scaling-up is realized by the Window feature.
In this case, ScaliedWidth and ScaledHeight in r_wm_Window_t should be adjusted to the capture size.

Example

```
static r_wm_Window_t   Window;
static r_wm_Capture_t  Capt;

:
Window.Width              = 720;
Window.Height             = 480;
Window.Pitch              = 384;
Window.ScaledWidth        = 360;
Window.ScaledHeight       = 320;
:
R_WM_WindowCreate(UNIT_0, &Window);
R_WM_WindowEnable(UNIT_0, &Window);

:
Capt.StrideX              = Window.Pitch; /* It should be same as Window Pitch */
Capt.Width                = 360;
Capt.Height               = 320;
Capt.ScaledWidth          = 0;
Capt.ScaledHeight         = 0;
:
R_WM_CaptureCreate(UNIT_0, &Capt);
R_WM_CaptureEnable(UNIT_0, &Capt);
```

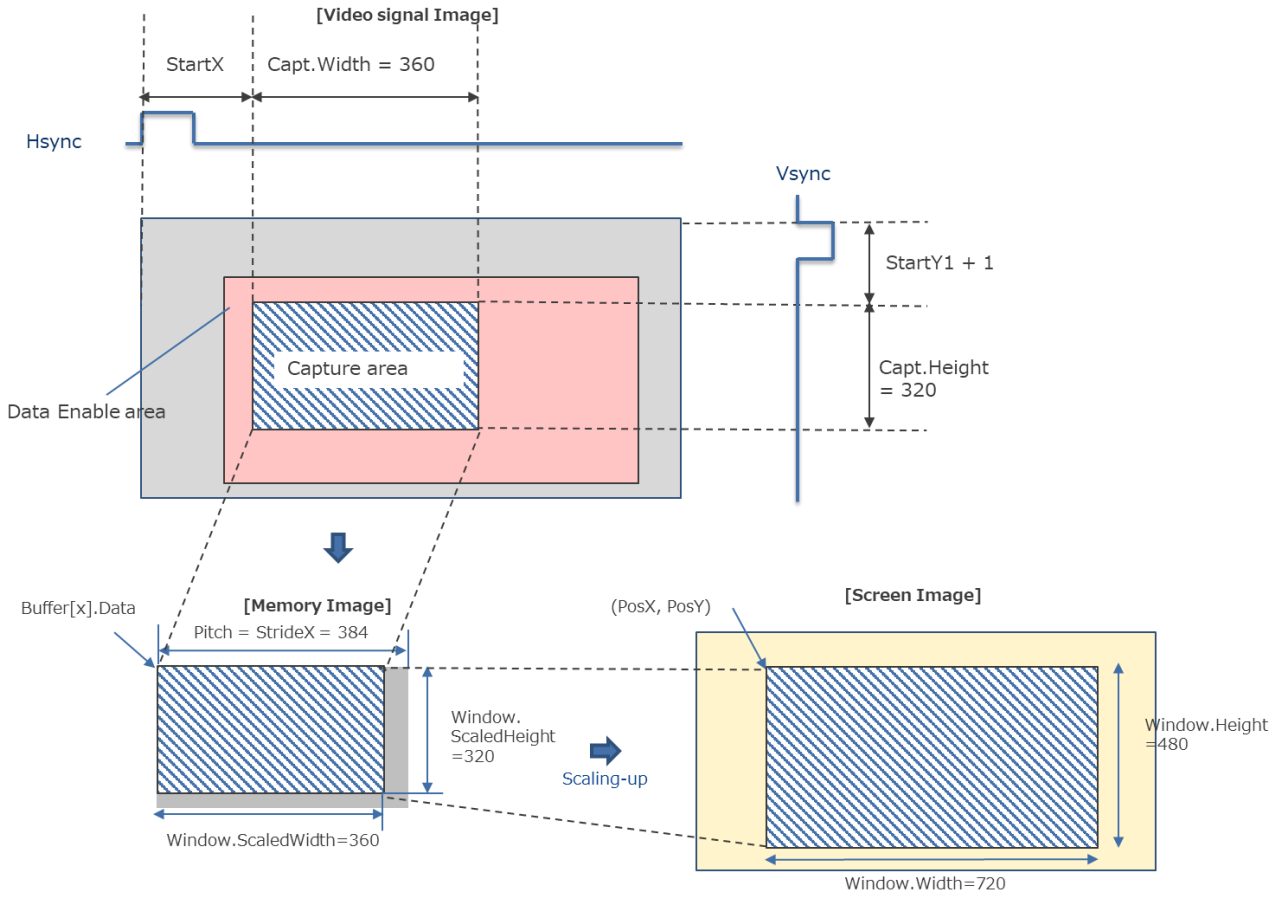


Figure 3-26 Display with scaling-up

3.2.8.7 Capture with DE mode

If DE mode is enabled by R_WM_CAPMODE_DE_MODE flag, video signal is captured by DE signal instead of Hsync signal. In case of DE mode, StartX and StartY1 can be set 0 to capture all data enable area.

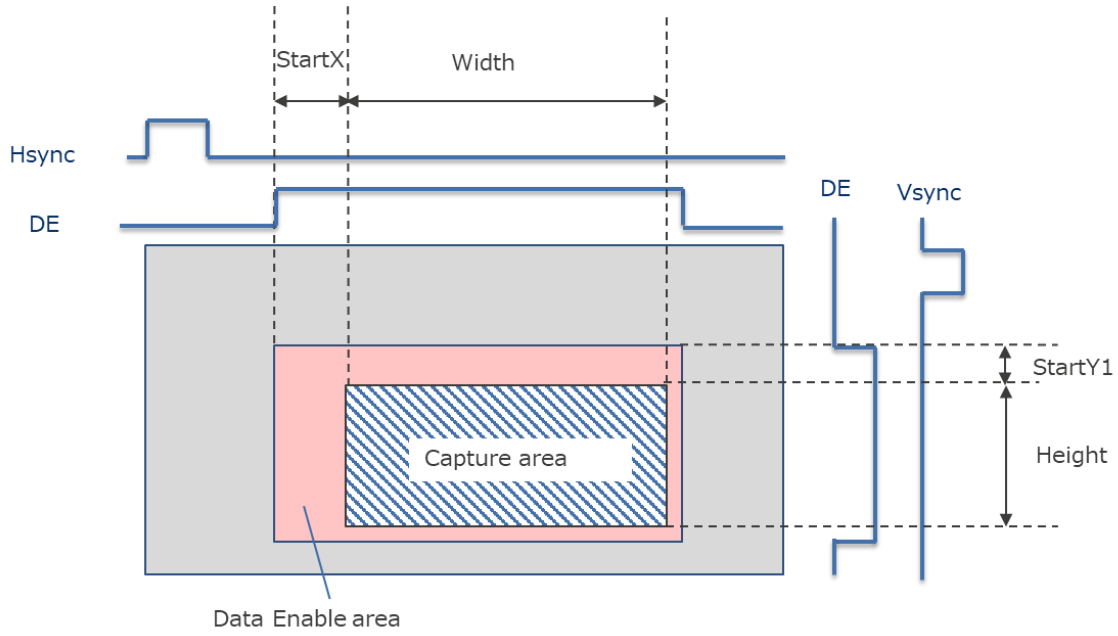


Figure 3-27 Capture area with DE signal

Following table shows the range information. The range of other parameters are same as Table 3-32.

Table 3-34 Parameter range of DE mode

Parameter	Setting unit	Range		Alignment
		Min	Max	
StartX	Pixel	0	2011	-
StartY1	Pixel	0	2035	-
StartX + Width	Pixel	4	2015	-
StartY1 + Height	Pixel	4	2039	-

3.2.8.8 Capture interlace signal

If video input format is BT656 or BT601, VDCE can capture the interlace signal.
If video input format is BT656, VDCE generates Hsync signal from SAV/EAV code.
See H/W user's manual for the detail.

In case of BT656 or BT601 format, each pixel is captured twice. Therefore, Width setting should be doubled and horizontal size should be scaled down by half in order to keep original input size.

This is the sample code to capture 720x480 interlace signal and display the image with 720x480 with BOB de-interlace mode.

Example

```
static r_wm_Window_t   Window;
static r_wm_Capture_t  Capt;
:
Window.Width             = 720;
Window.Height            = 480;
Window.Pitch             = 736;
Window.ScaledWidth       = 0;
Window.ScaledHeight      = 240;
:
R_WM_WindowCreate(UNIT_0, &Window);
R_WM_WindowEnable(UNIT_0, &Window);

:
Capt.Mode                = R_WM_CAPMODE_YUV_ITU656
                         | R_WM_CAPMODE_BOB_TOP
                         | R_WM_CAPMODE_YUV_Y_UV_INVERT
                         | R_WM_CAPMODE_DATA_CLK_INVERT;
Capt.StartX              = 16;
Capt.StrideX             = Window.Pitch; /* It should be same as Window Pitch */
Capt.StartY1             = 20;
Capt.Width               = 720*2;
Capt.Height              = 240;
Capt.ScaledWidth         = 720;
Capt.ScaledHeight        = 0;
:
R_WM_CaptureCreate(UNIT_0, &Capt);
R_WM_CaptureEnable(UNIT_0, &Capt);
```

BOB de-interlace mode captures only one field and displays it with scaling-up.

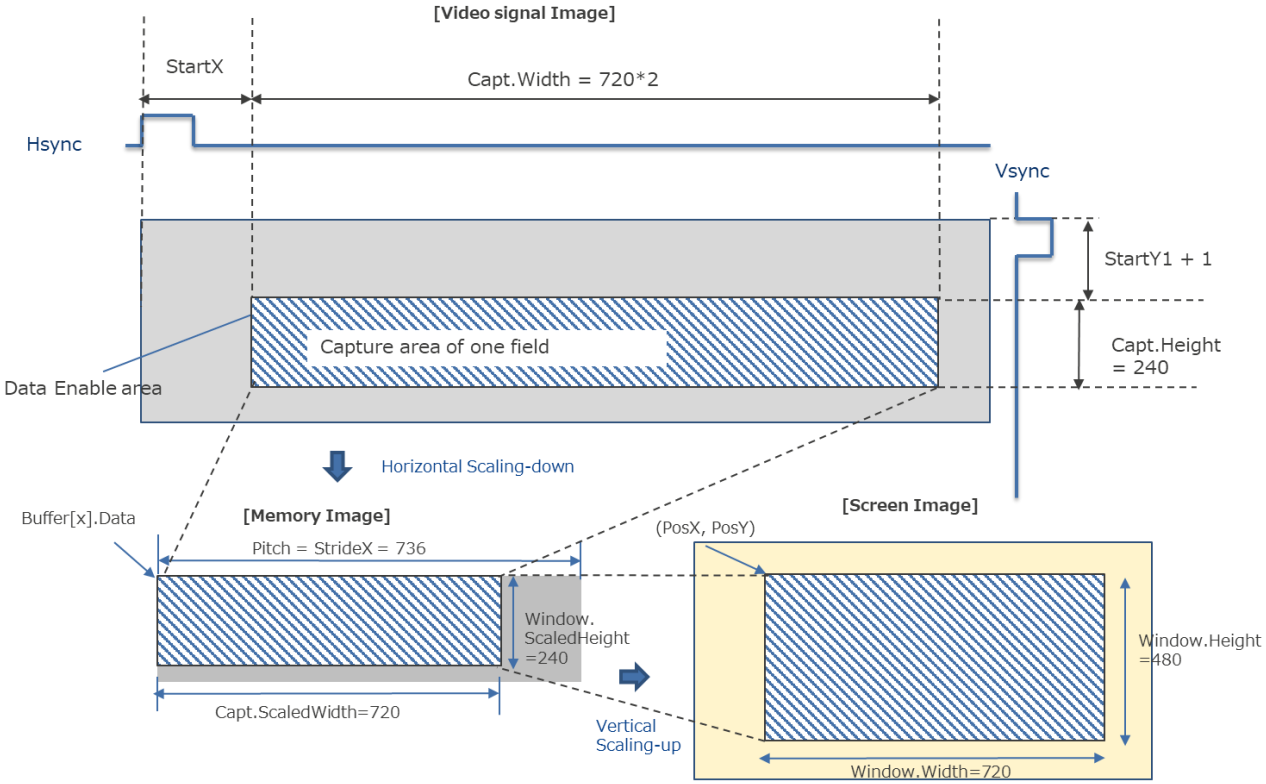


Figure 3-28 Deinterlace with BOB mode

This is the sample code to capture 720x480 interlace signal and display the image with 720x480 with WEAVE de-interlace mode.
WEAVE de-interlace mode corresponds only when the number of frame buffer is 1.

Example

```
static r_wm_Window_t   Window;
static r_wm_Capture_t  Capt;
static r_wm_Bufffer_t  Buffer[1] = {0};

:
Window.Width             = 720;
Window.Height            = 480;
Window.Pitch             = 736;
Window.ScaledWidth       = 0;
Window.ScaledHeight      = 0;
Window.Surface.Fb.BufMode= R_WM_WINBUF_ALOC_EXTERNAL;
Window.Surface.Fb.BufNum = 1;   /* Only single buffer is supported. */
Window.Surface.Fb.Buffer = Buffer;
Buffer[0].Data           = (void*)0x40000000;
:
R_WM_WindowCreate(UNIT_0, &Window);
R_WM_WindowEnable(UNIT_0, &Window);

:
Capt.Mode                = R_WM_CAPMODE_YUV_ITU656 |
                           R_WM_CAPMODE_WEAVE |
                           R_WM_CAPMODE_YUV_Y_UV_INVERT |
                           R_WM_CAPMODE_DATA_CLK_INVERT;
Capt.StartX              = 16;
Capt.StrideX             = Window.Pitch; /* It should be same as Window Pitch */
Capt.StartY1             = 20;
Capt.Width               = 720*2;
Capt.Height              = 240;
Capt.ScaledWidth         = 720;
Capt.ScaledHeight        = 0;
:
R_WM_CaptureCreate(UNIT_0, &Capt);
R_WM_CaptureEnable(UNIT_0, &Capt);
```

WEAVE de-interlace mode captures both top and bottom fields and displays with weaving.

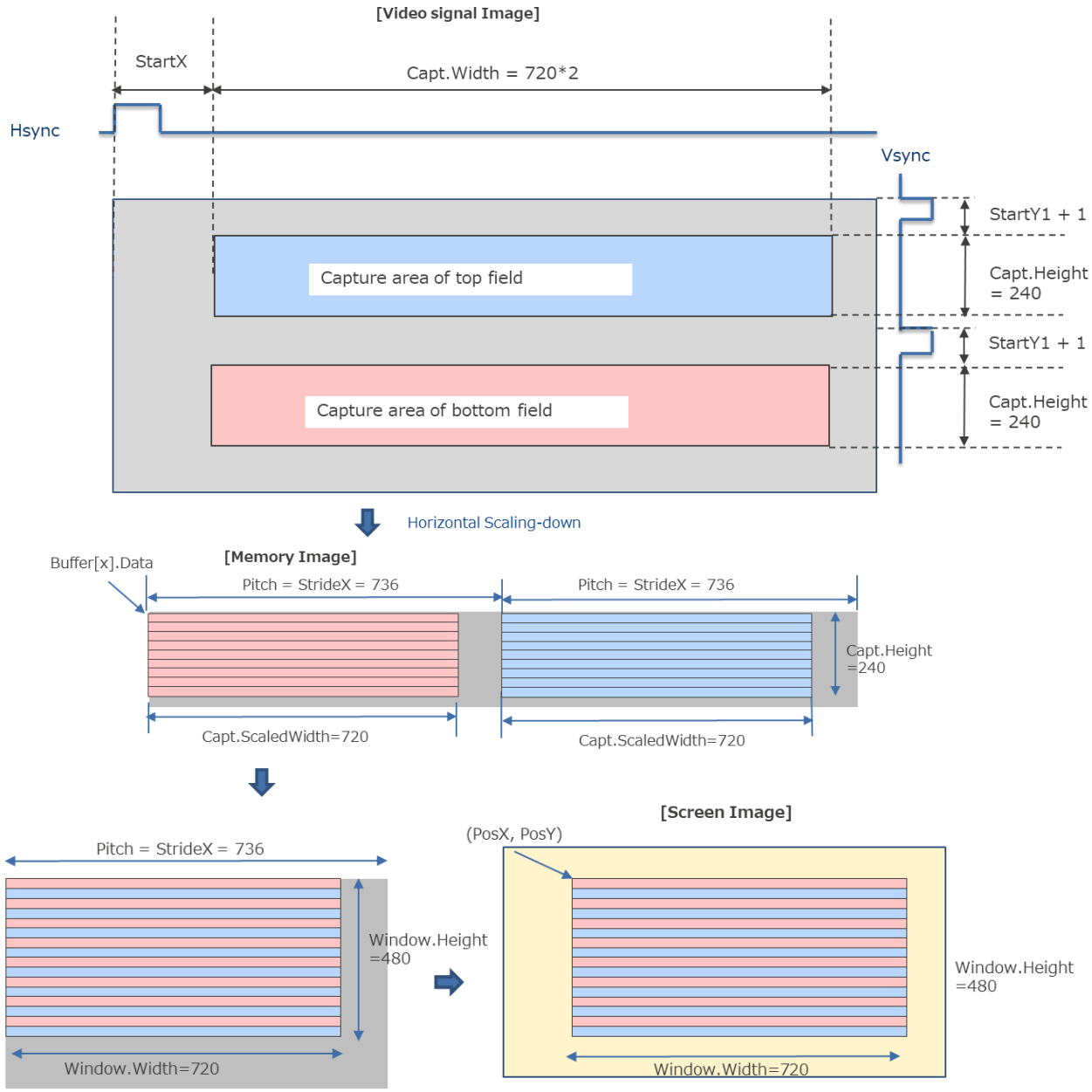


Figure 3-29 WEAVE de-interlace mode

3.2.8.9 Capture buffer State transition

Following figure shows frame buffer status transition when capture is enabled, and 3 frame buffers or more are used.

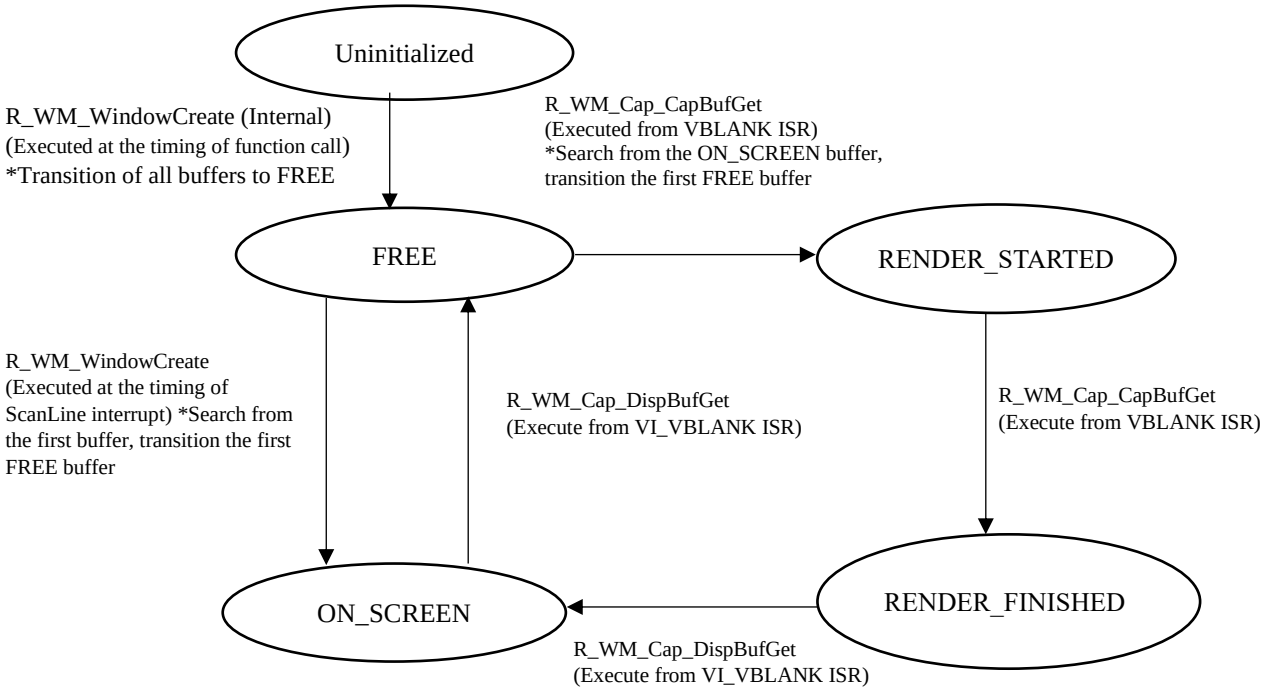


Figure 3-30 Buffer State Transition (Capture Input)

State diagram (3 or more buffers)

When Capture input is specified and the number of frame buffer is three or more, WM automatically swaps the buffer.

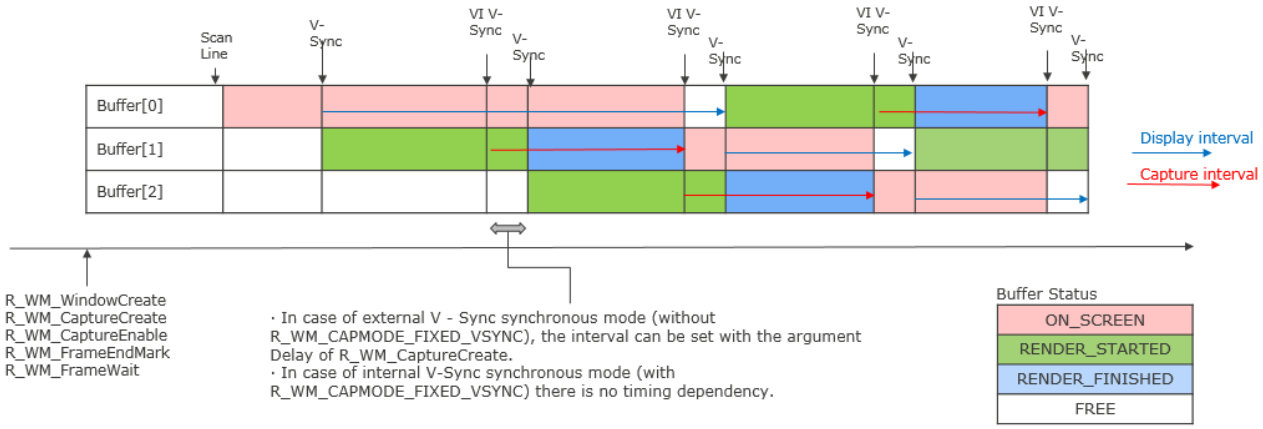


Figure 3-31 State diagram of triple buffers

State diagram (1 buffer)

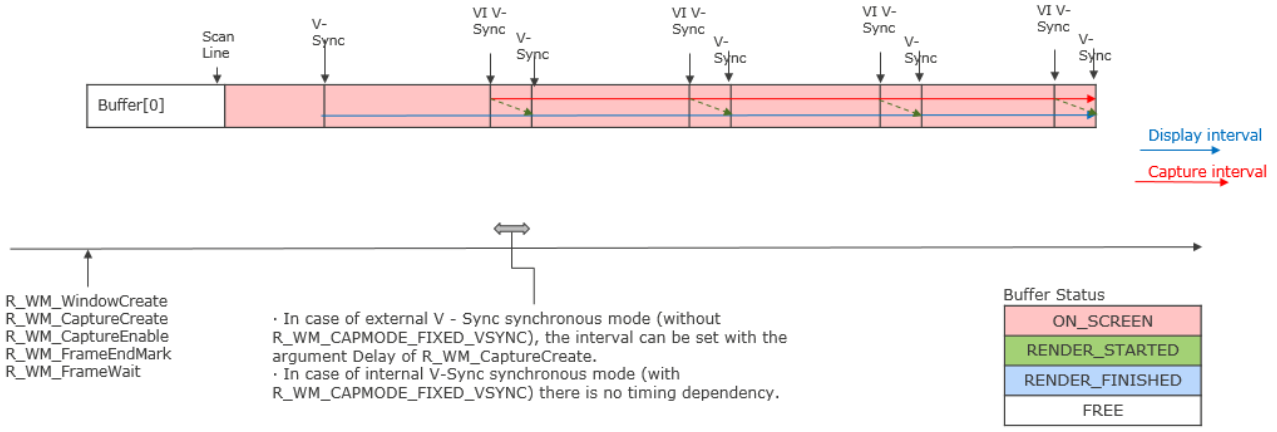


Figure 3-32 State diagram of single buffer

3.2.9 H/W update

WM requests are almost always executed asynchronously using the message queue: User issues a request without waiting for it to be executed. For executing these requests and being notified about the changes being propagated to the H/W (usually being accompanied by visual changes on the display), there are two approaches that can be distinguished: synchronous and asynchronous.

3.2.9.1 Synchronous H/W update

This is the case where user triggers the requests execution (i.e. the H/W update process) and waits in a blocking fashion for the H/W changes to be propagated and done. The function R_WM_FrameWait issues the execution of the stored requests up to a delimiter specified with Id, and blocks until all corresponding H/W changes are finished.

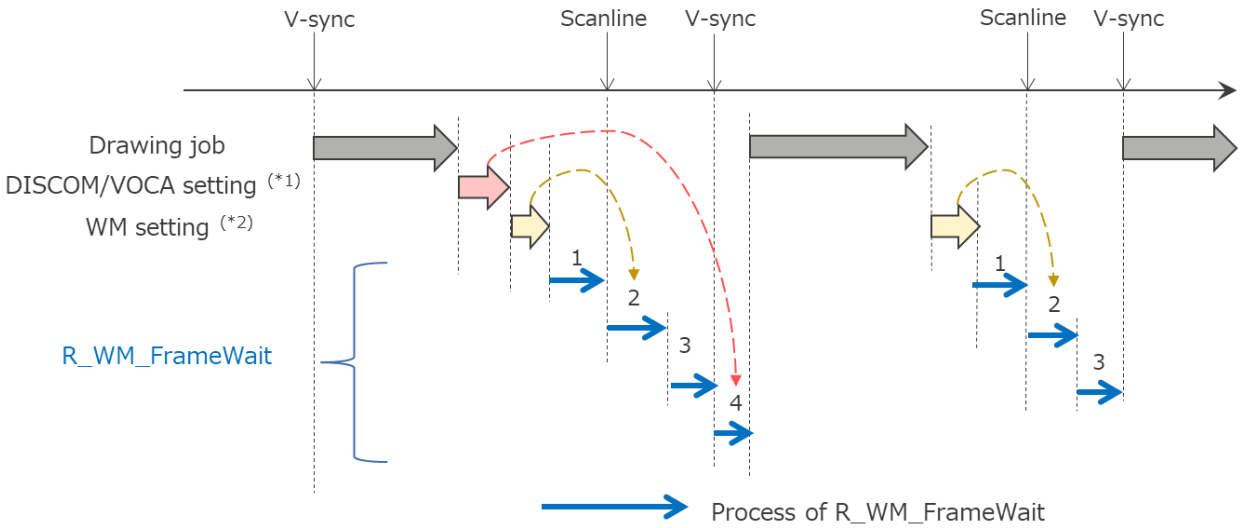


Figure 3-33 Synchronous Update (ideal case)

(*1) It means execution of the following WM APIs.

- R_WM_ScreenVocaExpImgSet
- R_WM_ScreenVocaClutSet
- R_WM_DiscomCrcSet

(*2) It means execution of the WM APIs that use message queues other than the above.

R_WM_FrameWait internal process is as follows.

1. R_WM_FrameWait waits for the scanline interrupt that is appropriate moment to start updating the H/W registers. By default, the scanline is set to a point in time after which the late of the ‘previous’ frame is refreshed. This should leave enough time for the H/W update functions to be executed before next V-sync.

2. Execute jobs in WM queue until ID of specified at R_WM_EndMark.

3. Wait for V-sync interrupt. VDCE register updates are reflected in the in synchronization with V-sync. It is still a good moment for allowing for the preparation of the next frame, because the old framebuffer is completely shown on the screen and it can start being reused by user.

4. Execute DISCOM / VOCA jobs if DISCOM / VOCA updates exist in the WM queue.

User can influence this process by customizing the following porting layer functions.

- R_WM_Sys_DevWaitForHwWriteReady
- R_WM_Sys_WaitForHwUpdated.

When drawing job becomes longer and the execution of R_WM_FrameWait is delayed, R_WM_FrameWait waits the next scanline timing to update registers. Thus, the drawing job will be skipped for one V-sync period.

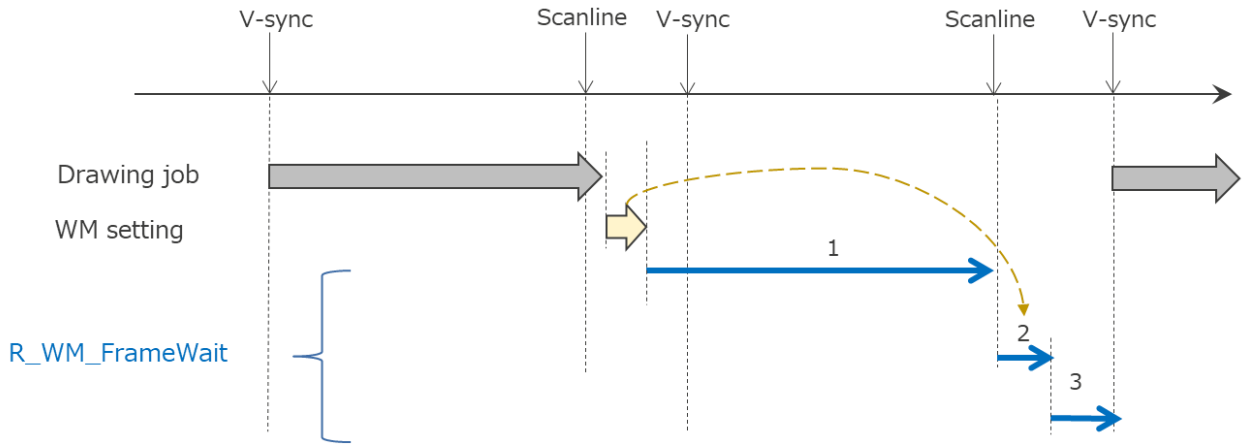


Figure 3-34 Synchronous Update (delay case)

3.2.9.2 Synchronous H/W update w/ VOWE

If distortion correction by VOWE is used, display timing will be delayed by the amount specified in the VOWE API. And, VOCA monitoring timing is also delayed accordingly.
So, the update of the VOCA expected value image may not match the frame update and R_WM_EVENT_VOCA_MISMACTH event will be notified for some frames.

3.2.9.3 Asynchronous H/W update

In the absence of R_WM_FrameWait, user can execute the stored requests by using R_WM_FrameExecuteNext, which executes the requests up to the next frame delimiter, which is then returned by the function. This function immediately processes the message queue and returns without waiting for any event.
R_WM_FrameExecuteDiscom and R_WM_FrameExecuteVoca are also provided for asynchronous update purpose.
R_WM_FrameExecuteDiscom is prepared for updating expected CRC value set by R_WM_DiscomCrcSet.
R_WM_FrameExecuteVoca is prepared for updating expected image set by R_WM_ScreenVocaExpImgSet and VOCA CLUT data set by R_WM_ScreenVocaClutSet.

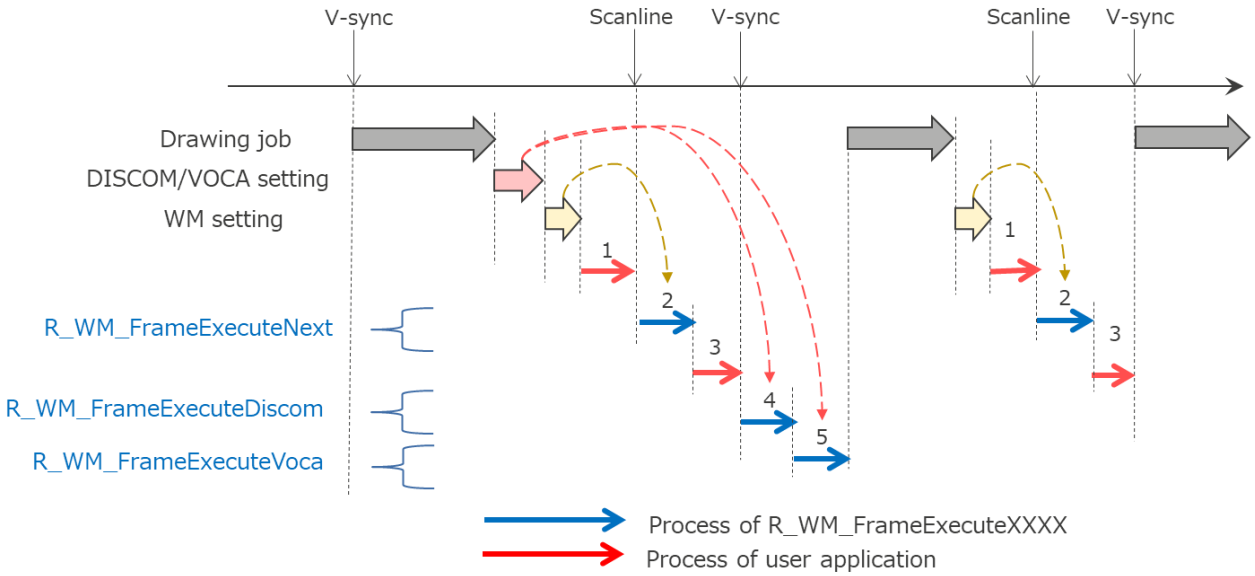


Figure 3-35 Asynchronous Update

The recommended sequence is as follows.

1. Wait for the Scanline interrupt by user.
2. R_WM_FrameExectueNext is called to execute jobs in WM queue until the queue by R_WM_EndMark is found.
3. Wait for V-sync interrupt by user.
4. R_WM_FrameExectueDiscom is called to execute DISCOM jobs in the WM queue.
5. R_WM_FrameExectueVoca is called to execute VOCA jobs in WM queue.

User can setup his own asynchronous update mechanism by registering for the callback of Scanline (R_WM_EVENT_SCANLINE) and V-sync (R_WM_EVENT_VBLANK) interrupts via R_WM_DevEventRegister call. Once user sets up his own Scanline interrupt handler, using R_WM_FrameWait during the same device life-time is prohibited.

3.2.9.4 Asynchronous H/W update w/ VOWE

If distortion correction by VOWE is used, display timing will be delayed by the amount specified in the VOWE API. And, VOCA monitoring timing is also delayed accordingly.
Using OIR Vsync interrupt (R_WM_EVENT_OIR_VBLANK) will synchronize frame updates and VOCA expected image updates.

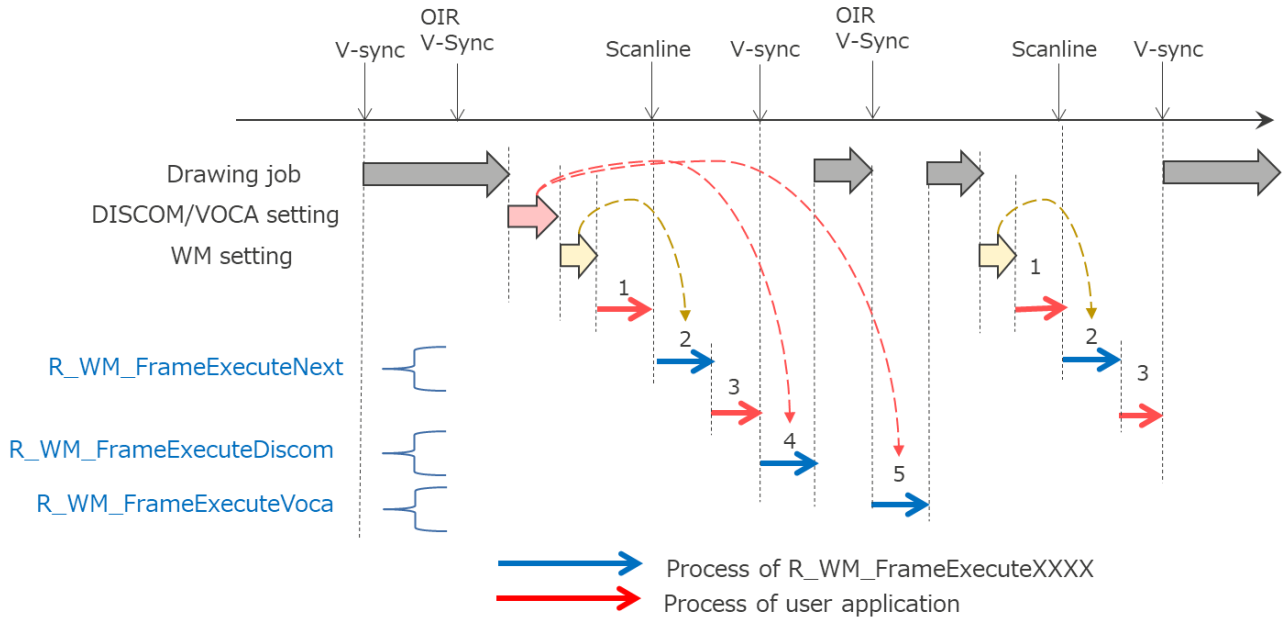


Figure 3-36 Asynchronous Update w/ VOWE

3.2.9.5 Scanline timing

The scanline interrupt timing is set by R_WM_ScreenTimingSet or R_WM_ScreenTimingSetByName as default.

Table 3-35 Default value

Condition	Scanline setting
ScreenHeight > 240	ScreenHeight - 50
ScreenHeight <= 240	ScreenHeight / 10

The scanline timing can be changed with R_WM_DevEventRegister.

Table 3-36 Parameter range

Parameter	Setting unit	Range	
		Min	Max
Arg	Line	0	2047

The scanline interrupt occurs in the range 0 to (V.Total - 1). If it is larger than (V.Total - 1), no scanline interrupt occurs.

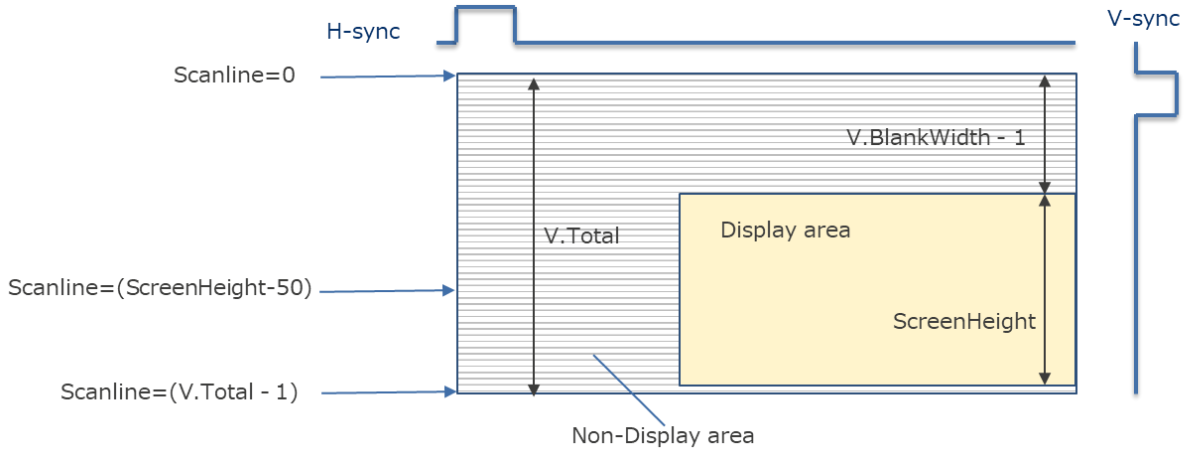


Figure 3-37 Scanline timing

3.2.9.6 Frame ID

R_WM_FrameEndMark and R_WM_FrameWait have the argument 'Id' for synchronization. In a usual use case, it is possible to operate with a fixed value (e.g. Id = 0) as example in *3.2.4.5*.

Here are two examples when multiple 'Id' value is used.

Example1:

```
/* Execute WM API group (x) */
R_WM_xxxx();
R_WM_xxxx();
R_WM_FrameEndMark(LOC_WM_UNIT, 0); /* Id = 0 */

/* Execute WM API group (y) */
R_WM_yyyy();
R_WM_yyyy();
R_WM_FrameEndMark(LOC_WM_UNIT, 1); /* Id = 1 */

R_WM_FrameWait(LOC_WM_UNIT, 0);    /* Id = 0 */
```

R_WM_FrameWait sequence is as follows.
1. Wait for scanline interrupt
2. Execute message queue of WM API group (x)
3. Wait for V-sync interrupt

WM API group (y) remains in the message queue and is processed with the next R_WM_FrameWait.

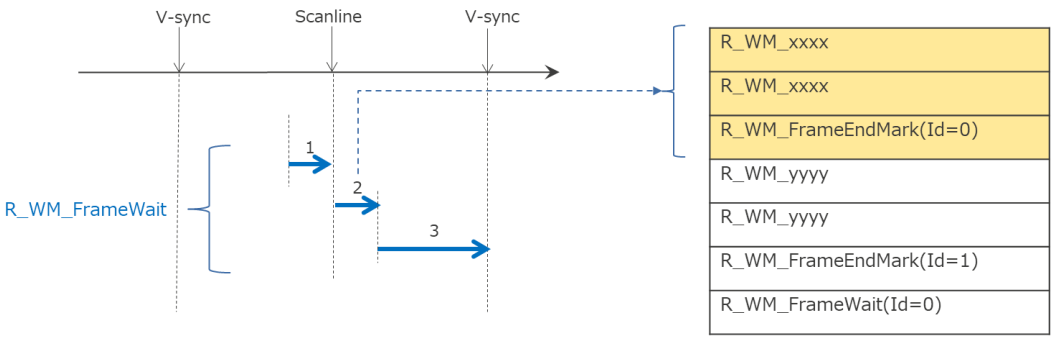


Figure 3-38 Frame ID example 1

Example2:

```
R_WM_xxxx();
R_WM_xxxx();
R_WM_FrameEndMark(LOC_WM_UNIT, 0); /* Id = 0 */

R_WM_yyyy();
R_WM_yyyy();
R_WM_FrameEndMark(LOC_WM_UNIT, 1); /* Id = 1 */

R_WM_FrameWait(LOC_WM_UNIT, 1);    /* Id = 1 */
```

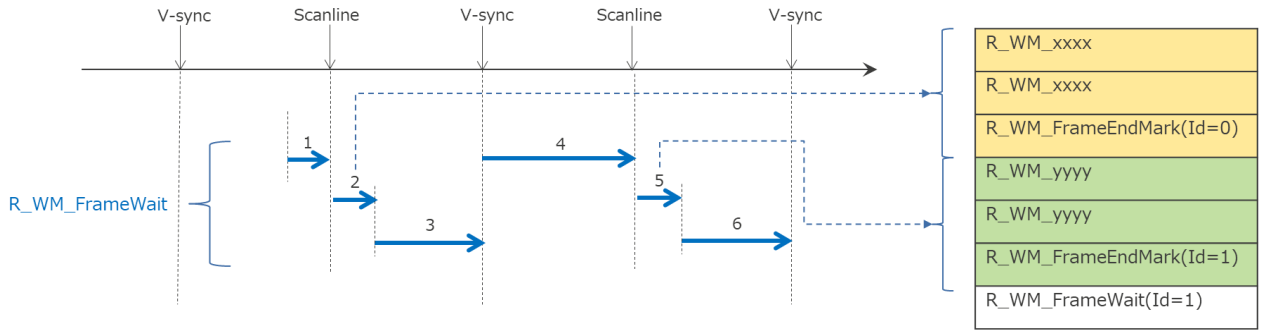


Figure 3-39 Frame ID example 2

R_WM_FrameWait sequence is as follows.

1. Wait for scanline interrupt
2. Execute WM API group (x)
3. Wait for V-sync interrupt
4. Wait for scanline interrupt
5. Execute WM API group (y)
6. Wait for V-sync interrupt

3.2.10 Flowchart and variable lifetimes

This chapter shall highlight the relations of different API layers to variable lifetimes and to display update timings. Below *Table 3-37* shows a waterfall graph on how different variables and queues are synchronized to API calls and Video Output timings.

Table 3-37 Description of the different columns shown in the Figure below.

Column	Explanation
win1, win2	The variables “win1” and “win2” of type r_wm_Window_t store the window configuration. The user application must allocate these variables. This column shows when the variables are in use by the application and by the WM driver. First, they are “used” by the application for initialization. After R_WM_WindowCreate, they are in use by the WM driver. They are “locked” and must not be written by the user-application.
APP	This is the extract of the user application calling the WM driver. The application may call any other functions and do other tasks between the WM API calls.
MSG QUEUE	Array of type r_wm_Msg_t collecting messages to be sent to the WM SYS Layer. Certain API calls send a message to be executed at a later point of time. A set of messages must be terminated with R_WM_FrameEndMark (“EndOfQueue”-flag) before start of processing by R_WM_FrameWait or R_WM_ExecuteNext. The user application must allocate this variable.
DRIVER-INTERNAL	This shows a simplified overview of the internal handling of the WM driver.
ISR	These are the Interrupts generated by the Video Output Hardware. The R_WM_FrameWait call uses these to synchronize write accesses to the Hardware. For a single threaded application, the WM will block until the Interrupt is generated. For multithreaded applications, other threads can continue. An alternative approach for single-threaded applications without blocking may be the use of R_WM_ExecuteNext instead of R_WM_FrameWait. In this case, the synchronization must be handled by the application layer.
SCREEN	The contents of the screen.

For the figure below, the user application is calling the WM API from the “APP” column. The driver runs in the same thread, thus driver calls are blocking.
The queue execution will be triggered by a call to FrameWait and the execution will then synchronize to the video output hardware using the SCANLINE and VBLANK interrupts. After updating the hardware, the user application can continue.
As an example, the lifetimes of two variables are shown, that are allocated in the user application. While the variables are locked, the variables must not be written by the user application. During scene changes, used variables overlap. Therefore, “Win1” cannot be reused for the second scene as long as it is still locked. The same also applies to variables of type r_wm_Sprite_t, r_wm_WinBuffer_t and r_wm_Capture_t.

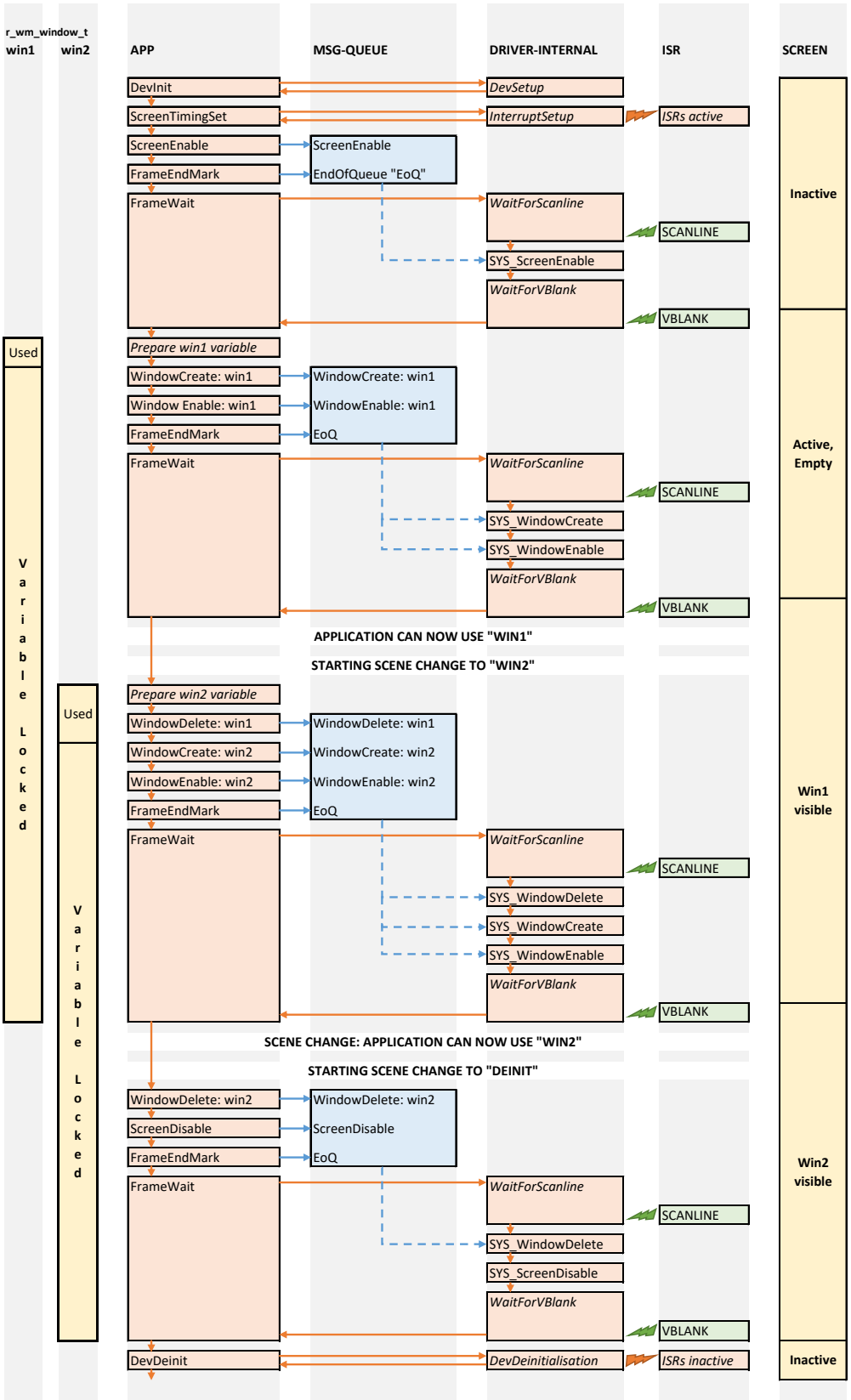


Figure 3-40 Flowchart for 2 variables lifetime

3.2.11 VOCA

3.2.11.1 Creation

Total 16 VOCA monitor areas can be created by R_WM_ScreenVocaCreate.
Following example shows to create a VOCA monitor area.

Example

```
static r_wm_Voca_t Voca0;
static r_wm_Voca_t Voca1;
static uint16_t ExpImg0[1000] = {0x0000,
    :
}; /* 100x80 pixel */
static uint16_t ExpImg1[1500] = {0x0000,
    :
}; /* 120x100 pixel */

R_WM_ScreenVocaInit(UNIT_0);

memset(&Voca0, 0, sizeof (r_wm_Vcoa_t));
Voca0.Threshold             = 1;
Voca0.AreaNum               = 0; /* Monitor area 0 */
Voca0.PosX                  = 10;
Voca0.PosY                  = 20;
Voca0.Width                 = 100;
Voca0.Height                = 80;
Voca0.RamAddr               = 0;
Voca0.ExpSize               = 1000;
Voca0.ExpImg                = &ExpImg0[0];
Voca0.Clut[0].RUpper        = 0xFF;
:
R_WM_ScreenVocaCreate(UNIT_0, &Voca0);
R_WM_ScreenVocaEnable(UNIT_0, &Voca0);

memset(&Voca1, 0, sizeof (r_wm_Vcoa_t));
Voca1.Threshold             = 1024;
Voca1.AreaNum               = 1; /* Monitor area 1 */
Voca1.PosX                  = 400;
Voca1.PosY                  = 200;
Voca1.Width                 = 120;
Voca1.Height                = 100;
Voca1.RamAddr               = 2048;
Voca1.ExpSize               = 1500;
Voca1.ExpImg                = &ExpImg1[0];
Voca1.Clut[0].RUpper        = 0xFF;
:
R_WM_ScreenVocaCreate(UNIT_0, &Voca1);
R_WM_ScreenVocaEnable(UNIT_0, &Voca1);
```

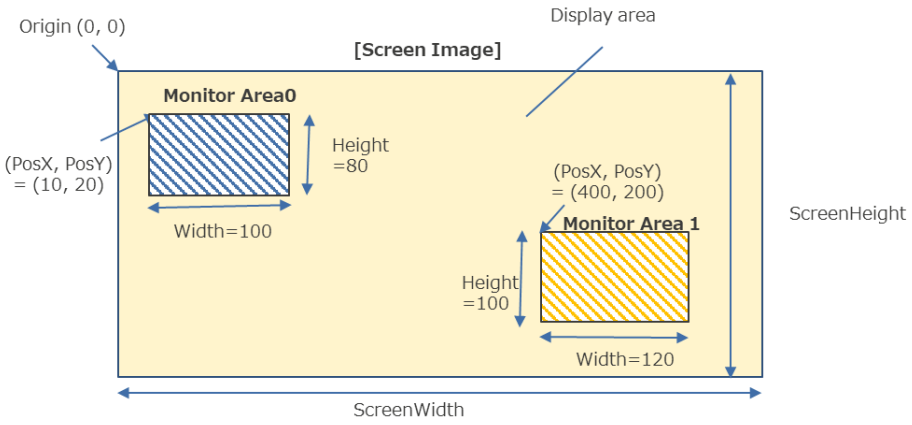


Figure 3-41 VOCA monitor area

Following table shows the range information for VOCA monitor area.
The monitor area must be set to the area on the screen.

Table 3-38 Parameter range

Parameter	Setting unit	Range	
		Min	Max
Threshold	-	1	0x3FFFF
AreaNum	-	0	15
PosX	Pixel	0	ScreenWidth - 1
Width	Pixel	1	128
(PosX + Width)	Pixel	1	ScreenWidth
PosY	Pixel	0	ScreenHeight - 1
Height	Pixel	1	128
(PosY + Height)	Pixel	1	ScreenHeight
RamAddr	16 bits	0	4095
ExpSize	16 bits	0	2048
(RamAddr + ExpSize)	16 bits	0	4096

VOCA can monitor one area in one Vsync interval even if several monitor areas are enabled.
If multiple monitor areas are set, the monitor area is cycled for each Vsync.
AreaNum=0 -> AreaNum=1 -> AreaNum=2 -> ... -> AreaNum=15

If the WM unit number changes from the previous monitor area, VOCA will wait the new WM unit Vsync and start monitoring. So, it is effective to keep the area numbers consecutive if monitoring multiple WM units.
e.g. AreaNum = 0 ~ 7 are assigned to WM unit 0 and AreaNum = 8 ~ 15 are assigned to WM unit 1.

3.2.11.2 Expected image

Expected image is set to internal RAM by R_WM_ScreenVocaCreate and R_WM_ScreenVocaExpImgSet.
Expected image color format is CLUT2. Each pixel is 2bit consisting of CLUT index number.
The internal RAM size is 8KBytes i.e. 32768 pixels.

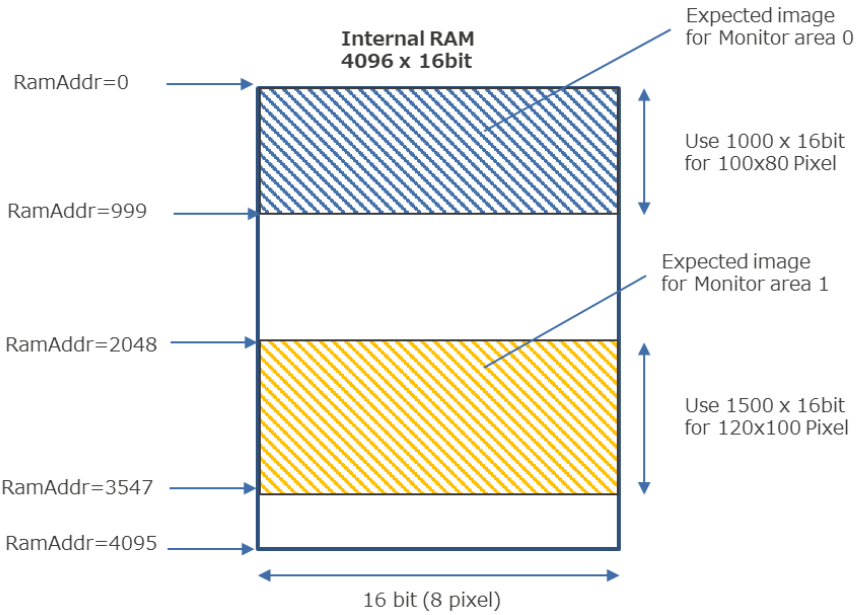


Figure 3-42 VOCA internal RAM

Example

```
static uint16_t ExpImg0[1000] = {0x0000,
    :
}; /* 100x80 pixel */
static uint16_t ExpImg1[1500] = {0x0000,
    :
}; /* 120x100 pixel */

R_WM_ScreenVocaExpImgSet(
    UNIT_0, &Voca0,
    1,              /* Threshold */
    0, 1000,        /* RAM address from 0 to 999 */
    &ExpImg0[0]);  /* Expected data */

R_WM_ScreenVocaExpImgSet(
    UNIT_0, &Voca1,
    1024,           /* Threshold */
    2048, 1500,     /* RAM address from 2048 to 3547 */
    &ExpImg1[0]);  /* Expected data */
```

Expected Image should be placed sequentially from top-left to bottom-down (top-down format).

Table 3-39 Expected image bit assign

Parameter	bit position							
	15-14	13-12	11-10	9-8	7-6	5-4	3-2	1-0
ExpImg[0]	pixel 0	pixel 1	pixel 2	pixel 3	pixel 4	pixel 5	pixel 6	pixel 7
ExpImg[1]	pixel 8	pixel 9	pixel 10	pixel 11	pixel 12	pixel 13	pixel 14	pixel 15
:	:	:	:	:	:	:	:	:
ExpImg[999]	pixel 7992	pixel 7993	pixel 7994	pixel 7995	pixel 7996	pixel 7997	pixel 7998	pixel 7999
:	:	:	:	:	:	:	:	:

3.2.11.3 VOCA CLUT

A VOCA CLUT entry has valid range (Upper limit and Lower limit) for each R, G, B component.
The range is specified as RGB888 format. Each range is 0 to 255.

For one pixel in the monitor area;
'R': The value of the upper 4 bits of the red component of the display signal.
'x': CLUT index number of the pixel set by R_WM_ScreenVocaExpImgSet.
VOCA H/W calculates the difference 'Diff' by following formula.

Formula:

```
if (R < (Clut[x].RLower >> 4))
{
    Diff = (Clut[x].RLower >> 4) - R;
}
else if (R > (Clut[x].RUpper >> 4))
{
    Diff = R - (Clut[x].RUpper >> 4);
}
else
{
    Diff = 0;
}
```

The same applies to the green and blue components.
R_WM_EVENT_VOCA_MISMACTH event is notified when the sum of the differences of all components of all pixels in the monitor area is greater than or equal to the "Threshold" value.

Example setting:
The image checks with R and G component. All pixels can be matched to any tables.

CLUT	R		G		B		Comment
	Lower	Upper	Lower	Upper	Lower	Upper	
Clut[0]	0x00	0x7F	0x00	0x7F	0x00	0xFF	R: low G: low
Clut[1]	0x00	0x7F	0x80	0xFF	0x00	0xFF	R: low G: high
Clut[2]	0x80	0xFF	0x00	0x7F	0x00	0xFF	R: high G: low
Clut[3]	0x80	0xFF	0x80	0xFF	0x00	0xFF	R: high G: high

The image checks with specific color. Even if the background changes depending on the scene, only characteristic colors can be detected.

CLUT	R		G		B		Comment
	Lower	Upper	Lower	Upper	Lower	Upper	
Clut[0]	0xF0	0xFF	0x00	0x00	0x00	0x00	Red
Clut[1]	0x70	0x8F	0x70	0x8F	0x70	0x8F	Gray
Clut[2]	0xF0	0xFF	0xF0	0xFF	0xF0	0xFF	White
Clut[3]	0x00	0xFF	0x00	0xFF	0x00	0xFF	Other

3.2.11.4 Update timing

Following example shows to synchronize expected image and frame updates.
This sample uses two monitor areas.

Example

```
/* Setup 2 monitor area in advance */
R_WM_ScreenVocaExpImgSet(UNIT_0, &Voca0, 1, 0, 1000, &ExpImg0[0]);
R_WM_ScreenVocaExpImgSet(UNIT_0, &Voca1, 1, 0, 1000, &ExpImg1[0]);
:

/* Get the next drawing buffer */
address = R_WM_WindowNewBufGet(LOC_WM_UNIT, &Window);

    ~ Drawing Frame#0 ~

/* Enable monitor area for Frame#0 and Disable old one */
R_WM_ScreenVocaDisable(UNIT_0, &Voca1);
R_WM_ScreenVocaEnable(UNIT_0, &Voca0);

/* Swap window */
R_WM_WindowSwap(LOC_WM_UNIT, &Window);
R_WM_FrameEndMark(LOC_WM_UNIT, 0);
R_WM_FrameWait(LOC_WM_UNIT, 0);
          :
/* Get the next drawing buffer */
address = R_WM_WindowNewBufGet(LOC_WM_UNIT, &Window);

    ~ Drawing Frame#1 ~

/* Enable monitor area for Frame#1 and Disable old one */
R_WM_ScreenVocaDisable(UNIT_0, &Voca0);
R_WM_ScreenVocaEnable(UNIT_0, &Voca1);

/* Swap window */
R_WM_WindowSwap(LOC_WM_UNIT, &Window);
R_WM_FrameEndMark(LOC_WM_UNIT, 0);
R_WM_FrameWait(LOC_WM_UNIT, 0);
```

3.2.12 DISCOM

3.2.12.1 Creation

A Discom device can be created per one DISCOM unit by R_WM_DiscomCreate.
Two DISCOM device can be created per WM unit.
DISCOM unit 0 and DISCOM unit 1 are connected to WM unit 0.
DISCOM unit 2 and DISCOM unit 3 are connected to WM unit 1.

Following example shows to create a DISCOM device.

Example

```
static r_wm_Discom_t Discom0;
static r_wm_Discom_t Discom1;

memset(&Discom0, 0, sizeof (r_wm_Discom_t));
Discom0.ExpCrc                = 0x12345678;
Discom0.PosX                  = 10;
Discom0.PosY                  = 20;
Discom0.Width                 = 240;
Discom0.Height                = 200;
Discom0.DiscomUnit            = 0;   /* Discom unit 0 */
Discom0.Next                  = R_NULL;
R_WM_DiscomCreate(UNIT_0, &Discom0);
R_WM_DiscomEnable(UNIT_0, &Discom0);

memset(&Discom1, 0, sizeof (r_wm_Discom_t));
Discom1.ExpCrc                = 0x3456789A;
Discom1.PosX                  = 400;
Discom1.PosY                  = 200;
Discom1.Width                 = 120;
Discom1.Height                = 100;
Discom1.DiscomUnit            = 1;  /* Discom unit 1 */
Discom1.Next                  = R_NULL;
R_WM_DiscomCreate(UNIT_0, &Discom1);
R_WM_DiscomEnable(UNIT_0, &Discom1);
```

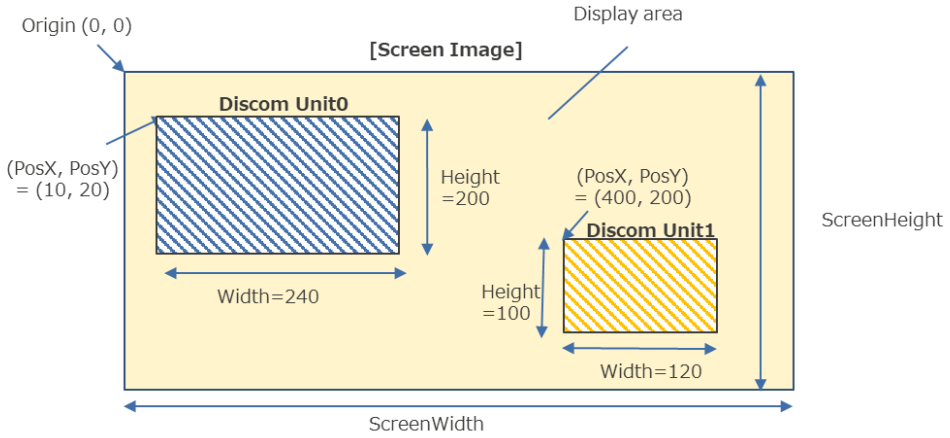


Figure 3-43 DISCOM calculation area

Following table shows the range information for DISCOM device.
The calculation area must be set to the area on the screen.

Table 3-40 Parameter range

Parameter	Setting unit	Range	
		Min	Max
PosX	Pixel	0	ScreenWidth - 1
Width	Pixel	1	ScreenWidth
(PosX + Width)	Pixel	1	ScreenWidth
PosY	Pixel	0	ScreenHeight - 1
Height	Pixel	1	ScreenHeight
(PosY + Height)	Pixel	1	ScreenHeight

DISCOM unit calculates CRC and compares it with the set expected value as “ExpCrc”.
R_WM_EVENT_DISCOM_MISMACTH event is notified if CRC does not match.

3.2.12.2 Update timing

Following example shows to synchronize expected CRC value and frame updates.

Example

```
/* Get the next drawing buffer */
address = R_WM_WindowNewBufGet(LOC_WM_UNIT, &Window);

    ~ Drawing Frame#0 ~

/* Set new CRC of Frame#0 */
R_WM_DiscomCrcSet(UNIT_0, &Discom0, ExpCrc0);

/* Swap window */
R_WM_WindowSwap(LOC_WM_UNIT, &Window);
R_WM_FrameEndMark(LOC_WM_UNIT, 0);
R_WM_FrameWait(LOC_WM_UNIT, 0);
          :
/* Get the next drawing buffer */
address = R_WM_WindowNewBufGet(LOC_WM_UNIT, &Window);

    ~ Drawing Frame#1 ~

/* Set new CRC of Frame#1 */
R_WM_DiscomCrcSet(UNIT_0, &Discom0, ExpCrc1);

/* Swap window */
R_WM_WindowSwap(LOC_WM_UNIT, &Window);
R_WM_FrameEndMark(LOC_WM_UNIT, 0);
R_WM_FrameWait(LOC_WM_UNIT, 0);
```

3.3 Device difference

The following table shows maximum value difference depending on the device.

Table 3-41 Maximum value

Feature	RH850/D1x Device Name					
	D1L2(H)	D1M1(H)	D1M1-V2	D1M1A	D1M2	D1M2H
Number of Units	1	1	1	2	2	2
Number of Capture units	0	1	1	1	1	2
Number of RLE windows per Unit	1	1	4	4	1	1
Number of Sprite windows per Unit	3	3	4	4	3	3
Display maximum screen width	480	1024	1024	1280	1280	1280
Display maximum screen height	320	1024	1024	1024	1024	1024
Number of DISCOM devices	0	2	2	4	4	4
Number of VOCA monitor areas	0	16	16	16	16	16

Following table shows available capture unit and synthesized layer of capture data.

Table 3-42 Capture Unit

Capture	Synthesize	RH850/D1x Device Name					
		D1L2(H)	D1M1(H)	D1M1-V2	D1M1A	D1M2	D1M2H
Unit0	Unit0 Layer0	-	✔	✔	✔	-	✔
Unit0	Unit1 Layer1	-	-	-	✔	-	✔
Unit1	Unit1 Layer0	-	-	-	-	✔	✔
Unit1	Unit0 Layer1	-	-	-	-	✔	✔

Following table shows available layer of Sprite window and RLE window.

Table 3-43 Sprite/RLE Window

WM Unit0	WM Unit1	Window type	RH850/D1x Device Name					
			D1L2(H)	D1M1(H)	D1M1-V2	D1M1A	D1M2	D1M2H
Layer0	Layer0	Sprite	-	-	✔ (*2)	✔(*2)	-	-
		RLE	✔	✔	✔ (*1)	✔(*1)	✔	✔
Layer1	Layer3	Sprite	✔	✔	✔ (*1)	✔(*1)	✔	✔
		RLE	-	-	✔ (*2)	✔(*2)	-	-
Layer2	Layer2	Sprite	✔	✔	✔ (*1)	✔(*1)	✔	✔
		RLE	-	-	✔ (*2)	✔(*2)	-	-
Layer3	Layer1	Sprite	✔	✔	✔ (*1)	✔(*1)	✔	✔
		RLE	-	-	✔ (*2)	✔(*2)	-	-

(*1): Available by default

(*2): Configurable by R_WM_WindowCapabilitiesSet.

The following table shows the function differences depending on the device.

Table 3-44 APIs supported by WM driver

Function	RH850/D1x Device Name				
	D1L2(H)	D1M1(H)	D1M1-V2	D1M1A	D1M2(H)
Scaling-Up [Function] - R_WM_WindowCreate - R_WM_WindowScaledSizeSet [Setting] - ScaledWidth != 0 - ScaledHeight != 0	NG	OK	OK	OK	OK
Display Gamma Correction [Function] - R_WM_ScreenColorCurveSet - R_WM_ScreenGammaSet	NG	OK	OK	OK	OK
Capture functions [Function] - R_WM_CaptureCreate - R_WM_CaptureDelete - R_WM_CaptureEnable - R_WM_CaptureDisable - R_WM_CaptureMove - R_WM_CaptureResize - R_WM_CaptureScaledSizeSet - R_WM_CaptureExtVsyncSet	NG	OK	OK	OK	OK
Capture / OIR / DISCOM / VOCA interrupt [Function] - R_WM_DevInit (EventCb) - R_WM_DevEventRegister [Setting] - R_WM_EVENT_VI_VBLANK - R_WM_EVENT_VI_OVERFLOW - 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	NG	OK	OK	OK	OK
Data Enable capturing [Function] - R_WM_CaptureCreate [Setting] - R_WM_CAPMODE_DE_MODE	NG	NG	OK	OK	OK
Sprite, RLE configuration [Function] - R_WM_WindowCapabilitiesSet	NG	NG	OK	OK	NG
VOCA functions [Function] - R_WM_ScreenVocaInit - R_WM_ScreenVocaDeInit - R_WM_ScreenVocaCreate - R_WM_ScreenVocaDelete - R_WM_ScreenVocaEnable - R_WM_ScreenVocaDisable - R_WM_ScreenVocaExpImgSet - R_WM_ScreenVocaClutSet - R_WM_ScreenActivityMonEnable - R_WM_ScreenActivityMonDisable	NG	OK	OK	OK	OK
DISCOM functions [Function] - R_WM_DiscomCreate - R_WM_DiscomDelete - R_WM_DiscomEnable - R_WM_DiscomDisable - R_WM_DiscomCrcSet - R_WM_DiscomCrcGet	NG	OK	OK	OK	OK

3.4 Header File List

Table 3-45 Header File List

No.	Header File Name	Description
(1)	r_wm_api.h	Header file for WM API.
(2)	r_ddb_api.h	Database for display timings (ddb).
(3)	r_typedefs.h	Header file for predefined data types.

4.Functions

4.1 Function List

This section describes about the WM API functions which are in *Table 4-1* and executable state of each function is described in the specification of each function.

Table 4-1 List of WM API Functions

Function Name	Purpose
R_WM_DevInit	This function initializes the driver and the hardware as far as necessary.
R_WM_DevEventRegister	Register for retrieving the notification on an event.
R_WM_DevDeinit	This function de initializes the driver and the hardware.
R_WM_DevInfoGet	Get information of screen and window parameters.
R_WM_GetVersionString	Version string of this WM driver.
R_WM_ScreenTimingSet	Create a screen on the specified video output.
R_WM_ScreenTimingSetByName	Create a screen on the specified video output.
R_WM_ScreenColorFormatSet	Set the color format for the signals of the specified video output.
R_WM_ScreenBgColorSet	Set the screen background color that is seen, if no window (or a transparent one) is on top of it.
R_WM_ScreenColorCurveSet	Set a gamma curve.
R_WM_ScreenGammaSet	Sets the output gamma correction.
R_WM_ScreenEnable	Switch on the display.
R_WM_ScreenDisable	Switch off the display.
R_WM_ScreenVocaInit	Setup VOCA H/W for specified WM unit.
R_WM_ScreenVocaDeInit	Disables all VOCA monitoring and deletes all created VOCA monitor areas.
R_WM_ScreenVocaCreate	Create a Video output monitor area
R_WM_ScreenVocaDelete	Delete a Video output monitor area.
R_WM_ScreenVocaEnable	Enable a Video output monitor area.
R_WM_ScreenVocaDisable	Disable a Video output monitor area.
R_WM_ScreenVocaExpImgSet	Set the expected image to internal RAM.
R_WM_ScreenVocaClutSet	Change the CLUT data of VOCA.
R_WM_ScreenActivityMonEnable	Enable activity monitor.
R_WM_ScreenActivityMonDisable	Disable activity monitor.
R_WM_WindowCapabilitiesSet	Configures the selectable window type, RLE window or Sprite window.
R_WM_WindowCreate	Create a window as specified in the Window parameter.
R_WM_WindowDelete	The function will delete the specified window.
R_WM_WindowEnable	Enable the window.
R_WM_WindowDisable	Disable the window.
R_WM_WindowMove	Move the window to the specified position.
R_WM_WindowResize	Resizes the window.
R_WM_WindowColorFmtSet	Sets the color format of the window.
R_WM_WindowAlphaSet	Change the window's alpha value.
R_WM_WindowPremultipliedAlphaEnable	Enable the window's pre-multiplied alpha mode.
R_WM_WindowPremultipliedAlphaDisable	Disable the window's pre-multiplied alpha mode.
R_WM_WindowVerticalMirrorEnable	Enable window's vertical mirroring.

R_WM_WindowVerticalMirrorDisable	Disable window's vertical mirroring.
R_WM_WindowSwap	If the window is a multi-buffer window, the background buffer is switched to the window surface.
R_WM_WindowNewDrawBufGet	Returns the next free buffer of the specified window.
R_WM_WindowVisibleBufGet	Returns the visible buffer of the specified window.
R_WM_WindowCurrentDrawBufGet	Return the current drawing frame buffer of the specified window.
R_WM_WindowExternalBufSet	Dynamically set the framebuffers in desired format for windows with externally allocated buffers.
R_WM_WindowColorKeyEnable	Enable the color key of the selected Window.
R_WM_WindowColorKeyDisable	Disable the color key of the selected Window
R_WM_WindowClutSet	Sets the color lookup table.
R_WM_WindowDeleteAllSprites	Deletes all Sprites associated to the window.
R_WM_WindowScaledSizeSet	Change the scaled size of the window.
R_WM_FrameEndMark	Frame delimiter marks the end of the sequence of the WM requests to be executed during one frame redraw.
R_WM_FrameWait	Executes the requests from the message queue up to the specified frame delimiter.
R_WM_FrameExecuteNext	Executes the requests in the message queue.
R_WM_FrameExecuteVoca	Execute jobs for VOCA in message queue.
R_WM_FrameExecuteDiscom	Execute jobs for DISCOM in message queue.
R_WM_CaptureCreate	Create a video capture surface.
R_WM_CaptureDelete	Delete the specified capturing surface.
R_WM_CaptureEnable	Enable the video capturing surface and start the capturing.
R_WM_CaptureDisable	Disable the video capturing.
R_WM_Cap_CapBufGet	Get the capture buffer.
R_WM_Cap_DispBufGet	Get the display buffer.
R_WM_CaptureMove	Move the capturing start position.
R_WM_CaptureResize	Resize the capturing signal size.
R_WM_CaptureScaledSizeSet	Resize the scaling-down size.
R_WM_CaptureExtVsyncSet	Sets the external Vsync and Hsync.
R_WM_SpriteCreate	Create a Sprite data.
R_WM_SpriteEnable	Enable the specified Sprite.
R_WM_SpriteDisable	Disable the specified Sprite.
R_WM_SpriteMove	Move the Sprite to the specified X/Y and Z-order location.
R_WM_SpriteBufSet	Set the Sprite buffer.
R_WM_SpriteDelete	Remove the Sprite from the host window.
R_WM_ErrorCallbackSet	Set the error callback function.
R_WM_ErrorHandler	The function is the driver's central error handler.
R_WM_ColorFmtBitsPerPixGet	Return the bits per pixel count for the specified format.
R_WM_DiscomCreate	Create a Discom device.
R_WM_DiscomDelete	Delete a Discom device.
R_WM_DiscomEnable	Enable a Discom device.
R_WM_DiscomDisable	Disable a Discom device.
R_WM_DiscomCrcSet	Changes expected CRC.
R_WM_DiscomCrcGet	Get the latest calculated CRC.

4.2 WM API Functions

This chapter describes the application interface functions, which are required for general use of the driver.

4.2.1 Device functions

The section describes driver functions, which are required for general use of the driver, but which are related to a specific functionality of the macro itself.

4.2.1.1 R_WM_DevInit

Function Prototypes

```
r_wm_Error_t R_WM_DevInit(const unit32_t     Unit,
                          r_wm_Msg_t  *const MsgQueue,
                          const uint32_t     Size,
                          void(*EventCb)     (uint32_t Unit,const r_wm_Event_t* Event),
                          void        *const CpuHeap,
                          void        *const VidHeap)
```

Input Parameter

Table 4-2 Input parameter of R_WM_DevInit

Parameter	Description
Unit	Specifies the WM unit number.
MsgQueue	Specifies an array of type R_WM_Msg_t. This array is used by Window Manager to store messages.
Size	Specifies the number of elements in the message queue. The maximum size is 1024.
EventCb	Specifies the event notification callback function. If this is R_NULL, no callback will be invoked upon event detection.
CpuHeap	Specifies the managed CPU heap. It can be set as R_NULL, when frame buffer is allocated by external mode only.
VidHeap	Specifies the managed video heap. It can be set as R_NULL, when frame buffer is allocated by external mode only.

Table 4-3 Input parameter of EventCb

Parameter	Description
Unit	WM unit number.
Event	Event information.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_PARAM_INCORRECT	- Parameter provided to a function is incorrect.
R_WM_ERR_NOT_UNINITIALIZED	- WM unit status is invalid.
R_WM_ERR_SYS_LAYER_INIT_FAILURE	- Error occurs in R_WM_Sys_MsgQueueSetup or R_WM_Sys_DevInit .

Description

This function initializes the driver and the hardware as far as necessary. The driver makes sure, that the driver is set into a default configuration.
WM unit status will become Initialized state after the execution of this function.

Reentrancy

Non-reentrant

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t
r_wm_Msg_t
r_wm_Event_t

4.2.1.2 R_WM_DevEventRegister

Function Prototypes

```
r_wm_Error_t R_WM_DevEventRegister(const uint32_t        Unit,
                                   const r_wm_EventId_t   EventId,
                                   const uint32_t         Arg)
```

Input Parameter

Table 4-4 Input parameter of R_WM_ DevEventRegister

Parameter	Description
Unit	Specifies the WM unit number.
EventId	Specifies the event of the event to be registered. 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 scanline number on which the event will trigger. This argument is valid when EventId is R_WM_EVENT_SCANLINE or R_WM_EVENT_OIR_SCANLINE. The range is 0 to 2047.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_DevEventRegister.

Description

This function enables the notification of specified event.
Once the event is enabled, it cannot be disabled until R_WM_DevDeInit is called.
Callback function to notify events are installed by R_WM_DevInit.

Because V-sync signal is not stable while WM unit is Uninitialized state or Initialized state, it is recommended to enable these events while WM unit is Display Initialized state or Display Active state.

Regarding R_WM_EVENT_SCANLINE event, the line position that triggers the event is set to the default value by function R_WM_ScreenTimingSet or R_WM_ScreenTimingSetByName. And, this function can be changed the trigger position. See *3.2.9.5* for the detail.

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.

Regarding R_WM_EVENT_OIR_VBLANK and R_WM_EVENT_OIR_SCANLINE, only WM unit 0 is supported.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_VDCE_Sys_Lock
- R_VDCE_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t
r_wm_EventId_t

4.2.1.3 R_WM_DevDeinit

Function Prototypes

```
r_wm_Error_t R_WM_DevDeinit(const uint32_t    Unit)
```

Input Parameter

Table 4-5 Input parameter of R_WM_DevDeinit

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_DELETED	- Allocating window is remained.
R_WM_ERR_DEV_DEINIT_FAILED	- Error occurs in R_WM_Sys_DevDeinit.

Description

This function de-initializes the driver and the hardware.
WM unit status will become Uninitialized state after executing this function.
If WM unit is already de-initialized status, this function does nothing and returns R_WM_ERR_OK.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_VDCE_Sys_Lock
- R_VDCE_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
All created windows should be deleted by R_WM_WindowDelete.
All created capture surface should be deleted by R_WM_CaptureDelete.
All created DISCOM device should be deleted by R_WM_DiscomDelete.
Initialized VOCA should be de-initialized by R_WM_ScreenVocaDeInit.

See also

r_wm_Error_t

4.2.1.4 R_WM_DevInfoGet

Function Prototypes

```
r_wm_Error_t R_WM_DevInfoGet(const uint32_t      Unit,
                             uint32_t     *const LayerNum,
                             uint32_t     *const PitchMax,
                             uint32_t     *const WidthMax,
                             uint32_t     *const HeightMax)
```

Input Parameter

Table 4-6 Input parameter of R_WM_DevInfoGet

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

Table 4-7 Output parameter of parameter of R_WM_DevInfoGet

Parameter	Description
LayerNum	WM stores the number of usable layers. Specifies the parameter to R_NULL if it is unnecessary.
PitchMax	WM stores the maximum pitch of a layer. Specifies the parameter to R_NULL if it is unnecessary.
WidthMax	WM stores the maximum width of a layer. Specifies the parameter to R_NULL if it is unnecessary.
HeightMax	WM stores the maximum height of a layer. Specifies the parameter to R_NULL if it is unnecessary.

Return Codes

R_WM_ERR_OK - No error occurred.
R_WM_ERR_INVALID_WM_UNIT - Unit number is outside the range.

Description

This function gets information of window parameters.

Reentrancy

Reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t

4.2.1.5 R_WM_GetVersionString

Function Prototypes

```
const int8_t *R_WM_GetVersionString(void)
```

Input Parameter

None

Input-Output Parameter

None

Output Parameter

None

Return Codes

Version string.

Description

This function returns version string of the WM driver.

Reentrancy

Reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

None

4.2.2 Screen functions

A screen is a physical video output unit. There is no frame-buffer associated with a screen, but a screen has a background color.

4.2.2.1 R_WM_ScreenTimingSet

Function Prototypes

```
r_wm_Error_t R_WM_ScreenTimingSet(const uint32_t              Unit,
                                  const r_ddb_Timing_t *const Timing)
```

Input Parameter

Table 4-8 Input parameter of R_WM_ScreenTimingSet

Parameter	Description
Unit	Specifies the WM unit number.
Timing	Specifies the timing parameter contains all the information to drive the display.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit is not initialized.
R_WM_ERR_NOT_DISABLED	- WM unit is Display Active state.
R_WM_ERR_DISPLAY_TIMING_SET	- Error occurs in R_WM_Sys_ScreenTimingSet.

Description

This function sets display area and timings. See *3.2.2.1* for the details.
The setting values will be active after executing R_WM_ScreenEnable.
WM unit status will become Display Initialized state after execution this function.
About r_ddb_Timing_t structure, see VDCE driver User's manual.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_VDCE_Sys_Lock
- R_VDCE_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t
R_WM_ScreenEnable

4.2.2.2 R_WM_ScreenTimingSetByName

Function Prototypes

```
r_wm_Error_t R_WM_ScreenTimingSetByName(const uint32_t       Unit,
                                        const int8_t *const  Name)
```

Input Parameter

Table 4-9 Input parameter of R_WM_ScreenTimingSetByName

Parameter	Description
Unit	Specifies the WM unit number.
Name	Specifies the name (identifier), which is used to lookup for the correct screen timing in the display data base.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit is not initialized.
R_WM_ERR_NOT_DISABLED	- WM unit is Display Active status.
R_WM_ERR_DISPLAY_TIMING_SET	- Error occurs in R_WM_Sys_ScreenTimingSet.

Description

This function sets display area and timings from specified name (identifier).
This function gets the r_ddb_Timing_t parameter from data base and execute R_WM_ScreenTimingSet.
About the database, see R_WM_Sys_ScreenTimingSetByName in Porting Layer Guide.

The new values will be active after executing R_WM_ScreenEnable.
WM unit status will become Display Initialized state after execution this function.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_VDCE_Sys_Lock
- R_VDCE_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t
R_WM_ScreenEnable

4.2.2.3 R_WM_ScreenColorFormatSet

Function Prototypes

```
r_wm_Error_t R_WM_ScreenColorFormatSet(const uint32_t             Unit,
                                       const r_wm_OutColorFmt_t   OutFmt)
```

Input Parameter

Table 4-10 Input parameter of R_WM_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

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit is not initialized.
R_WM_ERR_NOT_DISABLED	- WM unit is Display Active status.
R_WM_ERR_DISPLAY_OUTPUT_FORMAT_SET	- Error occurs in R_WM_Sys_ScreenColorFormatSet.

Description

This function sets the color format of the video output signal. See *3.2.2.2* for the detail.
If user doesn't call this function, RGB888 with no optional flags is the default.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_VDCE_Sys_Lock
- R_VDCE_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t
r_wm_OutColorFmt_t

4.2.2.4 R_WM_ScreenBgColorSet

Function Prototypes

```
r_wm_Error_t R_WM_ScreenBgColorSet(const uint32_t  Unit,
                                   const uint8_t   Red,
                                   const uint8_t   Green,
                                   const uint8_t   Blue)
```

Input Parameter

Table 4-11 Input parameter of R_WM_ScreenBgColorSet

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

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK - No error occurred.
R_WM_ERR_INVALID_WM_UNIT - Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED - WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE - Message queue is overflow.

Description

This function set the screen background color that is seen if no window (or a transparent one) is on top of it.
If user doesn't call this function, black (Red = 0, Green = 0, Blue = 0) is set as default.

Following error code will be notified by error callback when error occurs asynchronously.
R_WM_ERR_COULD_NOT_SET_SCREEN_BG_COLOR- Error occurs in R_WM_Sys_ScreenBgColorSet.
R_WM_ERR_VOUT_INTERNAL - Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t

4.2.2.5 R_WM_ScreenColorCurveSet

Function Prototypes

```
r_wm_Error_t R_WM_ScreenColorCurveSet(const uint32_t                Unit,
                                      const uint32_t                NumEntries,
                                      const r_wm_ClutEntry_t *const ColorCurve)
```

Input Parameter

Table 4-12 Input parameter of R_WM_ScreenColorCurveSet

Parameter	Description
Unit	Specifies the WM unit number.
NumEntries	Specifies the number of entries in the Color Curve table. Set to 33 fixed.
ColorCurve	Pointer to Color curve table. The head shall point to the starting address of an array of 33 elements of type r_wm_ClutEntry_t. Element A of this structure is not used.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_PARAM_INCORRECT	- Parameter provided to a function is incorrect.
R_WM_ERR_NULL_PTR_ARGUMENT	- Pointer provided to a function is R_NULL.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function sets gamma correction with color curve reference points.
Using this curve, each RGB color channel is individually corrected according to the given curve.
See *3.2.2.3* for the detail.

The instance of ColorCurve must be hold until R_WM_FrameWait corresponding to this function is completed.
This function will overwrite the settings of R_WM_ScreenGammaSet.

Following error code will be notified by error callback when error occurs asynchronously.
R_WM_ERR_COULD_NOT_SET_SCREEN_COLOR_CURVE - Error occurs in R_WM_Sys_ScreenColorCurveSet.
R_WM_ERR_VOUT_INTERNAL - Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t
r_wm_ClutEntry_t
R_WM_ScreenGammaSet

4.2.2.6 R_WM_ScreenGammaSet

Function Prototypes

```
r_wm_Error_t R_WM_ScreenGammaSet(const uint32_t   Unit,
                                 const uint8_t    GammaRed,
                                 const uint8_t    GammaGreen,
                                 const uint8_t    GammaBlue)
```

Input Parameter

Table 4-13 Input parameter of R_WM_ScreenGammaSet

Parameter	Description
Unit	Specifies the WM unit number.
GammaRed	Specifies the gamma factor for red. Range is 32-255 which corresponds to gamma = [0.25(=32/128) …1.99 (=255/128)]. 128 is the default value. It means gamma = 1.0.
GammaGreen	Specifies the gamma factor for green. Range is 32-255 which corresponds to gamma = [0.25(=32/128) …1.99 (=255/128)]. 128 is the default value. It means gamma = 1.0.
GammaBlue	Specifies the gamma factor for blue. Range is 32-255 which corresponds to gamma = [0.25(=32/128) …1.99 (=255/128)]. 128 is the default value. It means gamma = 1.0.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function sets the gamma correction with gamma value.
Each RGB color channel is individually corrected according to the given gamma value.
See *3.2.2.4* for the detail.
This function will overwrite the settings of R_WM_ScreenColorCurveSet.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_COULD_NOT_SET_SCREEN_GAMMA	- Error occurs in R_WM_Sys_ScreenGammaSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t
R_WM_ScreenColorCurveSet

4.2.2.7 R_WM_ScreenEnable

Function Prototypes

```
r_wm_Error_t R_WM_ScreenEnable(const uint32_t    Unit)
```

Input Parameter

Table 4-14 Input parameter of R_WM_ScreenEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables the screen.
WM unit status will become Display Active state after asynchronous execution is complete.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_SCREEN_TIMING_NOT_SET	- WM unit status is Initialized state.
R_WM_ERR_COULD_NOT_ENABLE_SCREEN	- Error occurs in R_WM_Sys_ScreenEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t

4.2.2.8 R_WM_ScreenDisable

Function Prototypes

```
r_wm_Error_t R_WM_ScreenDisable(const uint32_t Unit)
```

Input Parameter

Table 4-15 Input parameter of R_WM_ScreenDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables the screen.
WM unit status will become Display Initialized state after asynchronous execution is complete.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_COULD_NOT_DISABLE_SCREEN	- Error occurs in R_WM_Sys_ScreenEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
All created DISCOM device should be deleted by R_WM_DiscomDelete before this function is called.
Initialized VOCA should be de-initialized by R_WM_ScreenVocaDeInit before this function is called.

See also

r_wm_Error_t

4.2.2.9 R_WM_ScreenVocaInit

Function Prototypes

```
r_wm_Error_t R_WM_ScreenVocaInit(const uint32_t       Unit)
```

Input Parameter

Table 4-16 Input parameter of R_WM_ScreenVocaInit

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_SCREEN_TIMING_NOT_SET	- WM unit status is not Display Active state.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function setups VOCA H/W of specified WM unit.

When WM unit 0 is specified, WM unit 0 should be in Display Active state.
When WM unit 1 is specified, both WM unit 0 and WM unit 1 should be in Display Active state.
That is, the message queue of R_WM_ScreenEnable should also be completed.

This function can be executed when of all Video monitor areas and activity monitors, including another WM unit, are disabled.

VOCA requires a front porch of 1 pixel or more.
If front porch is set to 0 by R_WM_ScreenTimingSet or R_WM_ScreenTimingSetByName, this function fails.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_SYS_LAYER_INIT_FAILURE	- Error occurs in R_WM_Sys_ScreenVocaInit.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.
When WM unit 0 is specified, WM unit 0 should be in Display Active state.
When WM unit 1 is specified, both WM unit 0 and WM unit 1 should be in Display Active state.
When any video output monitor or activity monitor is enabled of another WM unit, disable all monitors.

See also

r_wm_Error_t

4.2.2.10 R_WM_ScreenVocaDeInit

Function Prototypes

```
r_wm_Error_t R_WM_ScreenVocaDeInit(const uint32_t       Unit)
```

Input Parameter

Table 4-17 Input parameter of R_WM_ScreenVocaDeInit

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables all VOCA monitoring (Video output monitor and Activity monitor) and deletes all created VOCA monitor areas of specified WM unit.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_DEV_DEINIT_FAILED	- Error occurs in R_WM_Sys_ScreenVocaDeInit.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t

4.2.2.11 R_WM_ScreenVocaCreate

Function Prototypes

```
r_wm_Error_t R_WM_ScreenVocaCreate(const uint32_t       Unit,
                                   r_wm_Voca_t   *const Voca)
```

Input Parameter

Table 4-18 Input parameter of R_WM_ScreenVocaCreate

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-19 Input-Output parameter of R_WM_ScreenVocaCreate

Parameter	Description
Voca	Specifies the pointer of VOCA structure. The contents of the structure are updated by WM, but user does not need to refer to the contents of the output.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function creates a Video output monitor area to specified WM unit.
16 monitor areas can be created in all WM units.
The instance of r_wm_Voca_t structure must be hold until R_WM_ScreenVocaDelete is completed.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_VOCA_CREATE_FAILED	- Error occurs in R_WM_Sys_ScreenVocaCreate.
R_WM_ERR_VOCA_NOT_FOUND	- Specified VOCA monitor area is already created.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t
r_wm_Voca_t

4.2.2.12 R_WM_ScreenVocaDelete

Function Prototypes

```
r_wm_Error_t R_WM_ScreenVocaDelete(const uint32_t       Unit,
                                   r_wm_Voca_t   *const Voca)
```

Input Parameter

Table 4-20 Input parameter of R_WM_ScreenVocaDelete

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-21 Input-Output parameter of R_WM_ScreenVocaDelete

Parameter	Description
Voca	Specifies the pointer of VOCA structure generated by R_WM_ScreenVocaCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function deletes a Video output monitor area.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_VOCA_DELETE_FAILED	- Error occurs in R_WM_Sys_ScreenVocaDelete.
R_WM_ERR_VOCA_NOT_FOUND	- Specified VOCA monitor area is not found.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See *Table 2-6* about WM unit status conditions.
See *Table 2-14* about VOCA monitor area status conditions.

See also

r_wm_Error_t
r_wm_Voca_t

4.2.2.13 R_WM_ScreenVocaEnable

Function Prototypes

```
r_wm_Error_t R_WM_ScreenVocaEnable(const uint32_t       Unit,
                                   r_wm_Voca_t   *const Voca)
```

Input Parameter

Table 4-22 Input parameter of R_WM_ScreenVocaEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-23 Input-Output parameter of R_WM_ScreenVocaEnable

Parameter	Description
Voca	Specifies the pointer of VOCA structure generated by R_WM_ScreenVocaCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables a Video output monitor area.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_VOCA_ENABLE_FAILED	- Error occurs in R_WM_Sys_ScreenVocaEnable.
R_WM_ERR_VOCA_NOT_FOUND	- Specified VOCA monitor area is not found.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t
r_wm_Voca_t

4.2.2.14 R_WM_ScreenVocaDisable

Function Prototypes

```
r_wm_Error_t R_WM_ScreenVocaDisable(const uint32_t       Unit,
                                    r_wm_Voca_t   *const Voca)
```

Input Parameter

Table 4-24 Input parameter of R_WM_ScreenVocaDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-25 Input-Output parameter of R_WM_ScreenVocaDisable

Parameter	Description
Voca	Specifies the pointer of VOCA structure generated by R_WM_ScreenVocaCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables a Video output monitor area.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_VOCA_ENABLE_FAILED	- Error occurs in R_WM_Sys_ScreenVocaEnable.
R_WM_ERR_VOCA_NOT_FOUND	- Specified VOCA monitor area is not found.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t
r_wm_Voca_t

4.2.2.15 R_WM_ScreenVocaExpImgSet

Function Prototypes

```
r_wm_Error_t R_WM_ScreenVocaExpImgSet(const uint32_t        Unit,
                                      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 4-26 Input parameter of R_WM_ScreenVocaExpImgSet

Parameter	Description
Unit	Specifies the WM unit number.
Threshold	Acceptable mismatch (difference) of a Monitor Area. Same as Threshold of r_wm_Voca_t.
RamAddr	Internal RAM start index to update. Same as RamAddr of r_wm_Voca_t. This function overwrites RAM from index #(RamAddr) to index #(RamAddr + ExpSize - 1).
ExpSize	The number of array of Data to update. Same as ExpSize of r_wm_Voca_t. The range is 1 to 2048. If ExpSize is 0, only Threshold and RamAddr for specified monitor area is updated.
ExpImg	Pointer to update expected image array. Same as ExpImg of r_wm_Voca_t. User should prepare the number of arrays specified by 'ExpSize'. If ExpSize is 0, set R_NULL.

Input-Output Parameter

Table 4-27 Input-Output parameter of R_WM_ScreenVocaExpImgSet

Parameter	Description
Voca	Specifies the pointer of VOCA structure generated by R_WM_ScreenVocaCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function sets the expected image to internal RAM.
This function also updates the Threshold and RamAddr for specified monitor area.
See *3.2.11.2* for the detail.
The instance of ExpImg must be hold until R_WM_FrameWait or R_WM_FrameExecuteVoca corresponding to this function is completed.
This function is valid only once per Vsync in one monitor area. If this function is executed more than once, the last setting becomes effective.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_VOCA_NOT_FOUND	- Specified VOCA monitor area is not found.
R_WM_ERR_SYS_VOCA_SET_FAILED	- Error occurs in R_WM_Sys_ScreenVocaExpImgSet.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteVoca.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t
r_wm_Voca_t

4.2.2.16 R_WM_ScreenVocaClutSet

Function Prototypes

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

Input Parameter

Table 4-28 Input parameter of R_WM_ScreenVocaClutSet

Parameter	Description
Unit	Specifies the WM unit number.
NumEntries	Number of CLUT entries. The range is 1 to 4. This function updates from Clut #0 to #(NumEntries - 1)
Clut	Pointer to array of CLUT. User should prepare the number of arrays specified by 'NumEntries'.

Input-Output Parameter

Table 4-29 Input-Output parameter of R_WM_ScreenVocaClutSet

Parameter	Description
Voca	Specifies the pointer of VOCA structure generated by R_WM_ScreenVocaCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function changes the CLUT data of VOCA.
See *3.2.11.3* for the detail.
The instance of Clut must be hold until R_WM_FrameWait or R_WM_FrameExecuteVoca corresponding to this function is completed.
This function is valid only once per Vsync in one monitor area. If this function is executed more than once, the last setting becomes effective.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_VOCA_NOT_FOUND	- Specified VOCA monitor area is not found.
R_WM_ERR_SYS_VOCA_SET_FAILED	- Error occurs in R_WM_Sys_ScreenVocaClutSet.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteVoca.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t
r_wm_Voca_t
r_wm_VocaClutEntry_t

4.2.2.17 R_WM_ScreenActivityMonEnable

Function Prototypes

```
r_wm_Error_t R_WM_ScreenActivityMonEnable(const uint32_t      Unit,
                                          const uint32_t      UpperTime,
                                          const uint32_t      LowerTime)
```

Input Parameter

Table 4-30 Input parameter of R_WM_ScreenActivityMonEnable

Parameter	Description
Unit	Specifies the WM unit number.
UpperTime	Maximum detection time. Unit is microseconds. Valid range is 0 to 136467 [usec].
LowerTime	Minimum detection time. Unit is microseconds. Valid range is 0 to 136467 [usec].

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables activity monitor of VOCA.
Activity monitor notifies R_WM_EVENT_ACT_MON_ERROR when output Vsync interval is not from LowerTime to UpperTime.
LowerTime and UpperTime can be set in (100/3 = 33.3…) microseconds units. WM driver rounds to the nearest value.

Activity monitor can work when at least one Video Output Monitor is enabled by R_WM_ScreenVocaEnable for the respective WM unit.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_SYS_ACT_MON_FAILED	- Error occurs in R_WM_Sys_ScreenActMonEnable.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t

4.2.2.18 R_WM_ScreenActivityMonDisable

Function Prototypes

```
r_wm_Error_t R_WM_ScreenActivityMonDisable(const uint32_t       Unit)
```

Input Parameter

Table 4-31 Input parameter of R_WM_ScreenActivityMonDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables activity monitor of VOCA.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_SYS_ACT_MON_FAILED	- Error occurs in R_WM_Sys_ScreenActMonEnable.
R_WM_ERR_VOCA_INTERNAL	- Error occurs in VOCA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-14*** about VOCA monitor area status conditions.

See also

r_wm_Error_t

4.2.3 Window functions

4.2.3.1 R_WM_WindowCapabilitiesSet

Function Prototypes

```
r_wm_Error_t R_WM_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 4-32 Input parameter of R_WM_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

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_RANGE_WM	- Parameter is the outside of the range.
R_WM_ERR_NOT_UNINITIALIZED	- WM driver is not uninitialized.
R_WM_ERR_NOT_SUPPORTED	- The function is not supported with target device.

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 available depending on RH850/D1x device. See *Table 3-44* for the detail.

For both RLE and Sprite settings, Frame buffer type window can be created in the layer.

Default configuration is as follows.

Table 4-33 Default configuration

Modes	Default Values
Capability0	RLE Window
Capability1	Sprite Window
Capability2	Sprite Window
Capability3	Sprite Window

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
All WM units should be in Uninitialized state before using this function.

See also

r_wm_Error_t
r_wm_WinColorFmt_t
r_wm_WinMode_t
r_wm_WinCapbs_t

4.2.3.2 R_WM_WindowCreate

Function Prototypes

```
r_wm_Error_t R_WM_WindowCreate(const uint32_t          Unit,
                               r_wm_Window_t    *const Window)
```

Input Parameter

Table 4-34 Input parameter of R_WM_WindowCreate

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-35 Input-Output parameter of R_WM_WindowCreate

Parameter	Description
Window	Specifies the pointer of window structure. The contents of the structure are updated by WM, but user does not need to refer to the contents of the output.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Parameter provided to a function is NULL.
R_WM_ERR_PARAM_INCORRECT	- Parameter provided to a function is incorrect.
R_WM_ERR_COLORFMT	- Invalid color format.
R_WM_ERR_RANGE_WM	- Parameter is the outside of the range.
R_WM_ERR_MALLOC_FAILED	- Failed to allocate memory.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function creates a window as specified in the Window parameter on screen Unit.
User should initialize all members of parameter of Window structure before this function is called.

The instance of Window structure must be hold until R_WM_WindowDelete is completed.
When external frame buffer mode is selected, the instance of buffer information pointed by Window->Surface.Fb.Buffer must be hold until R_WM_WindowDelete is completed.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_COLORFMT	- Message queue is broken.
R_WM_ERR_NO_PHYS_WINDOW	- Available window is not remained.
R_WM_ERR_SYS_WIN_CREATE_FAILED	- Error occurs in R_WM_Sys_WindowCreate.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.3 R_WM_WindowDelete

Function Prototypes

```
r_wm_Error_t R_WM_WindowDelete(const uint32_t        Unit,
                               r_wm_Window_t  *const Window)
```

Input Parameter

Table 4-36 Input parameter of R_WM_WindowDelete

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-37 Input-Output parameter of R_WM_WindowDelete

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Parameter provided to a function is NULL.
R_WM_ERR_PARAM_INCORRECT	- Parameter provided to a function is incorrect.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

The function deletes the specified window.
All internally allocated data buffer will be freed by this function.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_FREE_FAILED	- Error occurs in R_WM_Sys_Free.
R_WM_ERR_SYS_WIN_DELETE_FAILED	- Error occurs in R_WM_Sys_WindowDelete.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.
All created sprites data in the window should be deleted by R_WM_SpriteDelete or R_WM_WindowDeleteAllSprites.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.4 R_WM_WindowEnable

Function Prototypes

```
r_wm_Error_t R_WM_WindowEnable(const uint32_t          Unit,
                               r_wm_Window_t    *const Window)
```

Input Parameter

Table 4-38 Input parameter of R_WM_WindowEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-39 Input-Output parameter of R_WM_WindowEnable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables the window. The window will be visible on the screen after calling this function.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SCREEN_TIMING_NOT_SET	- WM unit is not Display Active state.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_WindowEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.5 R_WM_WindowDisable

Function Prototypes

```
r_wm_Error_t R_WM_WindowDisable(const uint32_t         Unit,
                                r_wm_Window_t   *const Window)
```

Input Parameter

Table 4-40 Input parameter of R_WM_WindowDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-41 Input-Output parameter of R_WM_WindowDisable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables the window. The window will be invisible on the screen after calling this function.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_WindowEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.6 R_WM_WindowMove

Function Prototypes

```
r_wm_Error_t R_WM_WindowMove(const uint32_t        Unit,
                             r_wm_Window_t  *const Window,
                             const int32_t         PosX,
                             const int32_t         PosY,
                             const int32_t         PosZ)
```

Input Parameter

Table 4-42 Input parameter of R_WM_WindowMove

Parameter	Description
Unit	Specifies the WM unit number.
PosX	Specifies the new X position of the window on the screen. Unit is pixels. Range is -1280 to 1279.
PosY	Specifies the new Y position of the window on the screen. Unit is pixels. Range is -1024 to 1023.
PosZ	Specifies the new Z position of the window.

Input-Output Parameter

Table 4-43 Input-Output parameter of R_WM_WindowMove

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_RANGE_WM	- Parameter is the outside of the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function moves the window to the specified position.
PosX and PosY should be set to a value greater than or equal to 0 in case of RLE window type.
If Window position is out of screen, WM disable the layer internally.

When PosZ is changed from previous value, all window will be re-constructed and changes layer assignment of the Windows. See *3.2.3.3* for the detail.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Window is no found.
R_WM_ERR_SYS_WIN_MOVE_FAILED	- Error occurs in R_WM_Sys_WindowPosSet .
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.7 R_WM_WindowResize

Function Prototypes

```
r_wm_Error_t R_WM_WindowResize(const uint32_t         Unit,
                               r_wm_Window_t   *const Window,
                               const uint32_t         Pitch,
                               const uint32_t         Width,
                               const uint32_t         Height)
```

Input Parameter

Table 4-44 Input parameter of R_WM_WindowResize

Parameter	Description
Unit	Specifies the WM unit number.
Pitch	Specifies the distance in pixels between subsequent rows in the framebuffer memory. Range is from (1024 / bpp) to (261120 / bpp).
Width	Specifies the window width in pixels. Range is 3 - 1280. When scaling-up is disabled, this value is also window frame buffer width.
Height	Specifies the window height in pixels. Range is 1 - 1024. When scaling-up is disabled, this value is also window frame buffer height.

Input-Output Parameter

Table 4-45 Input-Output parameter of R_WM_WindowResize

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function changes the geometry of the window.
The window can be dynamically resized until the life time of the window.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_GEOMETRY_SET_FAILED	- Error occurs in R_WM_Sys_WindowGeomSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t

4.2.3.8 R_WM_WindowColorFmtSet

Function Prototypes

```
r_wm_Error_t R_WM_WindowColorFmtSet(const uint32_t             Unit,
                                    r_wm_Window_t       *const Window,
                                    const r_wm_WinColorFmt_t   ColorFmt)
```

Input Parameter

Table 4-46 Input parameter of R_WM_WindowColorFmtSet

Parameter	Description
Unit	Specifies the WM unit number.
ColorFmt	Specifies the window color format. 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

Table 4-47 Input-Output parameter of R_WM_WindowColorFmtSet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_COLORFMT	- Invalid color format.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function changes the color format of the window.
Color format should be changed to same bpp when window is enabled status. Otherwise, need to care about frame buffer size and pitch condition.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_COLORFMT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_WindowColorFmtSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t
r_wm_WinColorFmt_t

4.2.3.9 R_WM_WindowAlphaSet

Function Prototypes

```
r_wm_Error_t R_WM_WindowAlphaSet(const uint32_t         Unit,
                                 r_wm_Window_t   *const Window,
                                 const uint8_t          Alpha)
```

Input Parameter

Table 4-48 Input parameter of R_WM_WindowAlphaSet

Parameter	Description
Unit	Specifies the WM unit number.
Alpha	Specifies the new constant alpha value of the window. Range is 0 to 255

Input-Output Parameter

Table 4-49 Input-Output parameter of R_WM_WindowAlphaSet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function changes the constant alpha value of the window.
The alpha value multiplies the whole Pixel in the Window.

If specified window is assigned to layer 0, constant alpha is not effective.
If color format of specified window is YCbCr type, constant alpha is not effective.
If Chromakey feature is enabled, execution of this function will fail.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_ALPHA_SET_FAILED	- Error occurs in R_WM_Sys_WindowAlphaSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.10 R_WM_WindowPremultipliedAlphaEnable

Function Prototypes

```
r_wm_Error_t R_WM_WindowPremultipliedAlphaEnable(const uint32_t        Unit,
                                                 r_wm_Window_t  *const Window)
```

Input Parameter

Table 4-50 Input parameter of R_WM_WindowPremultipliedAlphaEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-51 Input-Output parameter of R_WM_WindowPremultipliedAlphaEnable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables the window's pre-multiplied alpha mode.

If specified window is assigned to layer 0, pre-multiplied alpha is not effective.
If color format of specified window is YCbCr type or no alpha channel format, pre-multiplied alpha value is not effective.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_ALPHA_SET_FAILED	- Error occurs in R_WM_Sys_WindowPremultipliedAlphaEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.11 R_WM_WindowPremultipliedAlphaDisable

Function Prototypes

```
r_wm_Error_t R_WM_WindowPremultipliedAlphaDisable(const uint32_t        Unit,
                                                  r_wm_Window_t  *const Window)
```

Input Parameter

Table 4-52 Input parameter of R_WM_WindowPremultipliedAlphaDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-53 Input-Output parameter of R_WM_WindowPremultipliedAlphaDisable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables the window's pre-multiplied alpha mode.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_ALPHA_SET_FAILED	- Error occurs in R_WM_Sys_WindowPremultipliedAlphaEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.12 R_WM_WindowVerticalMirrorEnable

Function Prototypes

```
r_wm_Error_t R_WM_WindowVerticalMirrorEnable(const uint32_t        Unit,
                                             r_wm_Window_t  *const Window)
```

Input Parameter

Table 4-54 Input parameter of R_WM_WindowVerticalMirrorEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-55 Input-Output parameter of R_WM_WindowVerticalMirrorEnable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables window's vertical mirroring.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_FLAG_UPDATE_FAILED	- Error occurs in R_WM_Sys_WindowFlagsUpdate.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.13 R_WM_WindowVerticalMirrorDisable

Function Prototypes

```
r_wm_Error_t R_WM_WindowVerticalMirrorDisable(const uint32_t        Unit,
                                              r_wm_Window_t  *const Window)
```

Input Parameter

Table 4-56 Input parameter of R_WM_WindowVerticalMirrorDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-57 Input-Output parameter of R_WM_WindowVerticalMirrorDisable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables window's vertical mirroring.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_FLAG_UPDATE_FAILED	- Error occurs in R_WM_Sys_WindowFlagsUpdate.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.14 R_WM_WindowSwap

Function Prototypes

```
r_wm_Error_t R_WM_WindowSwap(const uint32_t          Unit,
                             r_wm_Window_t     *const Window)
```

Input Parameter

Table 4-58 Input parameter of R_WM_WindowSwap

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-59 Input-Output parameter of R_WM_WindowSwap

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_PARAM_INCORRECT	- Parameter provided to a function is incorrect.
R_WM_ERR_NOT_FB_WINDOW	- Window type is Sprite window.
R_WM_ERR_WIN_SWAP_FAILED	- Failed in window swapping.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function swaps the window.

If the window is a multi-buffer window, the buffer whose status is Render Started is switched to the screen surface.
R_WM_WindowNewDrawBufGet should be called before using this function in case of multi frame buffer.
See *3.2.4.5* for the detail.

If the window is a single-buffer window, this function re-sets the frame buffer address holding with Window->Surface.Fb.Buffer[0].Data.
User can change the frame buffer address with modifying this parameter before this function is called.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_SWAP_FAILED	- Error occurs in R_WM_Sys_WindowSetFb.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t
r_wm_WinBufStatus_t

4.2.3.15 R_WM_WindowNewDrawBufGet

Function Prototypes

```
void *R_WM_WindowNewDrawBufGet(const uint32_t           Unit,
                               r_wm_Window_t     *const Window)
```

Input Parameter

Table 4-60 Input parameter of R_WM_WindowNewDrawBufGet

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-61 Input-Output parameter of R_WM_WindowNewDrawBufGet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

not R_NULL	- Address of the next frame buffer, which can be used for drawing operations.
R_NULL	- No free buffer is available or parameter is error.

Description

This function gives information about the next free buffer of the specified window.
This function also changes the buffer status. See *3.2.4.5* for the detail.
In normal sequence, this function should be called after asynchronous process of R_WM_WindowCreate is completed.

Following error code will be notified by error callback when error occurs in this function.

R_WM_ERR_INVALID_WM_UNIT	- WM unit is invalid.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_PARAM_INCORRECT	- Parameter is incorrect.
R_WM_ERR_NOT_FB_WINDOW	- Window type is Sprite window.

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Window_t

4.2.3.16 R_WM_WindowVisibleBufGet

Function Prototypes

```
void *R_WM_WindowVisibleBufGet(const uint32_t          Unit,
                               r_wm_Window_t  *const   Window)
```

Input Parameter

Table 4-62 Input parameter of R_WM_WindowVisibleBufGet

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-63 Input-Output parameter of R_WM_WindowVisibleBufGet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

not R_NULL - Address of the frame buffer which is currently on screen (or scheduled to be on screen).
R_NULL - On Screen buffer is not found or parameter is error.

Description

This function gives information about current On Screen buffer of the specified window.
This function is used by WM porting layer.

Following error code will be notified by error callback when error occurs in this function.
R_WM_ERR_INVALID_WM_UNIT - WM unit is invalid.
R_WM_ERR_NULL_PTR_ARGUMENT - Invalid null pointer.
R_WM_ERR_PARAM_INCORRECT - Parameter is incorrect.
R_WM_ERR_NOT_FB_WINDOW - Window type is Sprite window.

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Window_t

4.2.3.17 R_WM_WindowCurrentDrawBufGet

Function Prototypes

```
void *R_WM_WindowCurrentDrawBufGet(const uint32_t          Unit,
                                   r_wm_Window_t *const    Window)
```

Input Parameter

Table 4-64 Input parameter of R_WM_WindowCurrentDrawBufGet

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-65 Input-Output parameter of R_WM_WindowCurrentDrawBufGet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

not R_NULL	- Address of the frame buffer whose status is Render Started.
R_NULL	- Render Started buffer is not found or parameter is error.

Description

This function gives information about the current Render Started frame buffer of the specified window.

Following error code will be notified by error callback when error occurs in this function.

R_WM_ERR_INVALID_WM_UNIT	- WM unit is invalid.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_PARAM_INCORRECT	- Parameter is incorrect.
R_WM_ERR_NOT_FB_WINDOW	- Window type is Sprite window.

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Window_t

4.2.3.18 R_WM_WindowExternalBufSet

Function Prototypes

```
r_wm_Error_t R_WM_WindowExternalBufSet(const uint32_t             Unit,
                                       r_wm_Window_t       *const Window,
                                       r_wm_WinBuffer_t    *const Buf,
                                       const uint32_t             BufNum,
                                       const r_wm_WinColorFmt_t   ColorFormat)
```

Input Parameter

Table 4-66 Input parameter of R_WM_WindowExternalBufSet

Parameter	Description
Unit	Specifies the WM unit number.
BufNum	Specifies the number of buffers in the array.
ColorFormat	Specifies the window color format. 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 This parameter is used only to get bpp information.

Input-Output Parameter

Table 4-67 Input-Output parameter of R_WM_WindowExternalBufSet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.
Buf	Specifies the external buffer information.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_FB_WINDOW	- Window type is Sprite window.
R_WM_ERR_COLORFMT	- Invalid color format.
R_WM_ERR_PARAM_INCORRECT	- Parameter provided to a function is incorrect.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function dynamically changes the frame buffers for windows with externally allocated buffers.

If the window frame buffer was internal allocate mode, this function frees the all internal buffers and management area. Frame buffer allocate mode is changed to external mode.

One of the following status conditions shall be set as new external buffer.

- All buffer status are Free.
- One buffer status is Render Finished. The other buffer status are Free.

The instance of buffer information pointed by Buf must be hold until R_WM_WindowDelete is completed.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_FREE_FAILED	- Error occurs in R_WM_Sys_Free.
R_WM_ERR_SYS_WIN_EXTERNAL_BUF_SET_FAILED	- Error occurs in R_WM_Sys_WindowSetFb.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_WinColorFmt_t
r_wm_WinBuffer_t
r_wm_Window_t

4.2.3.19 R_WM_WindowColorKeyEnable

Function Prototypes

```
r_wm_Error_t R_WM_WindowColorKeyEnable(const uint32_t         Unit,
                                       r_wm_Window_t   *const Window)
```

Input Parameter

Table 4-68 Input parameter of R_WM_WindowColorKeyEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-69 Input-Output parameter of R_WM_WindowColorKeyEnable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables the chromakey (color keying) of the specified window.

Target color (before change) and replaced color (after change) should be set to window structure when R_WM_WindowCreate is called.

If chromakey is set to enable, the constant alpha setting will be disabled.
If specified window is assigned to layer 0, this function will fail.
The chromakey is supported when window color format is RGB type.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_WindowColorKeyEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.20 R_WM_WindowColorKeyDisable

Function Prototypes

```
r_wm_Error_t R_WM_WindowColorKeyDisable(const uint32_t         Unit,
                                        r_wm_Window_t   *const Window)
```

Input Parameter

Table 4-70 Input parameter of R_WM_WindowColorKeyDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-71 Input-Output parameter of R_WM_WindowColorKeyDisable

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables the chromakey (color keying) of the selected Window.
If user disables the chroma key, constant alpha settings will be effective.
If specified window is assigned to layer 0, this function will fail.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_WindowColorKeyEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.21 R_WM_WindowClutSet

Function Prototypes

```
r_wm_Error_t R_WM_WindowClutSet(const uint32_t               Unit,
                                r_wm_Window_t          *const Window,
                                const uint32_t               NumEntries,
                                const r_wm_ClutEntry_t *const Clut)
```

Input Parameter

Table 4-72 Input parameter of R_WM_WindowClutSet

Parameter	Description
Unit	Specifies the WM unit number.
NumEntries	Specifies the number of entries in the CLUT.
Clut	Specifies the pointer to CLUT. It must be 32 bits aligned.

Input-Output Parameter

Table 4-73 Input-Output parameter of R_WM_WindowClutSet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_RANGE_WM	- Parameter is the outside of the range.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_NOT_CLUT_WIN_FMT	- Invalid color format.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function sets color look-up table (CLUT).
The instance of CLUT must be hold until R_WM_FrameWait corresponding to this function is completed.

This function is applicable for the windows with CLUT color modes specified.

Table 4-74 Color Format applicable for windows

Color Format	Maximum NumEntries
R_WM_COLORFMT_CLUT8	256
R_WM_COLORFMT_CLUT4	16
R_WM_COLORFMT_CLUT1	2

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_WindowClutSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t
r_wm_ClutEntry_t

4.2.3.22 R_WM_WindowDeleteAllSprites

Function Prototypes

```
r_wm_Error_t R_WM_WindowDeleteAllSprites(const uint32_t        Unit,
                                         r_wm_Window_t  *const Window)
```

Input Parameter

Table 4-75 Input parameter of R_WM_WindowDeleteAllSprites

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-76 Input-Output parameter of R_WM_WindowDeleteAllSprites

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_SPRITE_WINDOW	- Window is not sprite window.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables and deletes all Sprite data associated to the window.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Internal sprite chain is broken.
R_WM_ERR_SYS_WIN_DELETE_ALL_SPRITES_FAILED	- Error occurs in R_WM_Sys_WindowDeleteAllSprites.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.
See ***Table 2-10*** about Sprite status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.3.23 R_WM_WindowScaledSizeSet

Function Prototypes

```
r_wm_Error_t R_WM_WindowScaledSizeSet(const uint32_t         Unit,
                                      r_wm_Window_t   *const Window,
                                      const uint32_t         ScaledWidth,
                                      const uint32_t         ScaledHeight)
```

Input Parameter

Table 4-77 Input parameter of R_WM_WindowScaledSizeSet

Parameter	Description
Unit	Specifies the WM unit number.
ScaledWidth	Specifies the original width to scaling-up in pixels. When scaling-up is disabled, set to 0. When scaling-up is enabled, range is 4 to (Width-1).
ScaledHeight	Specifies the original height to scaling-up in pixels. When scaling-up is disabled, set to 0. When scaling-up is enabled, range is 4 to (Height-1).

Input-Output Parameter

Table 4-78 Input-Output parameter of R_WM_WindowScaledSizeSet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function changes the scaled size of the window.
The window can be dynamically resized until the life time of the window.
See ***Figure 3-15*** for the detail.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SYS_WIN_SCALED_SET_FAILED	- Error occurs in R_WM_Sys_WindowScaleSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-8*** about Window status conditions.

See also

r_wm_Error_t
r_wm_Window_t

4.2.4 Message Queue functions

4.2.4.1 R_WM_FrameEndMark

Function Prototypes

```
r_wm_Error_t R_WM_FrameEndMark(const uint32_t    Unit,
                               const uint16_t    Id)
```

Input Parameter

Table 4-79 Input parameter of R_WM_FrameEndMark

Parameter	Description
Unit	Specifies the WM unit number.
Id	Specifies the ID of the frame up to which all the requests should be executed.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM driver is not initialized.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function marks end for message queue with specified Id.
User should adjust this function to call only once during 1 V-sync period when message queue is executed with R_WM_FrameExecuteNext.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t

4.2.4.2 R_WM_FrameWait

Function Prototypes

```
r_wm_Error_t R_WM_FrameWait(const uint32_t  Unit,
                            const uint16_t  Id)
```

Input Parameter

Table 4-80 Input parameter of R_WM_FrameWait

Parameter	Description
Unit	Specifies the WM unit number.
Id	Specifies the ID of the frame up to which all the requests should be executed.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM driver is not initialized.
R_WM_ERR_NG	- Error occurs while executing a job in the message queue.

Description

This function waits for the scanline interrupt.
Execute jobs in message queue until Id of specified at R_WM_FrameEndMark.
Then this function waits for V-sync interrupt.

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
R_WM_FrameEndMark should be executed with same “Id” before this function is called.

See also

r_wm_Error_t

4.2.4.3 R_WM_FrameExecuteNext

Function Prototypes

```
uint32_t R_WM_FrameExecuteNext(const uint32_t   Unit)
```

Input Parameter

Table 4-81 Input parameter of R_WM_FrameExecuteNext

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

0 ~ 0xFFFF	- The encountered frame-end ID.
0x80000000u	- Error occurs during job execution.
0x80000000u | R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
0x80000000u | R_WM_ERR_NOT_INITIALIZED	- WM driver is not initialized.
0x80000000u | R_WM_ERR_EVENT_FAILED	- Received an unexpected number of messages.

Description

This function executes jobs in message queue.
R_WM_FrameExecuteNext doesn't wait for any interrupt.
If R_WM_FrameExecuteNext function is used, user should control similar to R_WM_FrameWait with using interrupt.

When the job of R_WM_FrameEndMark is found, update processing for the next frame is executed.
User should adjust this update process so that it will be executed slightly before V-sync event.
It should be especially careful when there is control over Sprite, RLE, and CLUT.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue
- R_VDCE_Sys_Lock
- R_VDCE_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

None

4.2.4.4 R_WM_FrameExecuteVoca

Function Prototypes

```
r_wm_Error_t R_WM_FrameExecuteVoca(const uint32_t   Unit)
```

Input Parameter

Table 4-82 Input parameter of R_WM_FrameExecuteVoca

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM driver is not initialized.
R_WM_ERR_NG	- Error occurs while executing a job in the message queue.

Description

This function executes jobs for VOCA in message queue.
This function is part of R_WM_FrameWait process.
So, user doesn't need to call this function when using R_WM_FrameWait.

If this function is used, user should control similar to R_WM_FrameWait with using interrupt.
See *3.2.9.3* for the detail.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue
- R_VOCA_Sys_Lock
- R_VOCA_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
R_WM_FrameExecuteNext should be called to the target message queue before this function is called.

See also

r_wm_Error_t

4.2.4.5 R_WM_FrameExecuteDiscom

Function Prototypes

```
r_wm_Error_t R_WM_FrameExecuteDiscom(const uint32_t   Unit)
```

Input Parameter

Table 4-83 Input parameter of R_WM_FrameExecuteDiscom

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NOT_INITIALIZED	- WM driver is not initialized.
R_WM_ERR_NG	- Error occurs while executing a job in the message queue.

Description

This function executes jobs for DISCOM in message queue.
This function is part of R_WM_FrameWait process.
So, user doesn't need to call this function when using R_WM_FrameWait.

If this function is used, user should control similar to R_WM_FrameWait with using interrupt.
See *3.2.9.3* for the detail.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue
- R_DISCOM_Sys_Lock
- R_DISCOM_Sys_Unlock

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
R_WM_FrameExecuteNext should be called to the target message queue before this function is called.

See also

r_wm_Error_t

4.2.5 Sprites functions

4.2.5.1 R_WM_SpriteCreate

Function Prototypes

```
r_wm_Error_t R_WM_SpriteCreate(const uint32_t          Unit,
                               r_wm_Sprite_t    *const Sprite)
```

Input Parameter

Table 4-84 Input parameter of R_WM_SpriteCreate

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-85 Input-Output parameter of R_WM_SpriteCreate

Parameter	Description
Sprite	Specifies the sprite data structure. The contents of the structure are updated by WM, user does not need to refer to the contents of the output.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_SPRITE_WINDOW	- Window type is not sprite window.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function creates a Sprite data to sprite window.

The Sprite data will not be visible until it is enabled.
Up to 32 Sprite data can be created per window, but up to 16 Sprite data can be visible per window in descending order of Sprite->PosZ.

The instance of sprite structure must be hold until R_WM_SpriteDelete or R_WM_WindowDeleteAllSprites is completed.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SPRITE_NOT_FOUND	- Available Sprite data is not remained.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_CaptureCreate .
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-10*** about Sprite status conditions.
R_WM_WindowCreate should call before this function.

See also

r_wm_Error_t
r_wm_Sprite_t

4.2.5.2 R_WM_SpriteEnable

Function Prototypes

```
r_wm_Error_t R_WM_SpriteEnable(const uint32_t              Unit,
                               r_wm_Sprite_t         *const Sprite)
```

Input Parameter

Table 4-86 Input parameter of R_WM_SpriteEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-87 Input-Output parameter of R_WM_SpriteEnable

Parameter	Description
Sprite	Specifies the pointer of sprite structure generated by R_WM_SpriteCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables the specified Sprite data.
The Sprite data becomes visible on the screen.
Up to 16 Sprite data can be visible per window in descending order of Sprite->PosZ.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_SpriteEnable .
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-10*** about Sprite status conditions.

See also

r_wm_Error_t
r_wm_Sprite_t

4.2.5.3 R_WM_SpriteDisable

Function Prototypes

```
r_wm_Error_t R_WM_SpriteDisable(const uint32_t            Unit,
                                r_wm_Sprite_t       *const Sprite)
```

Input Parameter

Table 4-88 Input parameter of R_WM_SpriteDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-89 Input-Output parameter of R_WM_SpriteDisable

Parameter	Description
Sprite	Specifies the pointer of sprite structure generated by R_WM_SpriteCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables the specified Sprite.
The Sprite data becomes not visible on the screen.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_SpriteEnable .
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-10*** about Sprite status conditions.

See also

r_wm_Error_t
r_wm_Sprite_t

4.2.5.4 R_WM_SpriteMove

Function Prototypes

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

Input Parameter

Table 4-90 Input parameter of R_WM_SpriteMove

Parameter	Description
Unit	Specifies the WM unit number.
PosX	Specifies the new Sprite X position. Unit is pixels.
PosY	Specifies the new Sprite Y position. Unit is pixels.
PosZ	Specifies the new Sprite Z position.

Input-Output Parameter

Table 4-91 Input-Output parameter of R_WM_SpriteMove

Parameter	Description
Sprite	Specifies the pointer of sprite structure generated by R_WM_SpriteCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function moves the Sprite to the specified X/Y and Z-order location.
See *Table 3-29* about the range and alignment information.
Z position is the relative position of the Sprite data in the sprite window. The one with the larger value takes precedence.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SPRITE_NOT_FOUND	- Specified sprite data is not found.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_SpriteMove.
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-10*** about Sprite status conditions.

See also

r_wm_Error_t
r_wm_Sprite_t

4.2.5.5 R_WM_SpriteBufSet

Function Prototypes

```
r_wm_Error_t R_WM_SpriteBufSet(const uint32_t          Unit,
                               r_wm_Sprite_t    *const Sprite,
                               const void       *const Buffer)
```

Input Parameter

Table 4-92 Input parameter of R_WM_SpriteBufSet

Parameter	Description
Unit	Specifies the WM unit number.
Buffer	Specifies the new address stored the sprite source data. The address should be 8 Bytes alignment.

Input-Output Parameter

Table 4-93 Input-Output parameter of R_WM_SpriteBufSet

Parameter	Description
Sprite	Specifies the pointer of sprite structure generated by R_WM_SpriteCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function changes the Sprite data source address.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_SpriteBufSet.
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-10*** about Sprite status conditions.

See also

r_wm_Error_t
r_wm_Sprite_t

4.2.5.6 R_WM_SpriteDelete

Function Prototypes

```
r_wm_Error_t R_WM_SpriteDelete(const uint32_t          Unit,
                               r_wm_Sprite_t    *const Sprite)
```

Input Parameter

Table 4-94 Input parameter of R_WM_SpriteDelete

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-95 Input-Output parameter of R_WM_SpriteDelete

Parameter	Description
Sprite	Specifies the pointer of sprite structure generated by R_WM_SpriteCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function removes the specified Sprite data from the host window.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_SPRITE_NOT_FOUND	- Specified sprite data is not found.
R_WM_ERR_NG	- Error occurs in R_WM_Sys_SpriteDelete.
R_WM_ERR_SPEA_INTERNAL	- Error occurs in SPEA driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-10*** about Sprite status conditions.

See also

r_wm_Error_t
r_wm_Sprite_t

4.2.6 Video Capture functions

The window manager offers the possibility to create surfaces for the video capturing.

4.2.6.1 R_WM_CaptureCreate

Function Prototypes

```
r_wm_Error_t R_WM_CaptureCreate(const uint32_t          Unit,
                                r_wm_Capture_t   *const Capture)
```

Input Parameter

Table 4-96 Input parameter of R_WM_CaptureCreate

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-97 Input-Output parameter of R_WM_CaptureCreate

Parameter	Description
Capture	Specifies a Capture structure. The contents of the structure are updated by WM, user does not need to refer to the contents of the output.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Window type is invalid.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function creates a video capture surface inside a specific frame buffer type window.
For each capture unit, only one video capture surface can be created.

When video capture is used, following window color format is supported.

- R_WM_COLORFMT_RGB0888
- R_WM_COLORFMT_ARGB8888
- R_WM_COLORFMT_RGB565

The instance of capture structure must be hold until R_WM_CaptureDelete is completed.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_CAPTURE_UNIT_COUNT_EXCEEDED	- Available capture surface is not remained.
R_WM_ERR_SYS_CAPTURE_CREATE_FAILED	- Error occurs in R_WM_Sys_CaptureCreate .
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.
R_WM_WindowCreate should be called before this function is called.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.6.2 R_WM_CaptureDelete

Function Prototypes

```
r_wm_Error_t R_WM_CaptureDelete(const uint32_t            Unit,
                                r_wm_Capture_t      *const Capture)
```

Input Parameter

Table 4-98 Input parameter of R_WM_CaptureDelete

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-99 Input-Output parameter of R_WM_CaptureDelete

Parameter	Description
Capture	Specifies the pointer of capture structure generated by R_WM_CaptureCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function deletes the specified capturing surface.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Capture surface is not found.
R_WM_ERR_SYS_CAPTURE_DELETE_FAILED	- Error occurs in R_WM_Sys_CaptureDelete.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.6.3 R_WM_CaptureEnable

Function Prototypes

```
r_wm_Error_t R_WM_CaptureEnable(const uint32_t            Unit,
                                r_wm_Capture_t      *const Capture)
```

Input Parameter

Table 4-100 Input parameter of R_WM_CaptureEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-101 Input-Output parameter of R_WM_CaptureEnable

Parameter	Description
Capture	Specifies the pointer of capture structure generated by R_WM_CaptureCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables the video capturing surface and start the video capturing.
R_WM_WindowEnable is also needed to control window during capturing.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Capture surface is not found.
R_WM_ERR_SYS_CAPTURE_ENABLE_FAILED	- Error occurs in R_WM_Sys_CaptureEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.6.4 R_WM_CaptureDisable

Function Prototypes

```
r_wm_Error_t R_WM_CaptureDisable(const uint32_t            Unit,
                                 r_wm_Capture_t      *const Capture)
```

Input Parameter

Table 4-102 Input parameter of R_WM_CaptureDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-103 Input-Output parameter of R_WM_CaptureDisable

Parameter	Description
Capture	Specifies the pointer of capture structure generated by R_WM_CaptureCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables the video capturing.
After this function call, the last frame will remain in the frame buffer.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Capture surface is not found.
R_WM_ERR_SYS_CAPTURE_ENABLE_FAILED	- Error occurs in R_WM_Sys_CaptureEnable.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.6.5 R_WM_Cap_CapBufGet

Function Prototypes

```
r_wm_WinBuffer_t *R_WM_Cap_CapBufGet(const uint32_t           Unit,
                                     r_wm_Window_t  *const    Window)
```

Input Parameter

Table 4-104 Input parameter of R_WM_Cap_CapBufGet

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-105 Input-Output parameter of R_WM_Cap_CapBufGet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

not R_NULL - Pointer to frame buffer information of Free buffer.
R_NULL - Free buffer is not found or parameter is error.

Description

This function gets the Free buffer to be assigned for video capture buffer.
This function is called from WM porting layer.
This function changes the buffer status. See ***Figure 3-30*** for the detail.

Following error code will be notified by error callback when error occurs in this function.
R_WM_ERR_INVALID_WM_UNIT - WM unit is invalid.
R_WM_ERR_NULL_PTR_ARGUMENT - Invalid null pointer.
R_WM_ERR_PARAM_INCORRECT - Parameter is incorrect.
R_WM_ERR_NOT_FB_WINDOW - Window type is Sprite window.

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Permitted.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Window_t

4.2.6.6 R_WM_Cap_DispBufGet

Function Prototypes

```
r_wm_WinBuffer_t *R_WM_Cap_DispBufGet(const uint32_t         Unit,
                                      r_wm_Window_t  *const  Window)
```

Input Parameter

Table 4-106 Input parameter of R_WM_Cap_DispBufGet

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-107 Input-Output parameter of R_WM_Cap_DispBufGet

Parameter	Description
Window	Specifies the pointer of window structure generated by R_WM_WindowCreate. It is not necessary for user to change the value of the window structure.

Output Parameter

None

Return Codes

not R_NULL - Pointer to frame buffer information of Render finished buffer.
R_NULL - Render finished buffer is not found or parameter is error.

Description

This function gets the Render finished buffer to be assigned for on-screen buffer.
This function is called from WM porting layer.
This function changes the buffer status. See ***Figure 3-30*** for the detail.

Following error code will be notified by error callback when error occurs in this function.
R_WM_ERR_INVALID_WM_UNIT - WM unit is invalid.
R_WM_ERR_NULL_PTR_ARGUMENT - Invalid null pointer.
R_WM_ERR_PARAM_INCORRECT - Parameter is incorrect.
R_WM_ERR_NOT_FB_WINDOW - Window type is Sprite window.

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Permitted .

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Window_t

4.2.6.7 R_WM_CaptureMove

Function Prototypes

```
r_wm_Error_t R_WM_CaptureMove(const uint32_t            Unit,
                              r_wm_Capture_t      *const Capture
                              const uint32_t            StartX,
                              const uint32_t            StartY1)
```

Input Parameter

Table 4-108 Input parameter of R_WM_CaptureMove

Parameter	Description
Unit	Specifies the WM unit number.
StartX	X position of capturing start. Unit is pixels. This is same as StartX in r_wm_Capture_t.
StartY1	Y position of capturing start. Unit is pixels. This is same as StartY1 in r_wm_Capture_t.

Input-Output Parameter

Table 4-109 Input-Output parameter of R_WM_CaptureMove

Parameter	Description
Capture	Specifies the pointer of capture structure generated by R_WM_CaptureCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function moves the capturing start position.
Dynamic changes can cause momentary screen disruptions. To avoid this, change to Disabled state by R_WM_CaptureDisable and then call this function.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Capture surface is not found.
R_WM_ERR_SYS_CAPTURE_MOVE_FAILED	- Error occurs in R_WM_Sys_CaptureViewPortSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.6.8 R_WM_CaptureResize

Function Prototypes

```
r_wm_Error_t R_WM_CaptureResize(const uint32_t            Unit,
                                r_wm_Capture_t      *const Capture
                                const uint32_t            Width,
                                const uint32_t            Height)
```

Input Parameter

Table 4-110 Input parameter of R_WM_CaptureResize

Parameter	Description
Unit	Specifies the WM unit number.
Width	Width of capturing video data. Unit is pixels. This is same as Width in r_wm_Capture_t.
Height	Height of capturing video data. Unit is pixels. This is same as Height in r_wm_Capture_t.

Input-Output Parameter

Table 4-111 Input-Output parameter of R_WM_CaptureResize

Parameter	Description
Capture	Specifies the pointer of capture structure generated by R_WM_CaptureCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function resizes the capturing signal size.
If scaling-down is disabled, Width and Height is also capture buffer size.
Dynamic changes can cause momentary screen disruptions. To avoid this, change to Disabled state by R_WM_CaptureDisable and then call this function.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Capture surface is not found.
R_WM_ERR_SYS_CAPTURE_RESIZE_FAILED	- Error occurs in R_WM_Sys_CaptureViewPortSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.6.9 R_WM_CaptureScaledSizeSet

Function Prototypes

```
r_wm_Error_t R_WM_CaptureScaledSizeSet(const uint32_t            Unit,
                                       r_wm_Capture_t      *const Capture
                                       const uint32_t            ScaledWidth,
                                       const uint32_t            ScaledHeight)
```

Input Parameter

Table 4-112 Input parameter of R_WM_CaptureScaledSizeSet

Parameter	Description
Unit	Specifies the WM unit number.
ScaledWidth	Frame buffer width when scaling-down. Unit is pixels. This is same as ScaledWidth in r_wm_Capture_t.
ScaledHeight	Frame buffer height when scaling-down. Unit is pixels. This is same as ScaledHeight in r_wm_Capture_t.

Input-Output Parameter

Table 4-113 Input-Output parameter of R_WM_CaptureScaledSizeSet

Parameter	Description
Capture	Specifies the pointer of capture structure generated by R_WM_CaptureCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function resizes the scaling-down size.
See ***Figure 3-25*** for the detail.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Capture surface is not found.
R_WM_ERR_SYS_WIN_SCALED_SET_FAILED	- Error occurs in R_WM_Sys_WindowScaleSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.6.10 R_WM_CaptureExtVsyncSet

Function Prototypes

```
r_wm_Error_t R_WM_CaptureExtVsyncSet(const uint32_t            Unit,
                                     r_wm_Capture_t      *const Capture,
                                     const uint16_t            HsyncCycle,
                                     const uint32_t            VsyncMaskUs,
                                     const uint32_t            VsyncLackUs)
```

Input Parameter

Table 4-114 Input parameter of R_WM_CaptureExtVsyncSet

Parameter	Description
Unit	Specifies the WM unit number.
HsyncCycle	Horizontal cycle of input signal. Unit is cycle (pixel). The range is 4 to 2046. If 0 is set, WM driver sets automatically.
VsyncMaskUs	Prevent Vsync coming faster than VsyncMaskUs, Unit is microseconds. If non-0 value is set, vsync masking is enabled. If 0 is set, vsync masking is disabled.
VsyncLackUs	Compensate Vsync coming slower than VsyncLackUs. Unit is microseconds. If non-0 value is set, vsync compensation is enabled. If 0 is set, vsync compensation is disabled.

Input-Output Parameter

Table 4-115 Input-Output parameter of R_WM_CaptureExtVsyncSet

Parameter	Description
Capture	Specifies the pointer of capture structure generated by R_WM_CaptureCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function sets the Hsync cycle of input signal and Vsync protection.

Hsync cycle is used to determine the field of the interlace signal.
In case of NTSC signal input, the default value is 1716.
In case of PAL signal input, the default value is 1728.
If HsyncCycle is set as 0 or this function is not called, the default value is selected automatically.
NTSC or PAL is selected by R_WM_CAPMODE_PAL flag.

Vsync protections are effective when external Vsync synchronization is selected (R_WM_CAPMODE_FIXED_VSYNC is off). Vsync protections are disabled by default.
Vsync protection timing is made by PixelClock set by R_WM_ScreenTimingSet.
The maximum value of VsyncMaskUs and VsyncLackUs is (8,388,480 / PixelClock[MHz])[usec]
If setting value is over the range, this function assumes that the maximum value is set.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	- Capture surface is not found.
R_WM_ERR_SYS_CAPTURE_SET_VSYNC_FAILED	- Error occurs in R_WM_Sys_CaptureExtVsyncSet.
R_WM_ERR_VOUT_INTERNAL	- Error occurs in VDCE driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-12*** about Capture surface status conditions.

See also

r_wm_Error_t
r_wm_Capture_t

4.2.7 General functions

4.2.7.1 R_WM_ErrorCallbackSet

Function Prototypes

```
r_wm_Error_t R_WM_ErrorCallbackSet(const uint32_t Unit,
                                   void    (*ErrorCb)(uint32_t Unit, r_wm_Error_t Error))
```

Input Parameter

Table 4-116 Input parameter of R_WM_ErrorCallbackSet

Parameter	Description
Unit	Specifies the WM unit number.
ErrorCb	Specifies the error callback function. Set R_NULL when the callback is uninstalled.

Table 4-117 Input parameter of ErrorCb

Parameter	Description
Unit	WM, VDCE, VOCA or SPEA unit number where the error occurred.
Error	Error code.

Input-Output Parameter

None

Output Parameter

None

Return Codes

R_WM_ERR_OK - No error occurred.
R_WM_ERR_INVALID_WM_UNIT - Unit number is outside the range.
R_WM_ERR_RANGE_WM - Parameter is the outside of the range.
R_WM_ERR_NOT_INITIALIZED - WM unit status is invalid.

Description

This function sets the error callback function.
Callback is notified when an error occurs in WM, VDCE or SPEA driver.
Because this callback is the only way to get asynchronous control error details, it is recommended to set the callback.

Regarding the callback parameter Unit and Error, Unit has different meaning depending on Error.

Error	Unit	Note
R_WM_ERR_VOUT_INTERNAL	VDCE unit number	For capture control across two VDCE units, the VDCE unit may not match the WM unit.
R_WM_ERR_SPEA_INTERNAL	SPEA unit number	It is always 0.
R_WM_ERR_VOCA_INTERNAL	VOCA unit number	It is always 0.
Others	WM unit number	Following error is notified to Unit = 0. - WM Unit is error - Error of function without WM unit in argument

Callback with Unit =0 is notified to the function installed to Unit = 0.
Callback with Unit =1 is notified to the function installed to Unit = 1.

Error callback is notified during WM unit is not de-initialized state.
The installed error callback can be uninstalled by R_WM_DevInit or R_NULL setting by this function.

Reentrancy

Non-reentrant.

Sync/Async

Synchronous

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t

4.2.7.2 R_WM_ErrorHandler

Function Prototypes

```
void R_WM_ErrorHandler(const uint32_t         Unit,
                       const r_wm_Error_t     Error)
```

Input Parameter

Table 4-118 Input parameter of R_WM_ErrorHandler

Parameter	Description
Unit	Specifies the WM unit number.
Error	Specifies the error type.

Input-Output Parameter

None

Output Parameter

None

Return Codes

None

Description

This function is the driver's central error handler.
This function is internal functions. Application does not need to use this function.
WM Porting layer also calls this function to execute error callback set by R_WM_ErrorCallbackSet.

Reentrancy

Reentrant.

Sync/Async

Synchronous

Call from Interrupt

Permitted.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_Error_t

4.2.7.3 R_WM_ColorFmtBitsPerPixGet

Function Prototypes

```
uint8_t R_WM_ColorFmtBitsPerPixGet(const r_wm_WinColorFmt_t  Format)
```

Input Parameter

Table 4-119 Input parameter of R_WM_ColorFmtBitsPerPixGet

Parameter	Description
Format	Specifies the color format. 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

Return Codes

not 0 - Bits per pixel of the specified format.
0 - Unsupported format is specified.

Description

This function returns the bits per pixel (1 - 32) for the specified format.

Reentrancy

Reentrant.

Sync/Async

Synchronous

Call from Interrupt

Permitted.

Preconditions

See ***Table 2-6*** about WM unit status conditions.

See also

r_wm_WinColorFmt_t

4.2.8 Discom functions

4.2.8.1 R_WM_DiscomCreate

Function Prototypes

```
r_wm_Error_t R_WM_DiscomCreate(const uint32_t          Unit,
                               r_wm_Discom_t    *const Discom)
```

Input Parameter

Table 4-120 Input parameter of R_WM_DiscomCreate

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-121 Input-Output parameter of R_WM_DiscomCreate

Parameter	Description
Discom	Specifies the DISCOM device structure. The contents of the structure are updated by WM, user does not need to refer to the contents of the output.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function creates a DISCOM device to WM unit.
A DISCOM device can be created per one DISCOM Unit.
Two DISCOM devices can be created per one WM unit.

The instance of r_wm_Discom_t structure must be hold until R_WM_DiscomDelete is completed.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_DISCOM_NOT_FOUND	- Specified Discom unit is already created.
R_WM_ERR_SYS_DISCOM_CREATE_FAILED	- Error occurs in R_WM_Sys_DiscomCreate.
R_WM_ERR_DISCOM_INTERNAL	- Error occurs in DISCOM driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-16*** about DISCOM device status conditions.

See also

r_wm_Error_t
r_wm_Discom_t

4.2.8.2 R_WM_DiscomDelete

Function Prototypes

```
r_wm_Error_t R_WM_DiscomDelete(const uint32_t          Unit,
                               r_wm_Discom_t    *const Discom)
```

Input Parameter

Table 4-122 Input parameter of R_WM_DiscomDelete

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-123 Input-Output parameter of R_WM_DiscomDelete

Parameter	Description
Discom	Specifies the pointer of DISCOM device generated by R_WM_DiscomCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function deletes a DISCOM device.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_DISCOM_NOT_FOUND	- DISCOM device is not found.
R_WM_ERR_SYS_DISCOM_DELETE_FAILED	- Error occurs in R_WM_Sys_DiscomDelete.
R_WM_ERR_DISCOM_INTERNAL	- Error occurs in DISCOM driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-16*** about DISCOM device status conditions.

See also

r_wm_Error_t
r_wm_Discom_t

4.2.8.3 R_WM_DiscomEnable

Function Prototypes

```
r_wm_Error_t R_WM_DiscomEnable(const uint32_t          Unit,
                               r_wm_Discom_t    *const Discom)
```

Input Parameter

Table 4-124 Input parameter of R_WM_DiscomEnable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-125 Input-Output parameter of R_WM_DiscomEnable

Parameter	Description
Discom	Specifies the pointer of DISCOM device generated by R_WM_DiscomCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function enables a DISCOM device.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_DISCOM_NOT_FOUND	- DISCOM device is not found.
R_WM_ERR_SYS_DISCOM_ENABLE_FAILED	- Error occurs in R_WM_Sys_DiscomEnable.
R_WM_ERR_DISCOM_INTERNAL	- Error occurs in DISCOM driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-16*** about DISCOM device status conditions.

See also

r_wm_Error_t
r_wm_Discom_t

4.2.8.4 R_WM_DiscomDisable

Function Prototypes

```
r_wm_Error_t R_WM_DiscomDisable(const uint32_t          Unit,
                                r_wm_Discom_t    *const Discom)
```

Input Parameter

Table 4-126 Input parameter of R_WM_DiscomDisable

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-127 Input-Output parameter of R_WM_DiscomDisable

Parameter	Description
Discom	Specifies the pointer of DISCOM device generated by R_WM_DiscomCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function disables a DISCOM device.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_DISCOM_NOT_FOUND	- DISCOM device is not found.
R_WM_ERR_SYS_DISCOM_ENABLE_FAILED	- Error occurs in R_WM_Sys_DiscomEnable.
R_WM_ERR_DISCOM_INTERNAL	- Error occurs in DISCOM driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-16*** about DISCOM device status conditions.

See also

r_wm_Error_t
r_wm_Discom_t

4.2.8.5 R_WM_DiscomCrcSet

Function Prototypes

```
r_wm_Error_t R_WM_DiscomCrcSet(const uint32_t          Unit,
                               r_wm_Discom_t    *const Discom,
                               const uint32_t          ExpCrc)
```

Input Parameter

Table 4-128 Input parameter of R_WM_DiscomCrcSet

Parameter	Description
Unit	Specifies the WM unit number.
ExpCrc	Specifies the expected CRC.

Input-Output Parameter

Table 4-129 Input-Output parameter of R_WM_DiscomCrcSet

Parameter	Description
Discom	Specifies the pointer of DISCOM device generated by R_WM_DiscomCreate.

Output Parameter

None

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	- Message queue is overflow.

Description

This function changes expected CRC of DISCOM device.

Following error code will be notified by error callback when error occurs asynchronously.

R_WM_ERR_NULL_PTR_ARGUMENT	- Message queue is broken.
R_WM_ERR_DISCOM_NOT_FOUND	- DISCOM device is not found.
R_WM_ERR_SYS_DISCOM_SET_FAILED	- Error occurs in R_WM_Sys_DiscomCrcSet.
R_WM_ERR_DISCOM_INTERNAL	- Error occurs in DISCOM driver.

Reentrancy

Non-reentrant as default.
If user implements following functions to prevent multiple executions, this function will become re-entrant.

- R_WM_Sys_LockMsgQueue
- R_WM_Sys_UnLockMsgQueue

Sync/Async

Asynchronous.
This command is first queued in message queue and later executed by R_WM_FrameWait or R_WM_FrameExecuteNext.

Call from Interrupt

Prohibited.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-16*** about DISCOM device status conditions.

See also

r_wm_Error_t
r_wm_Discom_t

4.2.8.6 R_WM_DiscomCrcGet

Function Prototypes

```
r_wm_Error_t R_WM_DiscomCrcGet(const uint32_t          Unit,
                               r_wm_Discom_t    *const Discom,
                               uint32_t         *const Crc)
```

Input Parameter

Table 4-130 Input parameter of R_WM_DiscomCrcGet

Parameter	Description
Unit	Specifies the WM unit number.

Input-Output Parameter

Table 4-131 Input-Output parameter of R_WM_DiscomCrcGet

Parameter	Description
Discom	Specifies the pointer of DISCOM device generated by R_WM_DiscomCreate.

Output Parameter

Table 4-132 Output parameter of R_WM_DiscomCrcGet

Parameter	Description
Crc	The latest calculated CRC value.

Return Codes

R_WM_ERR_OK	- No error occurred.
R_WM_ERR_INVALID_WM_UNIT	- Unit number is outside the range.
R_WM_ERR_NULL_PTR_ARGUMENT	- Invalid null pointer.
R_WM_ERR_NOT_INITIALIZED	- WM unit status is invalid.
R_WM_ERR_SYS_DISCOM_GET_FAILED	- Error occurs in R_WM_Sys_DiscomCrcGet.

Description

This function gets the latest calculated CRC value.

Reentrancy

Reentrant.

Sync/Async

Synchronous.

Call from Interrupt

Permitted.

Preconditions

See ***Table 2-6*** about WM unit status conditions.
See ***Table 2-16*** about DISCOM device status conditions.

See also

r_wm_Error_t
r_wm_Discom_t

5.Types

5.1 Basic Types

This section shows the basic types used on this library.

Table 5-1 Basic type

Types	Definition		Basic types
char_t	typedef char	char_t	signed char
int8_t	typedef signed char	int8_t	signed char
int16_t	typedef signed short	int16_t	signed short
int32_t	typedef signed int	int32_t	signed int
int64_t	typedef signed long long	int64_t	signed long long
uint8_t	typedef unsigned char	uint8_t	unsigned char
uint16_t	typedef unsigned short	uint16_t	unsigned short
uint32_t	typedef unsigned int	uint32_t	unsigned int
uint64_t	typedef unsigned long long	uint64_t	unsigned long long
float32_t	typedef float	float32_t	float
float64_t	typedef double	float64_t	double

5.2 Definition

This section shows the definitions used in WM API.

5.2.1 API Version

This constant is the value which shows the version information of the WM driver.

Table 5-2 Definition of WM API version

Name	Description
R_WM_VERSION_HI	MSB byte of the version information. It is major version information. This value is changed with release version.
R_WM_VERSION_LO	LSB byte of the version information. It is miner version information. This value is changed with release version.

5.2.2 VOCA CLUT entry number

This constant is the value which shows the number of VOCA CLUT entries of the WM driver.

Table 5-3 Definition of VOCA CLUT entry

Name	Values	Description
R_WM_VOCA_CLUT_NUM	4u	The number of VOCA CLUT entries

5.3 Enumerated Type

This section shows the enumerated types used in WM API Function.

5.3.1 r_wm_Error_t

Description

WM driver error code.
If an error occurs, these enumerations give information about the reason.

Definition

```
typedef enum
{
    R_WM_ERR_OK                                 = 0,
    R_WM_ERR_NG                                 = 1,
    R_WM_ERR_RANGE_WM                           = 2,
    R_WM_ERR_PARAM_INCORRECT                    = 3,
    R_WM_UNIT_LOCKED                            = 4,
    R_WM_UNIT_NOTLOCKED                         = 5,
    R_WM_ERR_NO_PHYS_WINDOW                     = 6,
    R_WM_ERR_MALLOC_FAILED                      = 7,
    R_WM_ERR_FREE_FAILED                        = 8,
    R_WM_ERR_EVENT_FAILED                       = 9,
    R_WM_ERR_VOUT                               = 10,
    R_WM_ERR_NOT_DISABLED                       = 11,
    R_WM_ERR_NOT_DELETED                        = 12,
    R_WM_ERR_COLORFMT                           = 13,
    R_WM_ERR_VIN                                = 14,
    R_WM_ERR_NOT_SPRITE_WINDOW                  = 15,
    R_WM_ERR_DISPLAY_TIMING_SET                 = 16,
    R_WM_ERR_INVALID_WM_UNIT                    = 17,
    R_WM_ERR_DEV_DEINIT_FAILED                  = 18,
    R_WM_ERR_NULL_PTR_ARGUMENT                  = 19,
    R_WM_ERR_SPRITE_NOT_FOUND                   = 20,
    R_WM_ERR_SYS_LAYER_INIT_FAILURE             = 21,
    R_WM_ERR_NOT_CLUT_WIN_FMT                   = 22,
    R_WM_ERR_NO_SUCH_WINDOW                     = 23,
    R_WM_ERR_COULD_NOT_ENABLE_SCREEN            = 24,
    R_WM_ERR_COULD_NOT_DISABLE_SCREEN           = 25,
    R_WM_ERR_COULD_NOT_SET_SCREEN_BG_COLOR      = 26,
    R_WM_ERR_COULD_NOT_REGISTER_EVENT           = 27,
    R_WM_ERR_NOT_FB_WINDOW                      = 28,
    R_WM_ERR_SYNC_MODE_NOT_POSSIBLE             = 29,
    R_WM_ERR_SYS_WIN_ALPHA_SET_FAILED           = 30,
    R_WM_ERR_SYS_WIN_DELETE_ALL_SPRITES_FAILED  = 31,
    R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW        = 32,
    R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE          = 33,
    R_WM_ERR_SYS_CAPTURE_CREATE_FAILED          = 34,
    R_WM_ERR_SYS_CAPTURE_DELETE_FAILED          = 35,
    R_WM_ERR_SYS_CAPTURE_ENABLE_FAILED          = 36,
    R_WM_ERR_CAPTURE_UNIT_COUNT_EXCEEDED        = 37,
    R_WM_ERR_VOUT_INTERNAL                      = 38,
```

```
    R_WM_ERR_SCREEN_TIMING_NOT_SET                          = 39,
    R_WM_ERR_NOT_INITIALIZED                                = 40,
    R_WM_ERR_NOT_UNINITIALIZED                              = 41,
    R_WM_ERR_COLOR_KEYING_NOT_SUPPORTED_FOR_LAYER           = 42,
    R_WM_ERR_ADDRESS_ALIGNMENT                              = 43,
    R_WM_ERR_DISPLAY_OUTPUT_FORMAT_SET                      = 44,
    R_WM_ERR_SYS_WIN_SWAP_FAILED                            = 45,
    R_WM_ERR_SYS_WIN_DELETE_FAILED                          = 46,
    R_WM_ERR_SYS_WIN_EXTERNAL_BUF_SET_FAILED                = 47,
    R_WM_ERR_SYS_WIN_CREATE_FAILED                          = 48,
    R_WM_ERR_SYS_WIN_MOVE_FAILED                            = 49,
    R_WM_ERR_SYS_WIN_GEOMETRY_SET_FAILED                    = 50,
    R_WM_ERR_WIN_SWAP_FAILED                                = 51,
    R_WM_ERR_SPEA_INTERNAL                                  = 52,
    R_WM_ERR_SYS_WIN_FLAG_UPDATE_FAILED                     = 53,
    R_WM_ERR_COULD_NOT_SET_SCREEN_COLOR_CURVE               = 54,
    R_WM_ERR_COULD_NOT_SET_SCREEN_GAMMA                     = 55,
    R_WM_ERR_NOT_SUPPORTED                                  = 56,
    R_WM_ERR_SYS_WIN_SCALED_SET_FAILED                      = 57,
    R_WM_ERR_SYS_CAPTURE_MOVE_FAILED                        = 58,
    R_WM_ERR_SYS_CAPTURE_RESIZE_FAILED                      = 59,
    R_WM_ERR_SYS_CAPTURE_SET_VSYNC_FAILED                   = 60,
    R_WM_ERR_SYS_VOCA_CREATE_FAILED                         = 61,
    R_WM_ERR_SYS_VOCA_DELETE_FAILED                         = 62,
    R_WM_ERR_SYS_VOCA_ENABLE_FAILED                         = 63,
    R_WM_ERR_SYS_VOCA_SET_FAILED                            = 64,
    R_WM_ERR_SYS_ACT_MON_FAILED                             = 65,
    R_WM_ERR_SYS_DISCOM_CREATE_FAILED                       = 66,
    R_WM_ERR_SYS_DISCOM_DELETE_FAILED                       = 67,
    R_WM_ERR_SYS_DISCOM_ENABLE_FAILED                       = 68,
    R_WM_ERR_SYS_DISCOM_SET_FAILED                          = 69,
    R_WM_ERR_SYS_DISCOM_GET_FAILED                          = 70,
    R_WM_ERR_VOCA_NOT_FOUND                                 = 71,
    R_WM_ERR_DISCOM_NOT_FOUND                               = 72,
    R_WM_ERR_VOCA_INTERNAL                                  = 73,
    R_WM_ERR_DISCOM_INTERNAL                                = 74,
    R_WM_ERR_LAST
} r_wm_Error_t
```

Table 5-4 Enumerator of r_wm_Error_t

Name	Description
R_WM_ERR_OK	No error occurred.
R_WM_ERR_NG	An error has occurred, but no specific error code is defined for it.
R_WM_ERR_RANGE_WM	Parameter is the outside the range.
R_WM_ERR_PARAM_INCORRECT	Parameter provided to a function is incorrect.
R_WM_UNIT_LOCKED	Unit is locked.
R_WM_UNIT_NOTLOCKED	Unit is not locked.
R_WM_ERR_NO_PHYS_WINDOW	An error has occurred when trying to access non-existing window.
R_WM_ERR_MALLOC_FAILED	Failed to allocate memory.
R_WM_ERR_FREE_FAILED	Failed to free memory.
R_WM_ERR_EVENT_FAILED	An error occurred when message ID is not valid.
R_WM_ERR_VOUT	An error occurred when the number of WM unit is 0.
R_WM_ERR_NOT_DISABLED	An error has occurred when trying to change display parameters while display is active.

Name	Description
R_WM_ERR_NOT_DELETED	An error has occurred when trying to de-initializes the WM driver without freeing up all the allocated windows.
R_WM_ERR_COLORFMT	Invalid color format.
R_WM_ERR_VIN	Error video Input.
R_WM_ERR_NOT_SPRITE_WINDOW	Window type is not Sprite window.
R_WM_ERR_DISPLAY_TIMING_SET	Invalid display timing set.
R_WM_ERR_INVALID_WM_UNIT	Unit number is outside the range.
R_WM_ERR_DEV_DEINIT_FAILED	Failed to de-initialize WM unit.
R_WM_ERR_NULL_PTR_ARGUMENT	Invalid null pointer.
R_WM_ERR_SPRITE_NOT_FOUND	An error has occurred when trying to delete a Sprite which is not present.
R_WM_ERR_SYS_LAYER_INIT_FAILURE	Failed to initialize the layer.
R_WM_ERR_NOT_CLUT_WIN_FMT	An error has occurred when trying to set an invalid color format.
R_WM_ERR_NO_SUCH_WINDOW	Invalid window.
R_WM_ERR_COULD_NOT_ENABLE_SCREEN	Failed to enable the screen.
R_WM_ERR_COULD_NOT_DISABLE_SCREEN	Failed to disable the screen.
R_WM_ERR_COULD_NOT_SET_SCREEN_BG_COLOR	Failed to set the screen background color.
R_WM_ERR_COULD_NOT_REGISTER_EVENT	Failed to register event.
R_WM_ERR_NOT_FB_WINDOW	Window type is not Frame buffer window.
R_WM_ERR_SYNC_MODE_NOT_POSSIBLE	Invalid synchronize mode.
R_WM_ERR_SYS_WIN_ALPHA_SET_FAILED	Failed to set window alpha.
R_WM_ERR_SYS_WIN_DELETE_ALL_SPRITES_FAILED	Failed to delete all Sprites.
R_WM_ERR_NOT_SUITABLE_CAPTURE_WINDOW	This error code is returned when trying to create a capture window in a Sprite layer.
R_WM_ERR_COULD_NOT_WRITE_MSG_QUEUE	Failed to enqueue message.
R_WM_ERR_SYS_CAPTURE_CREATE_FAILED	Failed to create capture.
R_WM_ERR_SYS_CAPTURE_DELETE_FAILED	Failed to delete capture.
R_WM_ERR_SYS_CAPTURE_ENABLE_FAILED	Failed to enable capture.
R_WM_ERR_CAPTURE_UNIT_COUNT_EXCEEDED	Number of capture units exceeded the maximum allowed.
R_WM_ERR_VOUT_INTERNAL	Error occurs in VDCE driver.
R_WM_ERR_SCREEN_TIMING_NOT_SET	An error has occurred when trying to enable screen before setting the timing parameters.
R_WM_ERR_NOT_INITIALIZED	WM unit is not initialized.
R_WM_ERR_NOT_UNINITIALIZED	WM unit is not uninitialized.
R_WM_ERR_COLOR_KEYING_NOT_SUPPORTED_FOR_LAYER	Color key is not supported for layers.
R_WM_ERR_ADDRESS_ALIGNMENT	An error has occurred when trying to set a frame buffer with an address which is not 32 bytes aligned.
R_WM_ERR_DISPLAY_OUTPUT_FORMAT_SET	Invalid output color format.
R_WM_ERR_SYS_WIN_SWAP_FAILED	Failed to swap in physical window.
R_WM_ERR_SYS_WIN_DELETE_FAILED	Failed to delete in physical window.
R_WM_ERR_SYS_WIN_EXTERNAL_BUF_SET_FAILED	Failed to set external buffer.
R_WM_ERR_SYS_WIN_CREATE_FAILED	Failed to create physical window.
R_WM_ERR_SYS_WIN_MOVE_FAILED	Failed to move physical window.
R_WM_ERR_SYS_WIN_GEOMETRY_SET_FAILED	Failed to set the physical window geometry.
R_WM_ERR_WIN_SWAP_FAILED	Failed in window swapping.
R_WM_ERR_SPEA_INTERNAL	Error occurs in SEPA driver.

Name	Description
R_WM_ERR_SYS_WIN_FLAG_UPDATE_FAILED	An error has occurred when window Flags update fails.
R_WM_ERR_COULD_NOT_SET_SCREEN_COLOR_CURVE	Failed to set screen color curve.
R_WM_ERR_COULD_NOT_SET_SCREEN_GAMMA	Failed to set screen gamma.
R_WM_ERR_NOT_SUPPORTED	The function is not supported with target device.
R_WM_ERR_SYS_WIN_SCALED_SET_FAILED	Failed to change scaling window size.
R_WM_ERR_SYS_CAPTURE_MOVE_FAILED	Failed to change capture position.
R_WM_ERR_SYS_CAPTURE_RESIZE_FAILED	Failed to change capture size.
R_WM_ERR_SYS_CAPTURE_SET_VSYNC_FAILED	Failed to set external Vsync setting.
R_WM_ERR_SYS_VOCA_CREATE_FAILED	Failed to create VOCA monitor area.
R_WM_ERR_SYS_VOCA_DELETE_FAILED	Failed to delete VOCA monitor area.
R_WM_ERR_SYS_VOCA_ENABLE_FAILED	Failed to enable/disable VOCA monitor area.
R_WM_ERR_SYS_VOCA_SET_FAILED	Failed to set VOCA settings.
R_WM_ERR_SYS_ACT_MON_FAILED	Failed to enable/disable activity monitor.
R_WM_ERR_SYS_DISCOM_CREATE_FAILED	Failed to create DISCOM device.
R_WM_ERR_SYS_DISCOM_DELETE_FAILED	Failed to delete DISCOM device.
R_WM_ERR_SYS_DISCOM_ENABLE_FAILED	Failed to enable/disable DISCOM device.
R_WM_ERR_SYS_DISCOM_SET_FAILED	Failed to set DISCOM settings.
R_WM_ERR_SYS_DISCOM_GET_FAILED	Failed to get CRC value from DISCOM device.
R_WM_ERR_VOCA_NOT_FOUND	VOCA monitor area is not created.
R_WM_ERR_DISCOM_NOT_FOUND	DISCOM device is not created.
R_WM_ERR_VOCA_INTERNAL	Error occurs in VOCA driver.
R_WM_ERR_DISCOM_INTERNAL	Error occurs in DISCOM driver.

See also

None

5.3.2 r_wm_WinMode_t

Description

Window mode.

Definition

```
typedef enum
{
    R_WM_WINMODE_FB,
    R_WM_WINMODE_SPRITES
} r_wm_WinMode_t
```

Table 5-5 Enumerator of r_wm_WinMode_t

Name	Description
R_WM_WINMODE_FB	The window is a frame buffer window.
R_WM_WINMODE_SPRITES	The window is a Sprite-hosting window.

See also

r_wm_Window_t

5.3.3 r_wm_WinStatus_t

Description

Window status.

Definition

```
typedef enum
{
    R_WM_WINSTATUS_NOT_INITIALIZED = 0,
    R_WM_WINSTATUS_DISABLED,
    R_WM_WINSTATUS_ENABLED
} r_wm_WinStatus_t
```

Table 5-6 Enumerator of r_wm_WinStatus_t

Name	Description
R_WM_WINSTATUS_NOTINITIALIZED	The window has not been created.
R_WM_WINSTATUS_DISABLED	The window is invisible on the screen.
R_WM_WINSTATUS_ENABLED	The window is visible on the screen.

See also

r_wm_Window_t

5.3.4 r_wm_WinBufStatus_t

Description

Frame buffer status.

Definition

```
typedef enum
{
    R_WM_WINBUF_FREE = 0,
    R_WM_WINBUF_RENDER_STARTED,
    R_WM_WINBUF_RENDER_FINISHED,
    R_WM_WINBUF_ON_SCREEN
} r_wm_WinBufStatus_t
```

Table 5-7 Enumerator of r_wm_WinBufStatus_t

Name	Description
R_WM_WINBUF_FREE	The buffer can be used for rendering operations and it is not visible on the screen.
R_WM_WINBUF_RENDER_STARTED	Drawing operations have been started on the buffer.
R_WM_WINBUF_RENDER_FINISHED	The drawing operations in the buffer have been completed and the buffer is ready to be displayed.
R_WM_WINBUF_ON_SCREEN	The buffer is scheduled to be transferred to the screen, or it is already on the screen.

See also

r_wm_Window_t
r_wm_WinBuffer_t

5.3.5 r_wm_WinBufAllocMode_t

Description

Frame buffer allocation mode.

Definition

```
typedef enum
{
    R_WM_WINBUF_ALLOC_EXTERNAL,
    R_WM_WINBUF_ALLOC_INTERNAL
} r_wm_WinBufAllocMode_t
```

Table 5-8 Enumerator of r_wm_WinBufAllocMode_t

Name	Description
R_WM_WINBUF_ALLOC_EXTERNAL	External mode. User allocates the frame buffers. WM doesn't allocate the buffer.
R_WM_WINBUF_ALLOC_INTERNAL	Internal mode. R_WM_WindowCreate allocates the buffers from heap memory.

See also

R_WM_WindowCreate
R_WM_WindowExternalBufSet
r_wm_Window_t

5.3.6 r_wm_WinColorFmt_t

Description

Color format of a window.

Definition

```
typedef enum
{
    R_WM_COLORFMT_RGB565,
    R_WM_COLORFMT_ARGB1555,
    R_WM_COLORFMT_ARGB4444,
    R_WM_COLORFMT_RGB0444,
    R_WM_COLORFMT_RGB0888,
    R_WM_COLORFMT_ARGB8888,
    R_WM_COLORFMT_RGBA5551,
    R_WM_COLORFMT_RGBA4444,
    R_WM_COLORFMT_RGBA8888,
    R_WM_COLORFMT_CLUT8,
    R_WM_COLORFMT_CLUT4,
    R_WM_COLORFMT_CLUT1,
    R_WM_COLORFMT_RLE24ARGB8888,
    R_WM_COLORFMT_RLE18ARGB8888,
    R_WM_COLORFMT_RLE24RGB0888,
    R_WM_COLORFMT_RLE18RGB0888,
    R_WM_COLORFMT_RLE8CLUT8,
    R_WM_COLORFMT_RLE8CLUT4,
    R_WM_COLORFMT_RLE8CLUT1,
    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,
    R_WM_COLORFMT_LAST
} r_wm_WinColorFmt_t
```

Table 5-9 Enumerator of r_wm_WinColorFmt_t

Name	Description
R_WM_COLORFMT_RGB565	RGB565
R_WM_COLORFMT_ARGB1555	ARGB1555
R_WM_COLORFMT_ARGB4444	ARGB4444
R_WM_COLORFMT_RGB0444	This format is not supported.
R_WM_COLORFMT_RGB0888	RGB0888
R_WM_COLORFMT_ARGB8888	ARGB8888
R_WM_COLORFMT_RGBA5551	RGBA5551
R_WM_COLORFMT_RGBA4444	This format is not supported.
R_WM_COLORFMT_RGBA8888	RGBA8888
R_WM_COLORFMT_CLUT8	8-bit (256 colors) color-lookup index.
R_WM_COLORFMT_CLUT4	4-bit (16 colors) color-lookup index.
R_WM_COLORFMT_CLUT1	1-bit (2 colors) color lookup index.
R_WM_COLORFMT_RLE24ARGB8888	24bit RLE format which is expanded to ARGB8888
R_WM_COLORFMT_RLE18ARGB8888	This format is not supported.
R_WM_COLORFMT_RLE24RGB0888	24bit RLE format which is expanded to RGB0888
R_WM_COLORFMT_RLE18RGB0888	This format is not supported.
R_WM_COLORFMT_RLE8CLUT8	This format is not supported.
R_WM_COLORFMT_RLE8CLUT4	This format is not supported.
R_WM_COLORFMT_RLE8CLUT1	This format is not supported.
R_WM_COLORFMT_YCBCR_422	YCbCr422.
R_WM_COLORFMT_YCBCR_444	YCbCr444.
R_WM_COLORFMT_YUV_YUYV	YUV422 (Y->U->Y->V)
R_WM_COLORFMT_YUV_UYVY	YUV422 (Y->U->Y->V)
R_WM_COLORFMT_YUV_YVYU	YUV422 (Y->U->Y->V)
R_WM_COLORFMT_YUV_VYUY	YUV422 (Y->U->Y->V)
R_WM_COLORFMT_LAST	Delimiter of the enumerator. This format is not supported.

See also

r_wm_Window_t

5.3.7 r_wm_OutColorFmt_t

Description

The color mode of the video output and option flags.

Definition

```
typedef enum
{
    R_WM_OUTCOLORFMT_RGB888       = 0u,
    R_WM_OUTCOLORFMT_RGB666       = 1u,
    R_WM_OUTCOLORFMT_RGB565       = 2u,
    R_WM_OUTCOLORFMT_LAST         = 3u,
    R_WM_OUTCOLORFMT_FLAG_DITHER  = (int32_t)(1u<<28u),
    R_WM_OUTCOLORFMT_FLAG_SWAP_BR = (int32_t)(1u<<29u),
    R_WM_OUTCOLORFMT_FLAG_ENDIAN  = (int32_t)(1u<<30u),
    R_WM_OUTCOLORFMT_FLAG_MASK    = (R_WM_OUTCOLORFMT_FLAG_ENDIAN
                                   + R_WM_OUTCOLORFMT_FLAG_SWAP_BR
                                   + R_WM_OUTCOLORFMT_FLAG_DITHER)
} r_wm_OutColorFmt_t
```

Table 5-10 Enumerator of r_wm_OutColorFmt_t

Name	Description
R_WM_OUTCOLORFMT_RGB888	RGB888 format.
R_WM_OUTCOLORFMT_RGB666	RGB666 format.
R_WM_OUTCOLORFMT_RGB565	RGB565 format.
R_WM_OUTCOLORFMT_LAST	Delimiter of the enumerator. This format is not supported.
R_WM_OUTCOLORFMT_FLAG_DITHER	If flag is off, truncate dithering is applied. If flag is on, random pattern dithering is applied.
R_WM_OUTCOLORFMT_FLAG_SWAP_BR	Swap blue and red channel of output color.
R_WM_OUTCOLORFMT_FLAG_ENDIAN	Change endianness of output color.
R_WM_OUTCOLORFMT_FLAG_MASK	Mask of option flags.

See also

R_WM_ScreenColorFormatSet

5.3.8 r_wm_SpriteStatus_t

Description

Status of the Sprite data.

Definition

```
typedef enum
{
    R_WM_SPRITESTATUS_NOT_INITIALIZED = 0,
    R_WM_SPRITESTATUS_DISABLED,
    R_WM_SPRITESTATUS_ENABLED
} r_wm_SpriteStatus_t
```

Table 5-11 Enumerator of r_wm_SpritStatus_t

Name	Description
R_WM_SPRITESTATUS_NOT_INITIALIZED	Sprite status is not initialized.
R_WM_SPRITESTATUS_DISABLED	Sprite status is disabled.
R_WM_SPRITESTATUS_ENABLED	Sprite status is enabled.

See also

r_wm_Sprite_t

5.3.9 r_wm_WinFlags_t

Description

On/Off Switches for different functionalities of a WM window.

Definition

```
typedef enum
{
    R_WM_WINFLAG_NONE       = 0,
    R_WM_WINFLAG_V_MIRROR   = 1,
    R_WM_WINFLAG_MASK       = (R_WM_WINFLAG_V_MIRROR)
} r_wm_WinFlags_t
```

Table 5-12 Enumerator of r_wm_WinFlags_t

Name	Description
R_WM_WINFLAG_NONE	No flags are set.
R_WM_WINFLAG_V_MIRROR	Flip the output image vertically.
R_WM_WINFLAG_MASK	Mask of all flags.

See also

r_wm_Window_t

5.3.10 r_wm_EventId_t

Description

Interrupt event ID.
The events are dispatched to the callback function specified during the R_WM_DevInit call.
These events can be used for implementing the asynchronous drawing mechanism.

Definition

```
typedef enum
{
    R_WM_EVENT_VBLANK                   = 0x0,
    R_WM_EVENT_SCANLINE                 = 0x1,
    R_WM_EVENT_VI_VBLANK                = 0x2,
    R_WM_EVENT_VI_OVERFLOW              = 0x3,
    R_WM_EVENT_LAYER0_UNDERFLOW         = 0x4,
    R_WM_EVENT_LAYER1_UNDERFLOW         = 0x5,
    R_WM_EVENT_LAYER2_UNDERFLOW         = 0x6,
    R_WM_EVENT_LAYER3_UNDERFLOW         = 0x7,
    R_WM_EVENT_LAYER1_VBLANK            = 0x8,
    R_WM_EVENT_OIR_VBLANK               = 0x9,
    R_WM_EVENT_OIR_SCANLINE             = 0xA,
    R_WM_EVENT_DISCOM_MISMACTH          = 0xB,
    R_WM_EVENT_VOCA_MISMACTH            = 0xC,
    R_WM_EVENT_ACT_MON_ERROR            = 0xD,
    R_WM_EVENT_LAST
} r_wm_EventId_t
```

Table 5-13 Enumerator of r_wm_EventId_t

Name	Description
R_WM_EVENT_VBLANK	V-sync at layer 0 interrupt notification.
R_WM_EVENT_SCANLINE	Scanline interrupt at layer 3 notification.
R_WM_EVENT_VI_VBLANK	Video Input V-sync interrupt notification.
R_WM_EVENT_VI_OVERFLOW	Video Input overflow interrupt notification.
R_WM_EVENT_LAYER0_UNDERFLOW	Video output layer 0 underflow.
R_WM_EVENT_LAYER1_UNDERFLOW	Video output layer 1 underflow.
R_WM_EVENT_LAYER2_UNDERFLOW	Video output layer 2 underflow.
R_WM_EVENT_LAYER3_UNDERFLOW	Video output layer 3 underflow.
R_WM_EVENT_LAYER1_VBLANK	V-sync at layer 1 interrupt notification.
R_WM_EVENT_OIR_VBLANK	V-sync at OIR layer interrupt notification. Only WM Unit 0 is supported.
R_WM_EVENT_OIR_SCANLINE	Scanline interrupt at OIR layer notification. Only WM Unit 0 is supported.
R_WM_EVENT_DISCOM_MISMACTH	Mismatch occurs in DISCOM device.
R_WM_EVENT_VOCA_MISMACTH	Mismatch occurs in VOCA monitor area.
R_WM_EVENT_ACT_MON_ERROR	Error detected by activity monitor.

See also

R_WM_DevInit
R_WM_DevEventRegister

5.3.11 r_wm_CapMode_t

Description

Input video signal format and optional flags.
Input video signal format must be selected by one of these six flags, or the function call will fail.

- R_WM_CAPMODE_YUV_ITU656
- R_WM_CAPMODE_YUV_8BIT
- R_WM_CAPMODE_YUV_16BIT
- R_WM_CAPMODE_RGB_16BPP
- R_WM_CAPMODE_RGB_18BPP
- R_WM_CAPMODE_RGB_24BPP

Other flags are optional flags.

Definition

```
typedef enum
{
    R_WM_CAPMODE_NONE             = 0,
    R_WM_CAPMODE_YUV_ITU656       = (int32_t)(1uL<< 0),
    R_WM_CAPMODE_YUV_8BIT         = (int32_t)(1uL<< 1),
    R_WM_CAPMODE_YUV_16BIT        = (int32_t)(1uL<< 2),
    R_WM_CAPMODE_RGB_16BPP        = (int32_t)(1uL<< 3),
    R_WM_CAPMODE_RGB_18BPP        = (int32_t)(1uL<< 4),
    R_WM_CAPMODE_RGB_24BPP        = (int32_t)(1uL<< 5),
    R_WM_CAPMODE_DITHER           = (int32_t)(1uL<< 6),
    R_WM_CAPMODE_YUV_Y_UV_INVERT  = (int32_t)(1uL<< 10),
    R_WM_CAPMODE_VSYNC_INVERT     = (int32_t)(1uL<< 11),
    R_WM_CAPMODE_HSYNC_INVERT     = (int32_t)(1uL<< 12),
    R_WM_CAPMODE_DATA_CLK_INVERT  = (int32_t)(1uL<< 15),
    R_WM_CAPMODE_VSYNC_CLK_INVERT = (int32_t)(1uL<< 16),
    R_WM_CAPMODE_HSYNC_CLK_INVERT = (int32_t)(1uL<< 17),
    R_WM_CAPMODE_H_MIRRORING      = (int32_t)(1uL<< 18),
    R_WM_CAPMODE_V_MIRRORING      = (int32_t)(1uL<< 19),
    R_WM_CAPMODE_FIXED_VSYNC      = (int32_t)(1uL<< 20),
    R_WM_CAPMODE_BIG_ENDIAN       = (int32_t)(1uL<< 21),
    R_WM_CAPMODE_DE_MODE          = (int32_t)(1uL<< 22),
    R_WM_CAPMODE_PAL              = (int32_t)(1uL<< 23),
    R_WM_CAPMODE_EAV              = (int32_t)(1uL<< 24),
    R_WM_CAPMODE_SYNC_ONLY        = (int32_t)(1uL<< 25),
    R_WM_CAPMODE_BOB_TOP          = (int32_t)(1uL<< 28),
    R_WM_CAPMODE_BOB_BOTTOM       = (int32_t)(1uL<< 29),
    R_WM_CAPMODE_WEAVE            = (int32_t)(1ul<< 30)
} r_wm_CapMode_t
```

Table 5-14 Enumerator of r_wm_CapMode_t

Name	Description
R_WM_CAPMODE_NONE	Unsupported.
R_WM_CAPMODE_YUV_ITU656	Select ITU-R BT.656 mode.
R_WM_CAPMODE_YUV_8BIT	Select ITU-R BT.601 mode.
R_WM_CAPMODE_YUV_16BIT	Select YCbCr422 mode.
R_WM_CAPMODE_RGB_16BPP	Select RGB565 mode.
R_WM_CAPMODE_RGB_18BPP	Select RGB666 mode.
R_WM_CAPMODE_RGB_24BPP	Select RGB888 mode.
R_WM_CAPMODE_DITHER	If this flag is off, dithering is disabled.

	If this flag is on, dithering is enabled. Dithering is effective when Window color format is R_WM_COLORFMT_RGB565.
R_WM_CAPMODE_YUV_Y_UV_INVERT	If this flag is off, capturing order is Y->Cb->Y->Cr If this flag is on, capturing order is Cb->Y->Cr->Y This flag is effective in case of BT.656 or BT.601 input. See INP_H_POS description in H/W User's manual.
R_WM_CAPMODE_VSYNC_INVERT	If this flag is off, V-sync is positive polarity. If this flag is on, V-sync is negative polarity.
R_WM_CAPMODE_HSYNC_INVERT	If this flag is off, H-sync is positive polarity. If this flag is on, H-sync is negative polarity.
R_WM_CAPMODE_DATA_CLK_INVERT	If this flag is off, data capturing timing is falling edge. If this flag is on, data capturing timing is rising edge.
R_WM_CAPMODE_VSYNC_CLK_INVERT	If this flag is off, V-sync signal capturing timing is falling edge. If this flag is on, V-sync signal capturing timing is rising edge.
R_WM_CAPMODE_HSYNC_CLK_INVERT	If this flag is off, H-sync signal capturing timing is falling edge. If this flag is on, H-sync signal capturing timing is rising edge.
R_WM_CAPMODE_H_MIRRORING	If this flag is off, horizontal mirroring is disabled. If this flag is on, horizontal mirroring is enabled.
R_WM_CAPMODE_V_MIRRORING	If this flag is off, vertical mirroring is disabled. If this flag is on, vertical mirroring is enabled.
R_WM_CAPMODE_FIXED_VSYNC	If this flag is off, synchronous V-sync of connecting layer is external input V-sync. If this flag is on, synchronous V-sync of connecting layer is fixed as internally generated free-running V-sync signal.
R_WM_CAPMODE_BIG_ENDIAN	If this flag is off, capturing with little endian. If this flag is on, capturing with big endian.
R_WM_CAPMODE_DE_MODE	If this flag is off, DE mode is disabled. Hsync signal is used to capture. If this flag is on, DE mode is enabled. DE signal is used instead of Hsync signal. DE signal should be input to VDCE0_VI_HSYNC/VDCE1_VI_HSYNC port. R_WM_CAPMODE_HSYNC_INVERT or R_WM_CAPMODE_HSYNC_CLK_INVERT flag is valid for DE signal. DE mode can be selected in case of YCbCr422/RGB565/RGB666/ RGB888 input mode.
R_WM_CAPMODE_PAL	If this flag is off, input signal is assumed as NTSC (525 Lines/59.94 Hz) system. If this flag is on, input signal is assumed as PAL (625 Lines/50.00 Hz) system. This flag is valid in case of BT.656 or BT.601 input mode.
R_WM_CAPMODE_EAV	If this flag is off, SAV code is converted to Hsync signal. If this flag is on, EAV code is converted to Hsync signal. This flag is valid in case of BT.656 input mode.
R_WM_CAPMODE_SYNC_ONLY	If this flag is off, input data is written to capture buffer. If this flag is on, input data is not written to capture buffer. Only clock, Vsync Hsync are input. This mode is used when video data is not captured but the video is output in synchronization with the external Vsync input.
R_WM_CAPMODE_BOB_TOP	If this flag is off, bob de-interlacing mode is disabled. If this flag is on, bob de-interlacing mode by top field is enabled. Only top field is written to capture buffer in case of interlace video input. If this flag is on, once every two frames is written to capture buffer in case of progressive video input.
R_WM_CAPMODE_BOB_BOTTOM	If this flag is off, bob de-interlacing mode is disabled. If this flag is on, bob de-interlacing mode by top field is enabled. Only bottom field is written to capture buffer. This flag is effective in case of interlace video input.
R_WM_CAPMODE_WEAVE	If this flag is off, weave de-interlacing mode is disabled. If this flag is on, weave de-interlacing mode is enabled. Output by interweaving top field and bottom field alternately. The weave mode is effective in case of interlace video input. Only single frame buffer is supported, no multi-buffering.

See also

r_wm_Capture_t

5.3.12 r_wm_WinCapbs_t

Description

Selectable window type.

Definition

```
typedef enum
{
    R_WM_WINCAPBS_RLE      = 0,
    R_WM_WINCAPBS_SPRITES  = 1,
    R_WM_WINCAPBS_LAST     = 2
} r_wm_WinCapbs_t
```

Table 5-15 Enumerator of r_wm_WinCapbs_t

Name	Description
R_WM_WINCAPBS_RLE	RLE window is selectable.
R_WM_WINCAPBS_SPRITES	Sprite window is selectable.
R_WM_WINCAPBS_LAST	Delimiter of the enumerator. Not supported.

See also

R_WM_WindowCapabilitiesSet

5.4 Structure Type

This section shows the structure used in WM API Function.

5.4.1 r_wm_Msg_t

Description

Message queue information. This structure is used in WM driver internally.
User does not need to be aware of the content of the message.
User should prepare the buffer for message queue in R_WM_DevInit.
The structure size (one message queue size) is 20 Bytes.

Definition

```
typedef struct r_wm_Msg_s r_wm_Msg_t;
```

See also

R_WM_DevInit

5.4.2 r_wm_WinBuffer_t

Description

Frame buffer information.

Definition

```
typedef struct
{
    void*                   Data;
    r_wm_WinBufStatus_t     Status;
} r_wm_WinBuffer_t
```

Table 5-16 Member of r_wm_WinBuffer_t

Name	Description
Data	The start address of the frame buffer. It should be 128 Byte alignment.
Status	The status of the frame buffer. R_WM_WINBUF_FREE R_WM_WINBUF_RENDER_STARTED R_WM_WINBUF_RENDER_FINISHED R_WM_WINBUF_ON_SCREEN

See also

r_wm_WinBufStatus_t
r_wm_Window_t

5.4.3 r_wm_ClutEntry_t

Description

Defines an entry of the color lookup table.
It is also used as gamma curve parameter.

Definition

```
typedef struct
{
    uint8_t B;
    uint8_t G;
    uint8_t R;
    uint8_t A;
} r_wm_ClutEntry_t
```

Table 5-17 Member of r_wm_ClutEntry_t structure

Name	Description
B	8-bit blue component.
G	8-bit green component.
R	8-bit red component.
A	8-bit alpha component.

See also

r_wm_Window_t
R_WM_ScreenColorCurveSet
R_WM_WindowClutSet

5.4.4 r_wm_Sprite_t

Description

Each Sprite data information.
See *Table 3-29* about the range and alignment information.

Definition

```
typedef struct r_wm_Sprite_s
{
    struct r_wm_Window_s  *Window;
    struct r_wm_Sprite_s  *Next;
    r_wm_SpriteStatus_t   Status;
    void                  *Data;
    uint32_t              PosX;
    uint32_t              PosY;
    uint32_t              PosZ;
    uint32_t              Width;
    uint32_t              Height;
} r_wm_Sprite_t
```

Table 5-18 Member of r_wm_Sprite _t structure

Name	Description
Window	Pointer to Window structure which has been created already.
Next	Pointer to the next Sprite in the queue. This is used in WM internally. User should initialize as R_NULL when R_WM_SpriteCreate is called.
Status	Status of the Sprite. R_WM_SPRITESTATUS_NOT_INITIALIZED R_WM_SPRITESTATUS_DISABLED R_WM_SPRITESTATUS_ENABLED This is used in WM internally. User should initialize as R_WM_SPRITESTATUS_NOT_INITIALIZED when R_WM_SpriteCreate is called.
Data	Start address of Sprite source data. It should be 8 Bytes aligned.
PosX	X position of Sprite data. Unit is pixel. It should be 8 Bytes aligned.
PosY	Y position of Sprite data. Unit is pixel.
PosZ	Z position of Sprite data. Z position is the relative position in the Sprite window.
Width	Width of Sprite data. It should be 8 Bytes aligned.
Height	Height of Sprite data. Unit is pixel.

See also

r_wm_Window_t

5.4.5 r_wm_Window_t

Description

Each window has a data structure of this type. All windows are linked in a chain by the window manager. See *Table 3-10*, *Table 3-17*, *Table 3-26*, *Table 3-27* about the range and alignment information.

Definition

```
typedef struct r_wm_Window_s r_wm_Window_t;

struct r_wm_Window_s
{
    r_wm_WinStatus_t    Status;
    r_wm_WinMode_t      Mode;
    r_wm_WinColorFmt_t  ColorFmt;
    int32_t             PosX;
    int32_t             PosY;
    uint32_t            PosZ;
    uint32_t            Pitch;
    uint32_t            Width;
    uint32_t            Height;
    uint32_t            ScaledWidth;
    uint32_t            ScaledHeight;
    union
    {
        struct
        {
            r_wm_WinBuffer_t        *Buffer;
            uint8_t                 BufNum;
            r_wm_WinBufAllocMode_t  BufMode;
        } Fb;
        r_wm_Sprite_t          *SpritesRoot;
    } Surface;
    uint8_t                         Alpha;
    uint8_t                         UsePremultipliedAlpha;
    uint32_t                        ClutNumEntries;
    const r_wm_ClutEntry_t*         Clut;
    struct
    {
        uint8_t Enabled;
        union
        {
            struct
            {
                uint8_t             Red;
                uint8_t             Green;
                uint8_t             Blue;
            } RgbKey;
            uint8_t ClutKey;
        } In;
        struct
        {
            uint8_t                 Red;
            uint8_t                 Green;
            uint8_t                 Blue;
            uint8_t                 Alpha;
        } Out;
```

```
    } ColorKey;
    r_wm_WinFlags_t         Flags;
    struct r_wm_Window_s   *Next;
} /*r_wm_Window_t*/
```

Table 5-19 Member of r_wm_Window_t structure

Name	Description
Status	Status of the Window. R_WM_WINSTATUS_NOT_INITIALIZED R_WM_WINSTATUS_DISABLED R_WM_WINSTATUS_ENABLED This is used in WM internally. User should initialize as R_WM_WINSTATUS_NOT_INITIALIZED when R_WM_WindowCreate is called.
Mode	Window mode R_WM_WINMODE_FB R_WM_WINMODE_SPRITES
ColorFmt	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
PosX	X position of the window. Unit is pixels. Range is -1280 to 1279.
PosY	Y position of the window. Unit is pixels. Range is -1024 to 1023.
PosZ	Z position of the window.
Pitch	Line pitch of a frame buffer. Unit is pixels. It should be 128 bytes aligned.
Width	Width of the window. Unit is pixels. Range is 3 to 1280.
Height	Height of the window. Unit is pixels. Range is 1 to 1024.
ScaledWidth	Original width when scaling-up. Unit is pixels. When scaling-up is disabled, set to 0. When scaling-up is enabled, range is 4 to (Width-1).
ScaledHeight	Original height when scaling-up. Unit is pixels. When scaling-up is disabled, set to 0. When scaling-up is enabled, range is 4 to (Height-1).
Surface.Fb.Buffer	Pointer to an array of r_wm_WinBuffer_t elements. This is used when window mode is R_WM_WINMODE_FB.
Surface.Fb.BufNum	This is the number of frame buffers of the window. This is used when window mode is R_WM_WINMODE_FB.
Surface.Fb.BufMode	Frame buffer allocation mode. R_WM_WINBUF_ALLOC_EXTERNAL R_WM_WINBUF_ALLOC_INTERNAL This is used when window mode is R_WM_WINMODE_FB.
Surface.SpritesRoot	Root of Sprite data assigned to the window. This is used in WM internally.

Name	Description
	User should initialize as R_NULL when R_WM_WindowCreate is called. This is used when window mode is R_WM_WINMODE_SPRITES.
Alpha	The constant alpha value of the window. Range is 0 to 255.
UsePremultipliedAlpha	Pre-multiplied Alpha Mode. 0: Disable 1: Enable
ClutNumEntries	The number of entries of the CLUT assigned to this window. This is used only when ColorFmt is CLUT type.
Clut	Pointer to array of CLUT. It should be 4 Byte aligned. This is used only when ColorFmt is CLUT type.
ColorKey.Enabled	Chromakey Mode. 0: Disable 1: Enable Following ColorKey member is valid when Chromakey Mode is enabled.
ColorKey.In.RgbKey.Red	Red component of the target color to replace.
ColorKey.In.RgbKey.Green	Green component of the target color to replace.
ColorKey.In.RgbKey.Blue	Blue component of the target color to replace.
ColorKey.In.ClutKey	This is not supported.
ColorKey.Out.Red	Red component of the replacing color.
ColorKey.Out.Green	Green component of the replacing color.
ColorKey.Out.Blue	Blue component of the replacing color.
ColorKey.Out.Alpha	Alpha component of the replacing color.
Flags	Additional flags to set certain properties of a window. R_WM_WINFLAG_NONE R_WM_WINFLAG_V_MIRROR
Next	Internal pointer to the next window in the chain. This is used in WM internally. User should initialize as R_NULL when R_WM_WindowCreate is called.

See also

r_wm_WinStatus_t
r_wm_WinMode_t
r_wm_WinColorFmt_t
r_wm_WinBuffer_t
r_wm_WinBufAllocMode_t
r_wm_Sprite_t
r_wm_ClutEntry_t
r_wm_WinFlags_t
R_WM_WindowCreate

5.4.6 r_wm_Event_t

Description

The external event information.

Definition

```
typedef struct
{
    r_wm_EventId_t  Id;
    uint32_t        Data;
} r_wm_Event_t
```

Table 5-20 Member of r_wm_Event_t structure

Name	Description
Id	Event ID. 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
Data	The value depending on Event ID. See following table.

Table 5-21 Event Id and Data

Id	Data
R_WM_EVENT_DISCOM_MISMACTH	DISCOM unit number where mismatch occurs. 0x01: Error occurs at DISCOM unit 0 0x02: Error occurs at DISCOM unit 1 0x04: Error occurs at DISCOM unit 2 0x08: Error occurs at DISCOM unit 3 When multiple errors are acquired by one interrupt, this value is the logical sum.
R_WM_EVENT_VOCA_MISMACTH	VOCA monitor area index number where mismatch occurs. 0x0001: Error occurs at VOCA monitor area 0 0x0002: Error occurs at VOCA monitor area 1 : 0x4000: Error occurs at VOCA monitor area 14 0x8000: Error occurs at VOCA monitor area 15 When multiple errors are acquired by one interrupt, this value is the logical sum.
Others	It has no means.

See also

r_wm_EventId_t
R_WM_DevInit

5.4.7 r_wm_Capture_t

Description

The type describes the settings of a video capturing surface.

Definition

```
typedef struct r_wm_Capture_s r_wm_Capture_t;

struct r_wm_Capture_s
{
    r_wm_Window_t           *Window;
    r_wm_CapMode_t          Mode;
    uint32_t                StartX;
    uint32_t                StrideX;
    uint32_t                StartY1;
    uint32_t                StartY2;
    uint32_t                Width;
    uint32_t                Height;
    uint32_t                ScaledWidth;
    uint32_t                ScaledHeight;
    uint32_t                Delay;
    uint32_t                CapUnit;
    struct r_wm_Capture_s   *Next;
}
```

Table 5-22 Member of r_wm_Capture_t

Name	Description
Window	Pointer to Window structure which has been created already.
Mode	Video input type and capturing mode. R_WM_CAPMODE_YUV_ITU656 R_WM_CAPMODE_YUV_8BIT R_WM_CAPMODE_YUV_16BIT R_WM_CAPMODE_RGB_16BPP R_WM_CAPMODE_RGB_18BPP R_WM_CAPMODE_RGB_24BPP R_WM_CAPMODE_DITHER R_WM_CAPMODE_YUV_Y_UV_INVERT R_WM_CAPMODE_VSYNC_INVERT R_WM_CAPMODE_HSYNC_INVERT R_WM_CAPMODE_DATA_CLK_INVERT R_WM_CAPMODE_VSYNC_CLK_INVERT R_WM_CAPMODE_HSYNC_CLK_INVERT R_WM_CAPMODE_H_MIRRORING R_WM_CAPMODE_V_MIRRORING R_WM_CAPMODE_FIXED_VSYNC R_WM_CAPMODE_BIG_ENDIAN R_WM_CAPMODE_DE_MODE R_WM_CAPMODE_PAL R_WM_CAPMODE_EAV R_WM_CAPMODE_SYNC_ONLY R_WM_CAPMODE_BOB_TOP R_WM_CAPMODE_BOB_BOTTOM R_WM_CAPMODE_WEAVE
StartX	X position of capturing start. Unit is pixels. When DE mode is disabled, reference point is Hsync signal. StartX pixels are skipped after Hsync signal. The range is 16 to 2011. If setting value is 0 to 15, this function rounds up to 16.

	When DE mode is enabled (see R_WM_CAPMODE_DE_MODE), the reference point is DE signal. StartX pixels are skipped after DE signal. The range of 0 to 2011.
StrideX	The stride of the frame buffer. Unit is pixels. It should be 128 bytes aligned.
StartY1	Y position of capturing start. Unit is pixels. When DE mode is disabled, reference point is Vsync signal. (StartY + 1) lines are skipped after Vsync signal. The range is 4 to 2035. If setting value is 0 to 3, this function rounds up to 4. When DE mode is enabled (see R_WM_CAPMODE_DE_MODE), the reference point is DE signal after Vsync signal. StartY pixels are skipped after 1st DE signal. The range of 0 to 2035.
StartY2	This value is not used.
Width	Width of capturing video data. Unit is pixels. Range is 4 to 2011. It should be 4 pixels aligned. When Width is greater than 1024 pixels, horizontal image size must be scaled-down to 1024 pixels or less.
Height	Height of capturing video data. Unit is pixels. Range is 4 to 1024. It should be 4 pixels aligned.
ScaledWidth	Frame buffer width when scaling-down. Unit is pixels. When scaling-down is disabled, set to 0. When scaling-down is enabled, range is 4 to (Width-1). It should be 4 pixels aligned.
ScaledHeight	Frame buffer height when scaling-down. Unit is pixels. When scaling-down is disabled, set to 0. When scaling-down is enabled, range is 4 to (Height-1). It should be 4 pixels aligned.
Delay	Delay from video input V-sync to Layer output V-sync.
CapUnit	The WM unit number to be used for capturing. If another Unit is to be used than the assigned Window, make sure that this unit is already initialized.
Next	Internal pointer to the next capture in the chain. This is used in WM internally. User should initialize as R_NULL when R_WM_CaptureCreate is called.

See also

r_wm_Window_t
r_wm_CapMode_t

5.4.8 r_wm_VocaClutEntry_t

Description

Defines a VOCA CLUT entry. Each CLUT has valid level (range) for a pixel.
Set each component as RGB888 format. Each range is 0 to 255.

Definition

```
typedef struct
{
    uint8_t RUpper;
    uint8_t GUpper;
    uint8_t BUpper;
    uint8_t RLower;
    uint8_t GLower;
    uint8_t BLower;
} r_wm_VocaClutEntry_t
```

Table 5-23 Member of r_wm_VocaClutEntry_t structure

Name	Description
RUpper	Upper value of red component.
GUpper	Upper value of green component.
BUpper	Upper value of blue component.
RLower	Lower value of red component.
GLower	Lower value of green component.
BLower	Lower value of blue component.

See also

r_wm_Voca_t
R_WM_ScreenVocaClutSet

5.4.9 r_wm_Voca_t

Description

Each VOCA monitor area has a data structure of this type.

Definition

```
typedef struct r_wm_Voca_s r_wm_Voca_t;

struct r_wm_Voca_s
{
    uint32_t                Threshold;
    uint16_t                AreaNum;
    uint16_t                PosX;
    uint16_t                PosY;
    uint16_t                Width;
    uint16_t                Height;
    uint16_t                RamAddr;
    uint16_t                ExpSize;
    const uint16_t          *ExpImg;
    r_wm_VocaClutEntry_t    Clut[R_WM_VOCA_CLUT_NUM];
    struct r_wm_Voca_s      *Next;
}
```

Table 5-24 Member of r_wm_Voca_t

Name	Description
Threshold	Acceptable mismatch (difference) of a Monitor Area. The range is 1 to 0x3FFFF.
AreaNum	Monitor area index number. The range is 0 to 15.
PosX	X position of Monitor area. Unit is pixel.
PosY	Y position of Monitor area. Unit is pixel.
Width	Width of Monitor area. Unit is pixel. The range is 1 to 128.
Height	Height of Monitor area. Unit is pixel. The range is 1 to 128.
RamAddr	Internal RAM start index of a monitor area. The range is 0 to 4095.
ExpSize	The number of expected image array. The range is 0 to 2048.
ExpImg	Pointer to update expected image array. User should prepare the number of arrays specified by 'ExpSize'. If ‘ExpSize’ is 0, set R_NULL.
Clut	CLUT value of a Monitor Area. Each CLUT has valid level (range) for a pixel.
Next	Internal pointer to the next capture in the chain. This is used in WM internally. User should initialize as R_NULL when R_WM_ScreenVocaCreate is called.

See also

r_wm_VocaClutEntry_t
R_WM_ScreenVocaCreate

5.4.10 r_wm_Discom_t

Description

Each Discom device has a data structure of this type.

Definition

```
typedef struct r_wm_Discom_s r_wm_Discom_t;

struct r_wm_Discom_s
{
    uint32_t                ExpCrc;
    uint16_t                PosX;
    uint16_t                PosY;
    uint16_t                Width;
    uint16_t                Height;
    uint8_t                 DiscomUnit;
    struct r_wm_Discom_s    *Next;
}
```

Table 5-25 Member of r_wm_Discom_t

Name	Description
ExpCrc	Expected CRC data.
PosX	X position of Monitor area. Unit is pixel.
PosY	Y position of Monitor area. Unit is pixel.
Width	Width of Monitor area. Unit is pixel.
Height	Height of Monitor area. Unit is pixel.
DiscomUnit	DISCOM unit number. The range is 0 - 3. 0 - 1: DISCOM unit connected to WM unit 0 2 - 3: DISCOM unit connected to WM unit 1
Next	Internal pointer to the next capture in the chain. This is used in WM internally. User should initialize as R_NULL when R_WM_DiscomCreate is called.

See also

R_WM_DiscomCreate

6.Appendix

6.1 Alignment table

Table 6-1 Alignment table

Action	Alignment	Constraint of	Directly affected APIs
Window frame buffer start address (i.e. H/W layer start address)	128 Bytes	VDCE	R_WM_WindowCreate R_WM_WindowExternalBufSet
Window frame buffer stride (a.k.a. pitch)	128 Bytes (+128 Bytes RLE)	VDCE, SPEA	R_WM_WindowCreate
Sprite buffer address	8 Bytes	SEPA	R_WM_SpriteCreate, R_WM_SpriteBufSet
Sprite X start position	8 Bytes	SEPA	R_WM_SpriteCreate
Sprite width (i.e. X direction dimension)	8 Bytes	SEPA	R_WM_SpriteCreate
CLUT start address	4 Bytes	VDCE	R_WM_WindowClutSet, R_WM_ScreenColorCurveSet

Revision History	Renesas Graphics Library Window Manager (WM) Driver User's Manual: Software

Rev.	Date	Description	
		Page	Summary
0.1	Nov 29, 2018	-	First edition.
0.2	Dec 20, 2018	25, 26, 84	Added the section 3.2.2.3 Gamma correction (1) and moved the detail description of R_WM_ScreenColorCurveSet.
		27, 86	Added the section 3.2.2.4 Gamma correction (2) and moved the detail description of R_WM_ScreenGammaSet.
		33 - 36	Changed the chapter structure; Moved from 3.2.5.1 ~ 3.2.5.4 to 3.2.4.2 ~ 3.2.4.5 Moved from 3.2.4.2 to 3.2.5.2
		37 - 45	Added the chapter 3.2.5
		48	Changed the description of Pitch of Sprite window.
0.3	Mar 28, 2019	10-12, 14	Changed the status transition specification. R_WM_DevDeinit R_WM_ScreenEnable R_WM_WindowNewDrawBufGet R_WM_WindowVisibleBufGet R_WM_WindowCurrentDrawBufGet R_WM_Cap_CapBufGet R_WM_Cap_DispBufGet R_WM_ErrorCallbackSet
		22	Added the description of heap memory.
		25, 87	Deleted the limitation of gamma setting.
		29	Changed bpp of YCbCr422.
		36	Changed the example code.
		37	Added the description of window feature.
		47	Fixed the parameter range of RLE window.
		49	Fixed the parameter range of Sprite data.
		59-63	Changed the description for H/W update.
		71	Added the range of message queue size.
		73	Added the range of scan line.
		71-167	Added the "const" to argument. Fixed the return code.
		163	Changed the error callback specification.
		178	Added the description for dither mode.
		12, 21, 51, 70, 166, 169,	Fixed typo.
1.0	June 12, 2019	8, 9	Improved the description of Error Handling.
		12	Changed the error condition of R_WM_WindowNewDrawBufGet.
		22	Added the description of message queue.
		23	Added the range of PixelClock.
		33, 39, 47, 48, 106, 108, 198	Fixed the range of PosX, PosY and Pitch.
		72 - 169	Fixed the description of Return codes and error callback.

		75	Added the precondition.
		124	Added the description of execution timing.
		130	Added the description of ColorFormat.
		178, 179	Fixed the description of error code.
1.1	Oct 10, 2019	6, 7, 33, 34, 36, 77~80, 86~92, 93	Added the description for VOCA and DISCOM control.
		8, 11, 100, 101, 263, 276	Added the new event type to r_wm_EventId_t. 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
		10, 119, 121, 232, 251, 253	Added the new error code to r_wm_Error_t. R_WM_ERR_SYS_WIN_SCALED_SET_FAILED R_WM_ERR_SYS_CAPTURE_MOVE_FAILED R_WM_ERR_SYS_CAPTURE_RESIZE_FAILED R_WM_ERR_SYS_CAPTURE_SET_VSYNC_FAILED R_WM_ERR_SYS_VOCA_CREATE_FAILED R_WM_ERR_SYS_VOCA_DELETE_FAILED R_WM_ERR_SYS_VOCA_ENABLE_FAILED R_WM_ERR_SYS_VOCA_SET_FAILED R_WM_ERR_SYS_ACT_MON_FAILED R_WM_ERR_SYS_DISCOM_CREATE_FAILED R_WM_ERR_SYS_DISCOM_DELETE_FAILED R_WM_ERR_SYS_DISCOM_ENABLE_FAILED R_WM_ERR_SYS_DISCOM_SET_FAILED R_WM_ERR_SYS_DISCOM_GET_FAILED R_WM_ERR_VOCA_NOT_FOUND R_WM_ERR_DISCOM_NOT_FOUND R_WM_ERR_VOCA_INTERNAL R_WM_ERR_DISCOM_INTERNAL
		13, 14, 21, 22, 29, 30, 31, 94, 96, 97, 123~142, 196, 197	Added the new functions for VOCA. R_WM_ScreenVocaInit R_WM_ScreenVocaDeInit R_WM_ScreenVocaCreate R_WM_ScreenVocaDelete R_WM_ScreenVocaEnable R_WM_ScreenVocaDisable R_WM_ScreenVocaExpImgSet R_WM_ScreenVocaClutSet R_WM_ScreenActivityMonEnable R_WM_ScreenActivityMonDisable R_WM_FrameExecuteVoca
		14, 15, 23, 32, 94, 97, 198, 199, 237~248	Added the new functions for DISCOM. R_WM_FrameExecuteDiscom R_WM_DiscomCreate R_WM_DiscomDelete R_WM_DiscomEnable R_WM_DiscomDisable R_WM_DiscomCrcSet R_WM_DiscomCrcGet

		14, 15, 17, 20, 31, 94, 97, 188, 189, 224~231	Added the new functions for window and capture feature. R_WM_WindowScaledSizeSet R_WM_CaptureMove R_WM_CaptureResize R_WM_CaptureScaledSizeSet R_WM_CaptureExtVsyncSet
		25	Added the window allocation example.
		27, 212	Change specifications that two capture inputs can be synthesized in one WM unit.
		62	Changed the specification of maximum PosY value of sprite data. (8191->4095)
		64~76	Fixed the description for capturing.
		103, 122	Added the preconditions with VOCA and DISCOM. R_WM_DevDeinit R_WM_ScreenDisable
		155	Fixed the description for Width and Height. R_WM_WindowResize.
		222	Fixed the description for Call form Interrupt. R_WM_Cap_DispBufGet.
		249	Added the new definition. R_WM_VOCA_CLUT_NUM
		264, 265, 277	Added the new mode to r_wm_CapMode_t. R_WM_CAPMODE_BIG_ENDIAN R_WM_CAPMODE_DE_MODE R_WM_CAPMODE_PAL R_WM_CAPMODE_EAV R_WM_CAPMODE_SYNC_ONLY R_WM_CAPMODE_BOB_TOP R_WM_CAPMODE_BOB_BOTTOM
		268, 279, 280, 281	Added the new types r_wm_VocaStatus_t r_wm_VocaClutEntry_t r_wm_Voca_t r_wm_Discom_t
		278	Changed the specification how to specify DE mode.
1.2	Nov 29, 2019	62, 63	Added the description about creating more than 16 sprite data.
		86, 279	Added ‘ExpSize’ and ‘ExpImg’ to the structure member of r_wm_Voca_t. Delete ‘Status’ from the structure member of r_wm_Voca_t.
		88	Added the setting function.
		135	Changed argument name from ‘Size’ to ‘ExpSize’. Changed the description of argument.
		135, 137	Added the restriction of Vsync interval.
		139	Changed the parameter range.
1.3	Dec 25, 2019	87, 135, 279	Changed the maximum size of ‘ExpSize’ from 4096 to 2048.
		260	Fixed the integer value in r_wm_OutColorFmt_t.
2.0	May 10, 2020	23	Changed the status transition table of R_WM_DiscomCrcGet
		50	Fixed the support of scaling-up with RLE window.
		87	Fixed the minimum size of (PosX + Width) and (PosY + Height). Added the range information of (RamAddr + ExpSize)
		92	Fixed the minimum size of (PosX + Width) and (PosY + Height).
		103, 148	Moved the precondition to delete the sprite data from R_WM_DevDeinit to R_WM_WindowDelete.
		224, 226	Added the restriction of dynamic change. R_WM_CaptureMove R_WM_CaptureResize

Renesas Graphics Library
Window Manager (WM)
User's Manual: Software
Publication Date: Rev.0.1 Nov 29, 2018
Rev.2.0 May 10, 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/

Colophon 3.1

Renesas Graphics Library
Window Manager (WM) Driver

Renesas Electronics Corporation