

RZ/A2M Group

Graphics Library "RGA"

Introduction

This application note describes the Graphics Library RGA (Renesas Graphics Architecture) of RZ/A2M.

The following lists features of the RGA.

- Allows high-speed drawing using the hardware acceleration.
- The API created based on the W3C standard HTML Canvas 2D Context achieves easy learning.
- The memory area provided by the application is available as a drawing destination or for input images.
- Allows drawing of translucent images and translucent drawing using an alpha mask.
- The RGA provides a conversion tool that can access image files as global variables. (This conversion tool operates on the host PC.)

Target Device

RZ/A2M

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Restrictions

This library is not supported with the vector graphics supported.

This library cannot be used in multiple threads. Use the RGA in a single thread.

This library supports only affine transformation.

Contents

1. Specifications	7
2. File Configuration	12
3. Operation Confirmation Conditions	13
4. Description of Software.....	15
4.1 Outline of Operations	15
4.1.1 When Drawing on a Buffer Defined by the Application	15
(1) Flowchart.....	15
(2) Sequence Chart	15
4.1.2 Drawing on the Display Screen	16
(1) Flowchart.....	16
(2) Sequence Chart	17
4.2 Constants, Type, Classes and Functions.....	19
4.2.1 Comparison with RZ/A1 RGA.....	19
4.2.2 Error Code	22
4.2.3 Figure of byte_per_pixel_t.....	22
4.3 Porting Guide.....	23
4.4 Supplementary Explanation	24
4.4.1 Correspondence to Canvas 2D and Correspondence to Hardware Acceleration	24
4.4.2 Internal operation in initialize function	26
4.4.3 Identifying Image Format.....	27
4.4.4 Defaultable Flags	28
4.4.5 Function to Initialize Internal Variables with Constants (*_initConst Function)	30
4.4.6 Finalize (*_Finalize Function)	31
5. Functions and Methods	33
5.1 Functions Equivalent to graphics_t Class Member Function	33
5.1.1 List of Functions	33
5.1.2 R_GRAPHICS_STATIC_GetVersion	35
5.1.3 R_GRAPHICS_PrintRegisters	35
5.1.4 R_GRAPHICS_Start	35
5.1.5 R_GRAPHICS_FinishPreviousFrame	35
5.1.6 R_GRAPHICS_StartPreviousFrame	35
5.1.7 R_GRAPHICS_GetHasFramePipeline	36
5.1.8 R_GRAPHICS_BeginSoftwareRendering2.....	36
5.1.9 R_GRAPHICS_BeginSoftwareRenderingA	36
5.1.10 R_GRAPHICS_CONFIG_SetEmpty	36
5.1.11 R_GRAPHICS_InitConst.....	36
5.1.12 R_GRAPHICS_Initialize	37

5.1.13	R_GRAPHICS_Finalize.....	37
5.1.14	R_GRAPHICS_SetFrameBuffer.....	37
5.1.15	R_GRAPHICS_GetFrameBuffer	38
5.1.16	R_GRAPHICS_Finish	38
5.1.17	R_GRAPHICS_Save.....	38
5.1.18	R_GRAPHICS_Restore	39
5.1.19	R_GRAPHICS_ResetMatrix.....	39
5.1.20	R_GRAPHICS_SetMatrix_2x3	39
5.1.21	R_GRAPHICS_SetMatrix_3x3	39
5.1.22	R_GRAPHICS_GetMatrix_3x3.....	40
5.1.23	R_GRAPHICS_TranslateMatrixl	40
5.1.24	R_GRAPHICS_TranslateMatrix	40
5.1.25	R_GRAPHICS_ScaleMatrix	41
5.1.26	R_GRAPHICS_RotateMatrixDegree.....	41
5.1.27	R_GRAPHICS_ShearMatrix.....	42
5.1.28	R_GRAPHICS_TransformMatrix.....	43
5.1.29	R_GRAPHICS_MultiplyMatrix	43
5.1.30	R_GRAPHICS_GetProjectiveMatrix.....	44
5.1.31	R_GRAPHICS_SetBackgroundColor.....	44
5.1.32	R_GRAPHICS_GetBackgroundColor	45
5.1.33	R_GRAPHICS_GetClearColor	45
5.1.34	R_GRAPHICS_Clear	45
5.1.35	R_GRAPHICS_DrawImage.....	46
5.1.36	R_GRAPHICS_DrawImageResized.....	47
5.1.37	R_GRAPHICS_DrawImageChild	48
5.1.38	R_GRAPHICS_FillRect	49
5.1.39	R_GRAPHICS_SetFillColor	49
5.1.40	R_GRAPHICS_SetFillPattern	49
5.1.41	R_GRAPHICS_BeginPath	49
5.1.42	R_GRAPHICS_Rect.....	50
5.1.43	R_GRAPHICS_Clip	50
5.1.44	R_GRAPHICS_SetGlobalAlpha	50
5.1.45	R_GRAPHICS_GetGlobalAlpha.....	51
5.1.46	R_GRAPHICS_SetGlobalCompositeOperation	51
5.1.47	R_GRAPHICS_GetGlobalCompositeOperation.....	51
5.1.48	R_GRAPHICS_SetQualityFlags.....	51
5.1.49	R_GRAPHICS_GetQualityFlags	52
5.1.50	R_GRAPHICS_SetStrokeColor.....	52
5.1.51	R_GRAPHICS_StrokeRect	52
5.1.52	R_GRAPHICS_BeginSoftwareRendering	52
5.1.53	R_GRAPHICS_EndSoftwareRendering.....	53

5.1.54	R_GRAPHICS_EndRenderingInFin	53
5.2	Functions Equivalent to graphics_image_t Class Member Function	54
5.2.1	List of Functions	54
5.2.2	R_GRAPHICS_IMAGE_InitByShareFrameBuffer	54
5.2.3	R_GRAPHICS_IMAGE_InitR8G8B8A8	55
5.2.4	R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8	55
5.2.5	R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8	56
5.2.6	R_GRAPHICS_IMAGE_GetProperties	56
5.2.7	R_GRAPHICS_IMAGE_InitByShareFrameBufferEx	57
5.2.8	R_GRAPHICS_IMAGE_GetImageFormat	57
5.3	Functions Equivalent to graphics_pattern_t Class Member Function	57
5.3.1	List of Functions	57
5.3.2	R_GRAPHICS_PATTERN_Initialize	57
5.4	Functions Equivalent to window_surfaces_t Class Member Functions	58
5.4.1	List of Functions	58
5.4.2	R_WINDOW_SURFACES_InitConst	58
5.4.3	R_WINDOW_SURFACES_Initialize	59
5.4.4	R_WINDOW_SURFACES_Finalize	59
5.4.5	R_WINDOW_SURFACES_GetLayerFrameBuffer	59
5.4.6	R_WINDOW_SURFACES_GetLayerCount	59
5.4.7	R_WINDOW_SURFACES_SwapBuffers	60
5.4.8	R_WINDOW_SURFACES_DoMessageLoop	60
5.4.9	R_WINDOW_SURFACES_AccessLayerAttributes	60
5.4.10	R_WINDOW_SURFACES_AllocOffscreenStack	61
5.4.11	R_WINDOW_SURFACES_FreeOffscreenStack	61
5.4.12	R_WINDOW_SURFACES_SetLayerAttributes	61
5.4.13	R_WINDOW_SURFACES_CONFIG_SetEmpty	62
5.4.14	R_LAYER_ATTRIBUTES_SetEmpty	62
5.5	Functions Related to byte_per_pixel_t Class	63
5.5.1	List of Functions	63
5.5.2	R_RGA_BitPerPixelType_To_BytePerPixelType	63
5.5.3	R_RGA_BytePerPixelType_To_BitPerPixelType	63
5.5.4	R_BYTE_PER_PIXEL_IsInteger	63
5.5.5	R_BIT_PER_PIXEL_GetBytePerPixel	64
5.5.6	R_BYTE_PER_PIXEL_GetBitPerPixel	64
5.5.7	R_RGA_BYTE_PER_PIXEL_IsInteger	64
5.6	Functions Related to animation_timing_function_t Class	65
5.6.1	List of Functions	65
5.6.2	R_Get_AnimationTimingFunction	65
5.6.3	R_ANIMATION_TIMING_FUNCTION_GetValue	66
5.7	OS Porting Layer of JCU driver	67

5.7.1	List of Functions	67
5.7.2	R_GRAPHICS_OnInitialize	67
5.7.3	R_GRAPHICS_OnFinalize	67
5.7.4	R_GRAPHICS_OnInitialized	68
5.7.5	R_GRAPHICS_JCU_ClearCodecEvent.....	68
5.7.6	R_GRAPHICS_JCU_SetCodecEvent.....	68
5.7.7	R_GRAPHICS_JCU_WaitForCodecEvent.....	68
5.7.8	R_GRAPHICS_JCU_ClearTerminateEvent.....	68
5.7.9	R_GRAPHICS_JCU_SetTerminateEvent	68
5.7.10	R_GRAPHICS_JCU_WaitForTerminateEvent.....	69
5.8	OS Porting Layer of DRW driver	70
5.8.1	List of Functions	70
5.8.2	R_DRW_OnInitialize	70
5.8.3	R_DRW_OnFinalize	70
5.8.4	R_DRW_EnableInterrupt.....	70
5.8.5	R_DRW_DisableInterrupt.....	71
5.8.6	R_DRW_WaitForInterruptEvent.....	71
5.8.7	R_DRW_SetInterruptEvent	72
5.8.8	R_DRW_CheckThread.....	72
5.8.9	R_DRW_FlushCache	72
5.8.10	R_DRW_ToPhysicalAddress	73
5.8.11	R_DRW_ToCachedAddress	73
5.8.12	R_DRW_ToUncachedAddress.....	74
5.8.13	R_DRW_OnInterrupting	74
5.9	Other Functions	75
5.9.1	List of Functions	75
5.9.2	R_RGA_Get_R8G8B8A8	75
5.9.3	R_RGA_CalcWorkBufferB_Size	76
6.	Tools	77
6.1	Image Format Conversion by ImagePackager.....	77
6.1.1	Operational Procedure	77
6.1.2	List of files.....	78
6.1.3	Sample	78
6.1.4	Types of Output Binary File (Language)	79
6.1.5	File Formats in the Output Binary File.....	79
6.1.6	Input Formats	80
6.1.7	Parameters That Can Be Described in BinaryImageConfig.image.xml	81
6.1.8	Basic forms of writing XML.....	89
(1)	XML	89
(2)	XPath.....	90

(3)	# fragment	90
6.2	Converting binary by ConvertBin.....	91
6.3	Creating image file by RawToBmp.....	92
7.	Reference Documents	93
7.1	Reference Symbols	94
	Revision History	95

1. Specifications

The RGA is used to draw graphics images. Table 1-1 lists Peripheral Functions to Be Used and Applications, Figure 1-1 shows a Block Diagram, and Table 1-2 and Table 1-3 list available pixel formats.

Table 1.1 **Peripheral device used**

Peripheral device	Usage
2D Drawing Engine (DRW)	Graphics drawing
JPEG Codec Unit (JCU)	JPEG decompression
Video Display Controller 6 (VDC6)	Screen display

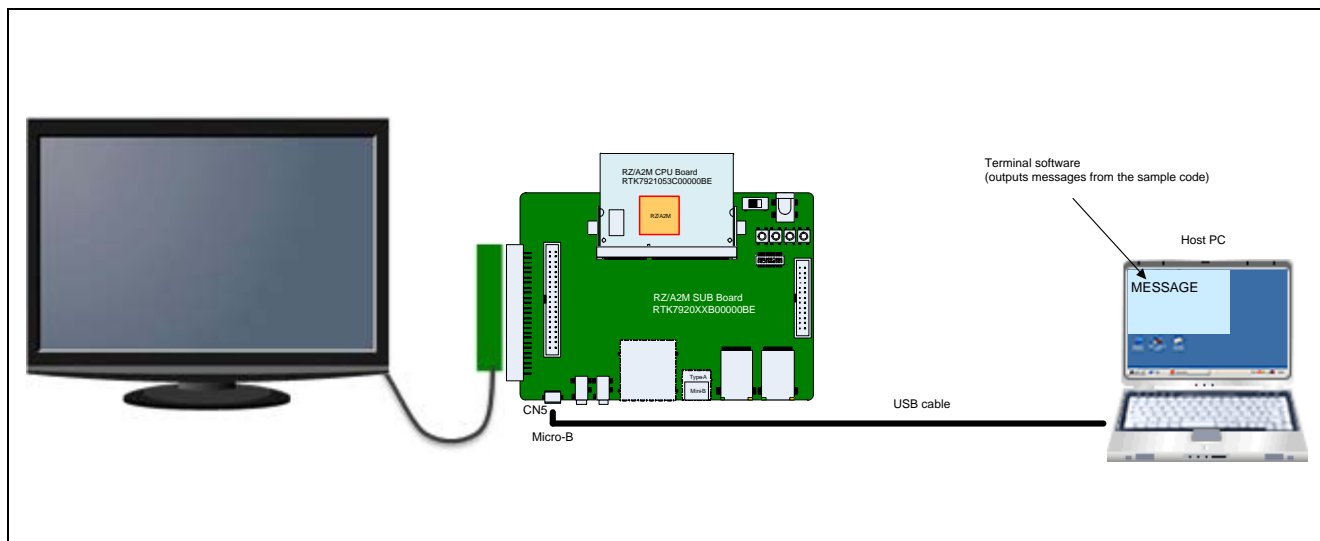


Figure 1-1 Operation check conditions

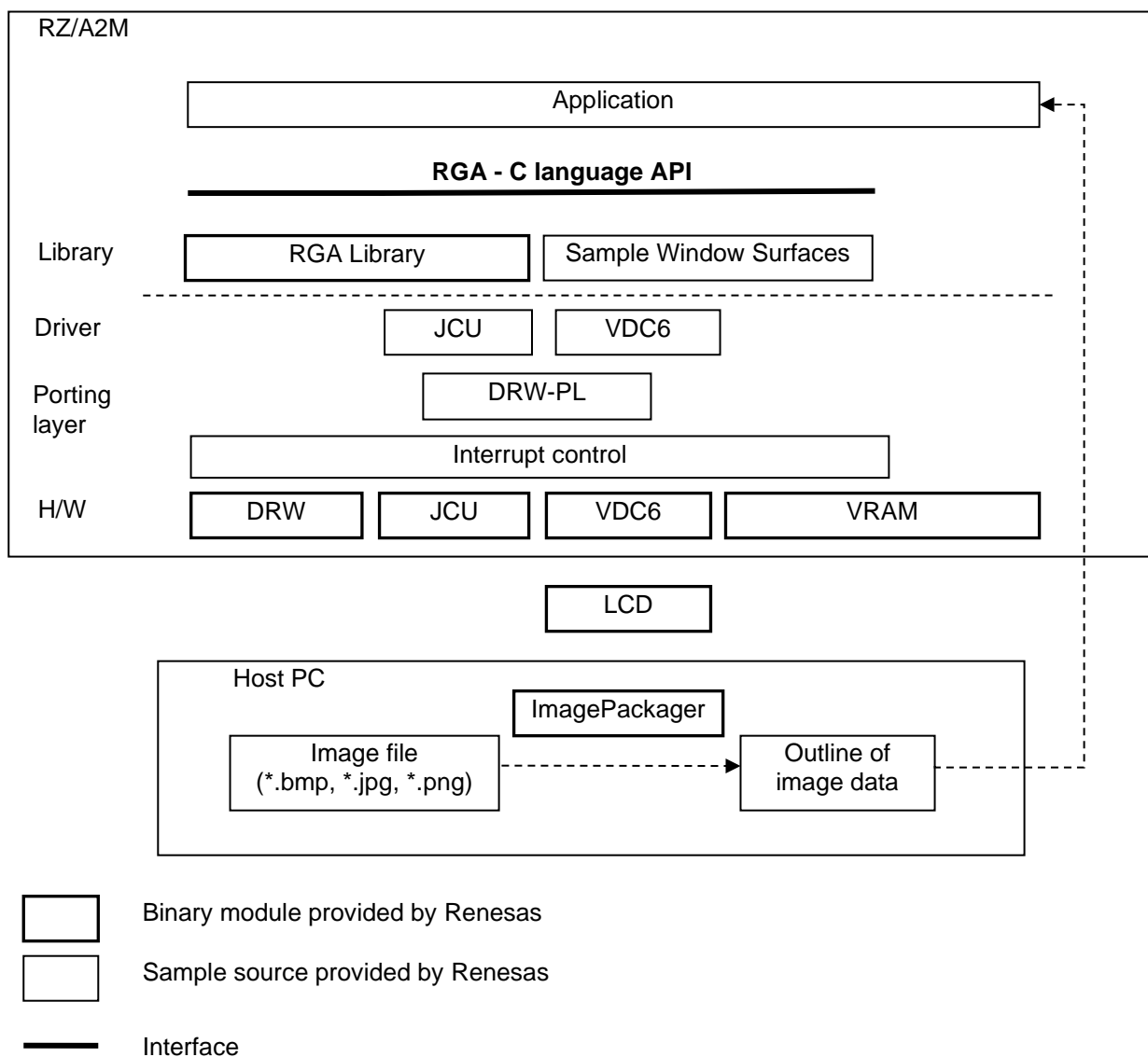


Figure 1-2 Block Diagram

The coordinate system of this library has an origin at the upper left. The value in the X-axis direction increases from left to right. The value in the Y-axis direction increases downward. The maximum width of a frame buffer to be drawn is 2048 pixels, and its maximum height is 2048 pixels. The maximum size of source image is 1024x1024. Maximum range of one fill and image enlargement (bounding box) is 1024x1024. Right end and bottom end of bounding box must not out of frame buffer. First address and size of 1 line of the frame buffer and source image must be a multiple of 32.

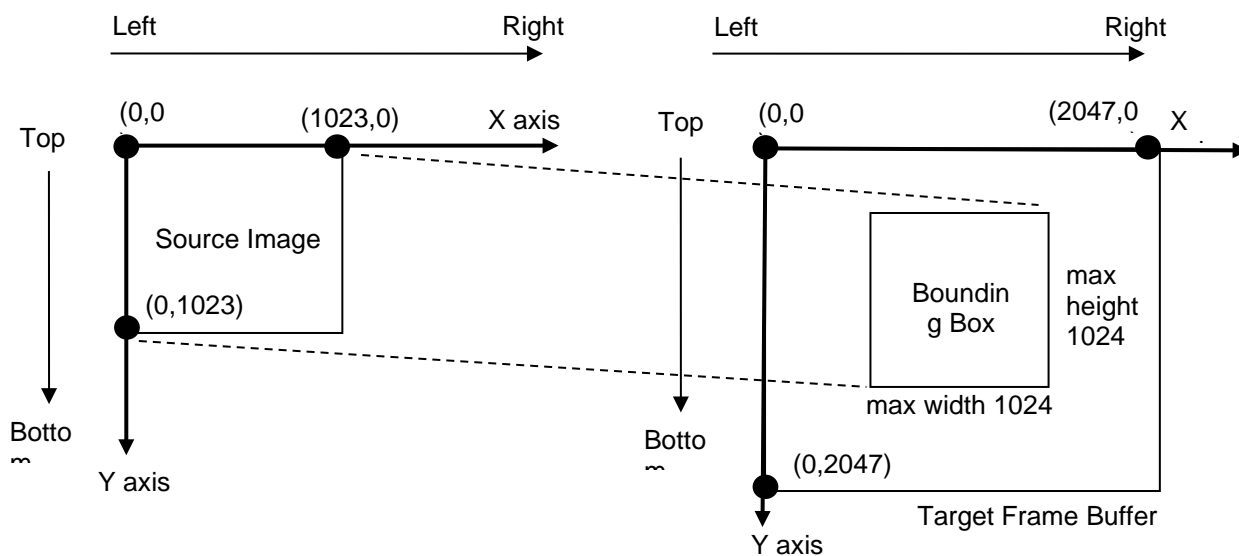


Figure 1-3 Max size of drawing target and source image

Table 1-2 Available Pixel Formats of Drawing Destination

	XRGB 8888	ARGB 8888	RGB 565	ARGB 4444	A8
Hardware rendering	✓	✓	✓	✓	✓
Software rendering	✓	✓	✓	✓	x
Matrix, enlargement/ reduction, blend	✓	✓	✓	✓	✓
Image drawing	✓	✓	✓	✓	✓
DrawImageChild	✓	✓	✓	✓	✓
Alpha only image	✓	✓	✓	✓	✓
Square fill	✓	✓	✓	✓	x
Byte per pixel (reference)	4	4	2	2	1

✓=Available, x=Not available

Table 1-3 Combinations of Available Pixel Formats of Images and Pixel Formats of Drawing Destination

Output Input Image	XRGB 8888	ARGB 8888	RGB 565	ARGB 4444	A8
JPEG	✓	✓	✓	✓	x
PNG	✓	✓	✓	✓	x
XRGB8888	✓	✓	✓	✓	x
ARGB8888	✓	✓	✓	✓	x
RGB565	✓	✓	✓	✓	x
ARGB4444	✓	✓	✓	✓	x
CLUT8	(x)	(x)	(x)	(x)	x
CLUT4	(x)	(x)	(x)	(x)	x
CLUT1	(x)	(x)	(x)	(x)	x
A8	✓	✓	✓	✓	✓

✓=Available, x=Not available, (x)=Not available but hardware is supported

The JPEG above is a case where JPEG data is specified for arguments of the image drawing function (R_GRAPHICS_DrawImage). Table 1-4 shows supported JPEG format. When a JPEG file or PNG file is converted to the raw format (including XRGB8888) by using the ImagePackager tool, see the column of the pixel format.

Table 1-4 Supported JPEG format

Decoding module	JPEG Codec Unit (JCU) in RZ/A
JPEG standard	baseline
Pixel format in JPEG	YCbCr420 (H=2:1:1,V=2:1:1) YCbCr422 (H=2:1:1,V=1:1:1) YCbCr444 (H=1:1:1,V=1:1:1) YCbCr411 (H=4:1:1,V=1:1:1)

Table 1-5 Supported PNG format

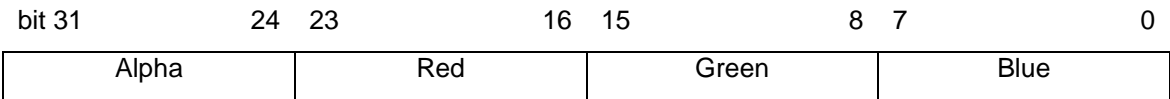
Decoding module	LibPNG, zlib
-----------------	--------------

The following tables describes the detail of pixel format. RGA for RZ/A is little endian. For example, Red of XRGB8888 is at first address of the pixel + 2.

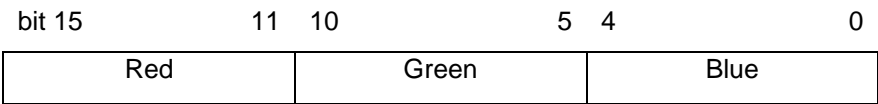
XRGB8888

bit 31	24	23	16	15	8	7	0
0	Red		Green		Blue		

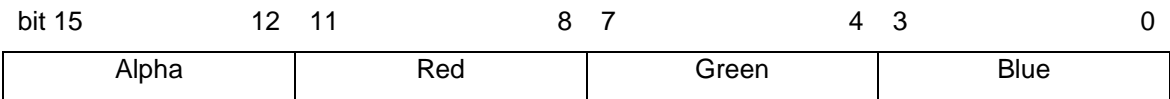
ARGB8888



RGB565



ARGB4444



A8

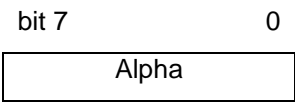
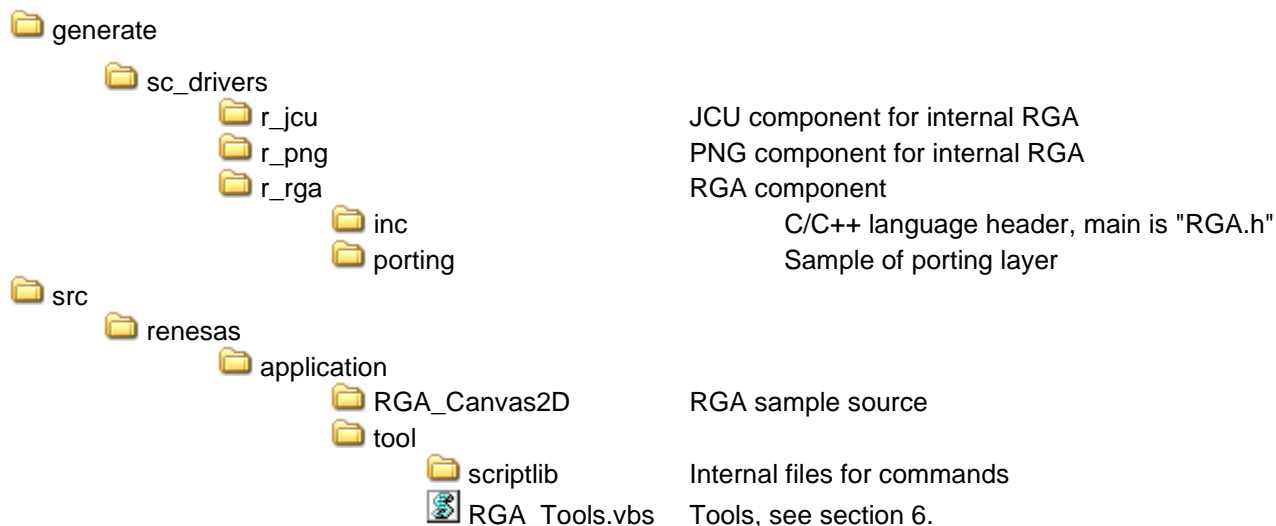


Figure 1-4 Structure of Pixel Format

2. File Configuration



To import RGA library to your project folder, refer to "How to use components bundled in this package" section in "RZ/A2M Graphics RGA Package Release Note" (R01AN4606).

Write '#include "RGA.h"' in the application program using RGA.

If memory area became few, reduce frame buffer to the size of showing only and set the size of work buffer B to 0. It is necessary to increase stack size depending on existing environment. The stack size of interrupt must be enough size.

Check that the memory map by setting in "r_memory_map.c" is same as the setting of MMU.

RGA requests to use vfp (using double register) of floating-point compiler option.

Table 2-1 Main header files (in src\drivers\RGA\inc)

File Name	Description
RGA.h	Main of RGA
RGA_API.h	Sub of RGA : API. This file is included from RGA.h
use_config_RGA.h	Sub of RGA : setting. This file is included from RGA.h

3. Operation Confirmation Conditions

Table 3.1. Operation Confirmation Conditions (1/2)

item	Contents
MCU used	RZ/A2M
Operating frequency (Note)	CPU Clock (I ϕ) : 528MHz Image processing clock (G ϕ) : 264MHz Internal Bus Clock (B ϕ) : 132MHz Peripheral Clock 1 (P1 ϕ) : 66MHz Peripheral Clock 0 (P0 ϕ) : 33MHz QSPI0_SPCLK : 66MHz CKIO : 132MHz
Operating voltage	Power supply voltage (I/O): 3.3 V Power supply voltage (either 1.8V or 3.3V I/O (PVcc SPI)) : 3.3V Power supply voltage (internal): 1.2 V
Integrated development environment	e2studio 2020-07
C compiler	"GNU Arm Embedded Tool chain 6-2017-q2-update" compiler options(except directory path) Release: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0

Note: The operating frequency used in clock mode 1 (Clock input of 24MHz from EXTAL pin)

Table 3.2. Operation Confirmation Conditions (2/2)

Operation mode	Boot mode 3 (Serial Flash boot 3.3V)
Terminal software communication settings	<ul style="list-style-type: none">• Communication speed: 115200bps• Data length: 8 bits• Parity: None• Stop bits: 1 bit• Flow control: None
Board to be used	RZ/A2M CPU board RTK7921053C00000BE RZ/A2M SUB board RTK79210XXB00000BE
Device (functionality to be used on the board)	<ul style="list-style-type: none">• Serial flash memory allocated to SPI multi-I/O bus space (channel 0) Manufacturer : Macronix Inc. Model Name : MX25L51245GXD• RL78/G1C (Convert between USB communication and serial communication to communicate with the host PC.)• LED1

4. Description of Software

4.1 Outline of Operations

4.1.1 When Drawing on a Buffer Defined by the Application

(1) Flowchart

Figure 6.1 shows a flowchart of drawing on a frame buffer defined by the application.

For the actual operation procedure, see the attached document, RGA Tutorial.

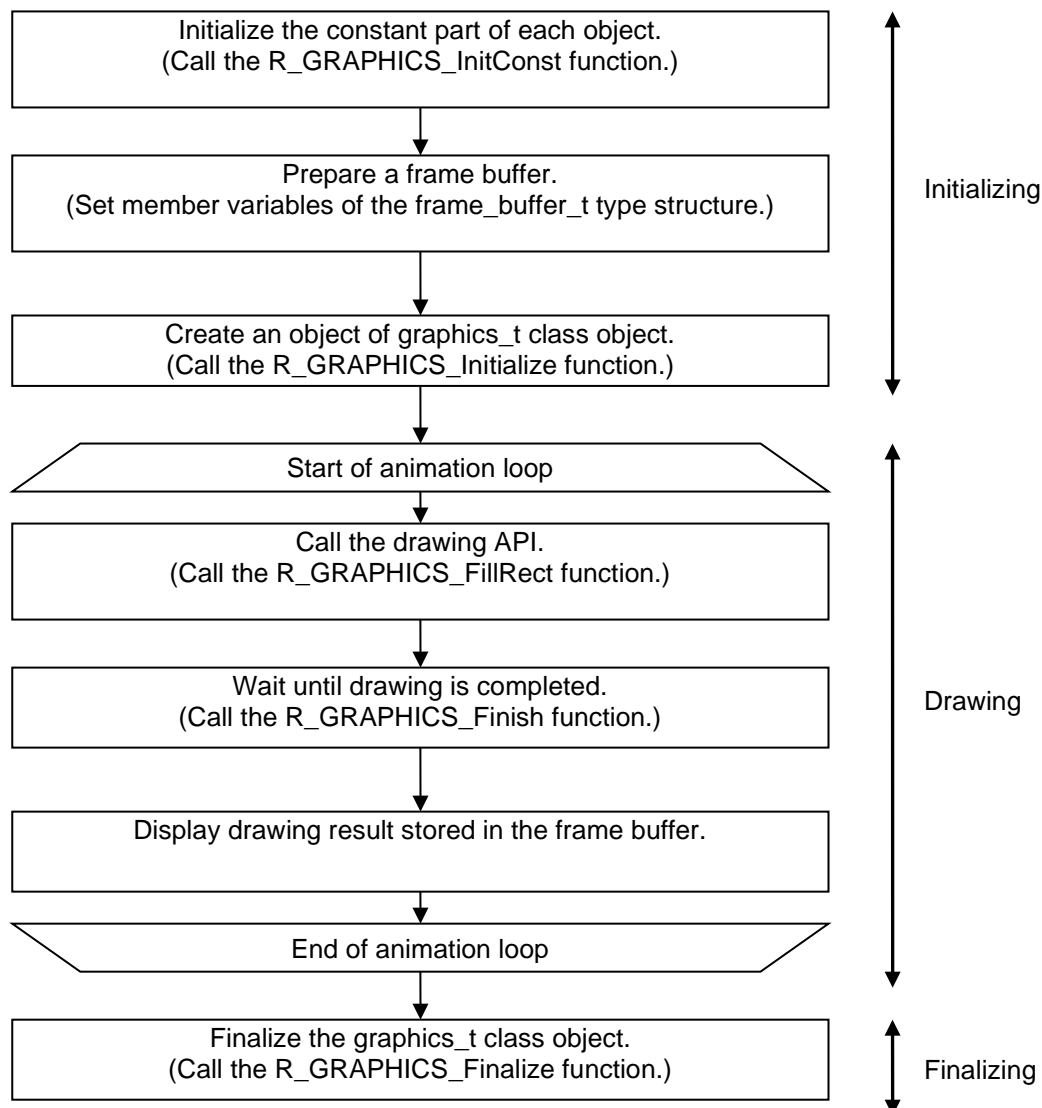


Figure 4.1 Drawing on a Buffer Defined by the Application - Flowchart

(2) Sequence Chart

The following shows an operation timing chart of the software (application and library) and hardware (drawing hardware and display hardware) in the case of drawing on the frame buffer provided by the application. Also, the process returns from drawing API after finishing the drawing, if the software rendering was used.

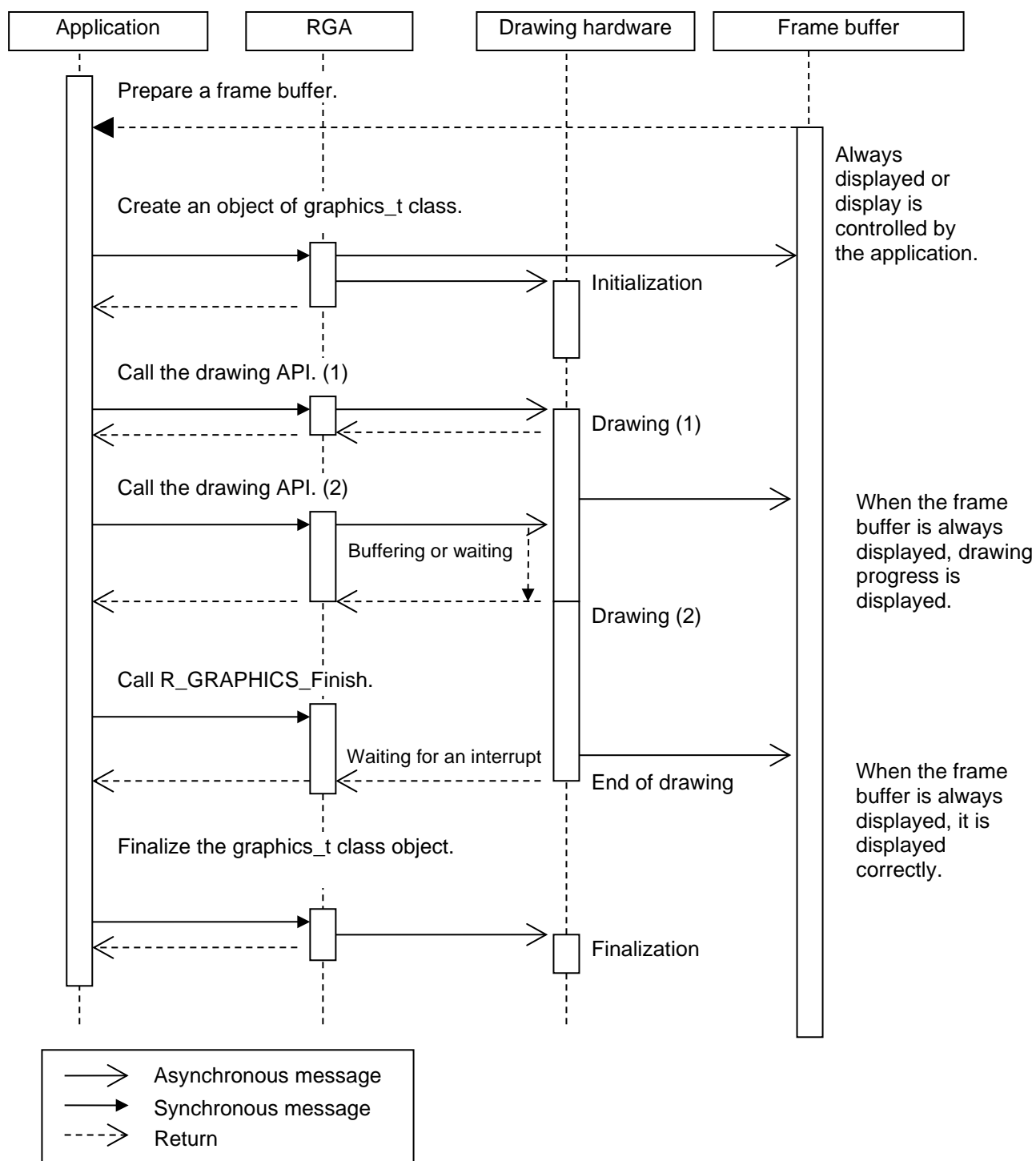


Figure 4.2 Operation sequence

4.1.2 Drawing on the Display Screen

(1) Flowchart

Figure 4.3 shows a flowchart of drawing on the display screen.

For the actual operation procedure, see the attached document, RGA Tutorial.

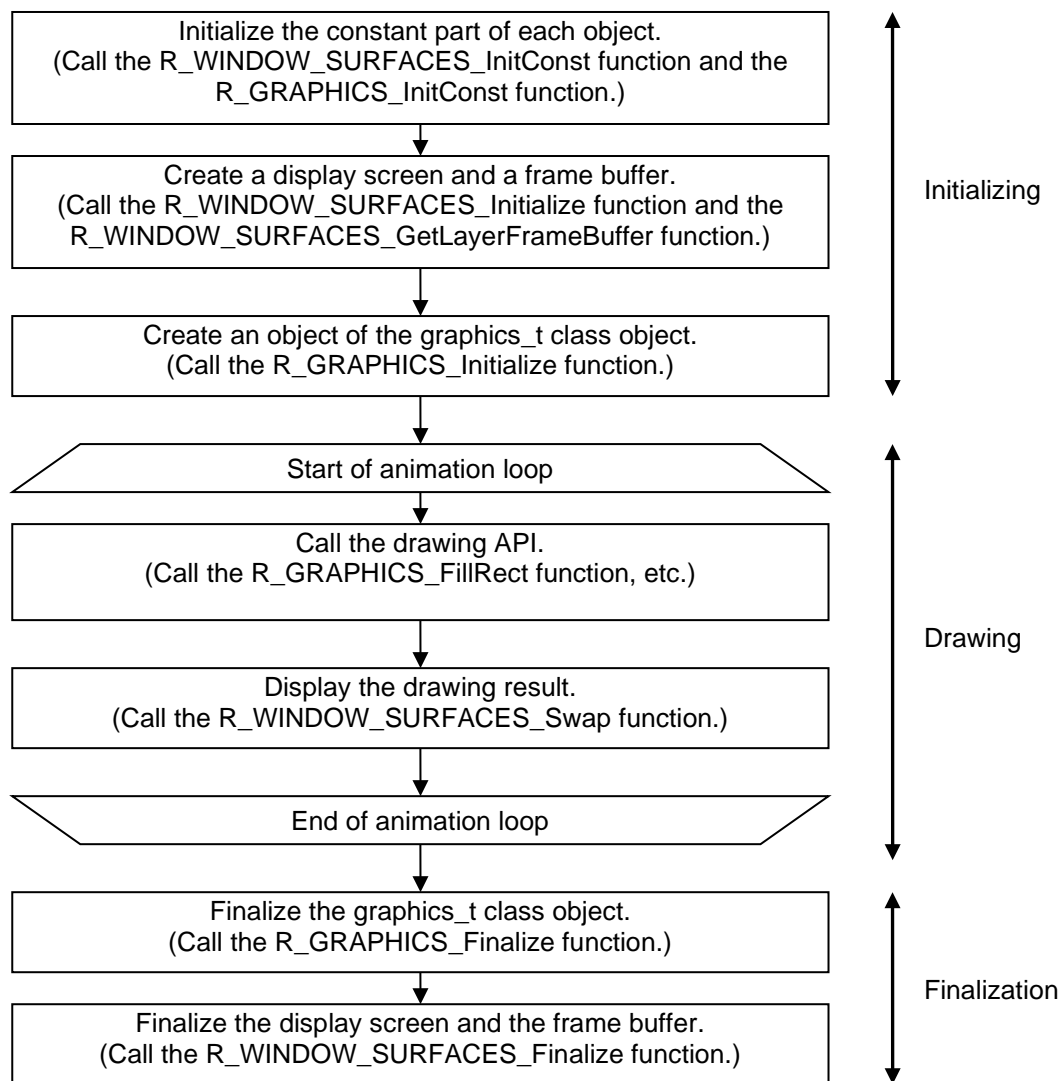


Figure 4.3 Drawing on the Display Screen - Flowchart Processing

(2) Sequence Chart

The following shows an operation timing chart of the software (application and library) and hardware (drawing hardware and display hardware) in the case of drawing on the frame buffer provided by the RGA's WindowSurfaces library. Also, the process returns from drawing API after finishing the drawing, if the software rendering was used.

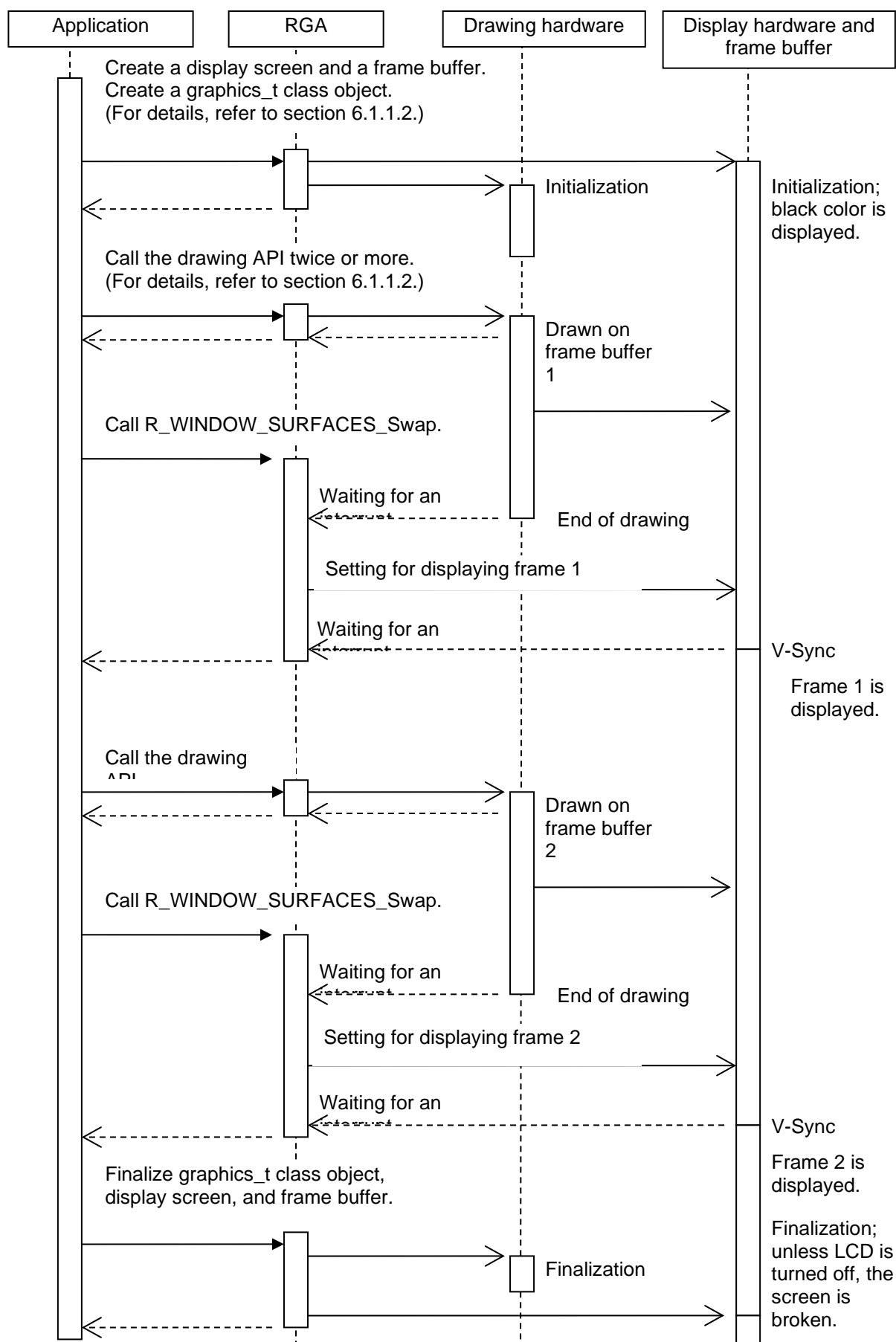


Figure 4.4 Operation sequence

4.2 Constants, Type, Classes and Functions

Refer the HTML file attached the project.

4.2.1 Comparison with RZ/A1 RGA

Types and functions defined by RGA for RZ/A2M is same as types defined by RGA for RZ/A1H. But some types and functions are not supported. The following table describes supported types and functions. C++ classes are not supported. --- means not supported.

Types/Classes

RGA for RZ/A2M	RGA for RZ/A1H
frame_buffer_t ^{*1}	frame_buffer_t
graphics_t	graphics_t
graphics_config_t ^{*1}	graphics_config_t
graphics_quality_flag_t	graphics_quality_flag_t
window_surfaces_t	window_surfaces_t
window_surfaces_config_t ^{*1}	window_surfaces_config_t
layer_attributes_t ^{*1}	layer_attributes_t
---	access_t
---	video_input_t
---	video_input_config_t
byte_per_pixel_t	byte_per_pixel_t
pixel_format_t	pixel_format_t
frame_buffer_delegate_t	frame_buffer_delegate_t
---	v_sync_t
---	vram_stack_t
---	vram_stack_config_t
---	vram_ex_stack_t
graphics_image_t	graphics_image_t
graphics_image_properties_t	graphics_image_properties_t
graphics_composite_operation_t	graphics_composite_operation_t
graphics_status_t	graphics_status_t
graphics_matrix_float_t	graphics_matrix_float_t
repetition_t	repetition_t
r8g8b8a8_t	r8g8b8a8_t
---	background_format_t
graphics_pattern_t	graphics_pattern_t
animation_timing_function_t	animation_timing_function_t
graphics_jpeg_decoder_t	graphics_jpeg_decoder_t
---	graphics_async_status_t

^{*1} The specification was changed.

Functions/Methods

RGA for RZ/A2M	RGA for RZ/A1H
R_GRAPHICS_InitConst	R_GRAPHICS_InitConst
R_GRAPHICS_Initialize	R_GRAPHICS_Initialize
R_GRAPHICS_Finalize	R_GRAPHICS_Finalize
R_GRAPHICS_CONFIG_SetEmpty	---
R_GRAPHICS_SetFrameBuffer	R_GRAPHICS_SetFrameBuffer
R_GRAPHICS_GetFrameBuffer	R_GRAPHICS_GetFrameBuffer
R_GRAPHICS_Start	---
R_GRAPHICS_StartPreviousFrame	---
R_GRAPHICS_FinishPreviousFrame	---
R_GRAPHICS_Finish	R_GRAPHICS_Finish
---	R_GRAPHICS_FinishStart
R_GRAPHICS_GetHasFramePipeline	---
---	R_GRAPHICS_GetAsyncStatus
---	R_GRAPHICS_OnInterrupting
R_GRAPHICS_Save	R_GRAPHICS_Save
R_GRAPHICS_Restore	R_GRAPHICS_Restore
R_GRAPHICS_ResetMatrix	R_GRAPHICS_ResetMatrix
R_GRAPHICS_SetMatrix_2x3	R_GRAPHICS_SetMatrix_2x3
R_GRAPHICS_SetMatrix_3x3	R_GRAPHICS_SetMatrix_3x3
R_GRAPHICS_GetMatrix_3x3	R_GRAPHICS_GetMatrix_3x3
R_GRAPHICS_TranslateMatrixI	R_GRAPHICS_TranslateMatrixI
R_GRAPHICS_TranslateMatrix	R_GRAPHICS_TranslateMatrix
R_GRAPHICS_ScaleMatrix	R_GRAPHICS_ScaleMatrix
R_GRAPHICS_RotateMatrixDegree	R_GRAPHICS_RotateMatrixDegree
R_GRAPHICS_ShearMatrix	R_GRAPHICS_ShearMatrix
R_GRAPHICS_TransformMatrix	R_GRAPHICS_TransformMatrix
R_GRAPHICS_MultiplyMatrix	R_GRAPHICS_MultiplyMatrix
R_GRAPHICS_GetProjectiveMatrix	R_GRAPHICS_GetProjectiveMatrix
R_GRAPHICS_SetBackgroundColor	R_GRAPHICS_SetBackgroundColor
R_GRAPHICS_GetBackgroundColor	R_GRAPHICS_GetBackgroundColor
R_GRAPHICS_GetClearColor	R_GRAPHICS_GetClearColor
R_GRAPHICS_Clear	R_GRAPHICS_Clear
R_GRAPHICS_DrawImage	R_GRAPHICS_DrawImage
R_GRAPHICS_DrawImageResized	R_GRAPHICS_DrawImageResized
R_GRAPHICS_DrawImageChild	R_GRAPHICS_DrawImageChild
R_GRAPHICS_FillRect	R_GRAPHICS_FillRect
R_GRAPHICS_SetFillColor	R_GRAPHICS_SetFillColor
R_GRAPHICS_SetFillPattern	R_GRAPHICS_SetFillPattern
R_GRAPHICS_BeginPath	R_GRAPHICS_BeginPath
R_GRAPHICS_Rect	R_GRAPHICS_Rect
R_GRAPHICS_Clip	R_GRAPHICS_Clip
R_GRAPHICS_SetGlobalAlpha	R_GRAPHICS_SetGlobalAlpha
R_GRAPHICS_GetGlobalAlpha	R_GRAPHICS_GetGlobalAlpha
R_GRAPHICS_SetGlobalCompositeOperation	R_GRAPHICS_SetGlobalCompositeOperation
R_GRAPHICS_GetGlobalCompositeOperation	R_GRAPHICS_GetGlobalCompositeOperation
---	R_GRAPHICS_STATIC_SetOnInitialize
---	R_GRAPHICS_STATIC_SetOnFinalize

R_GRAPHICS_OnInitialize *1	R_GRAPHICS_STATIC_OnInitializeDefault
R_GRAPHICS_OnFinalize *1	R_GRAPHICS_STATIC_OnFinalizeDefault
R_GRAPHICS_SetQualityFlags	R_GRAPHICS_SetQualityFlags
R_GRAPHICS_GetQualityFlags	R_GRAPHICS_GetQualityFlags
R_GRAPHICS_SetStrokeColor	R_GRAPHICS_SetStrokeColor
R_GRAPHICS_StrokeRect	R_GRAPHICS_StrokeRect
R_GRAPHICS_BeginSoftwareRendering	R_GRAPHICS_BeginSoftwareRendering
R_GRAPHICS_EndSoftwareRendering	R_GRAPHICS_EndSoftwareRendering
R_GRAPHICS_EndRenderingInFin	R_GRAPHICS_EndRenderingInFin
---	R_GRAPHICS_BeginSoftwareRendering2
---	R_GRAPHICS_BeginSoftwareRenderingA
R_GRAPHICS_IMAGE_InitByShareFrameBuffer	R_GRAPHICS_IMAGE_InitByShareFrameBuffer
R_GRAPHICS_IMAGE_InitByShareFrameBufferEx	---
R_GRAPHICS_IMAGE_InitR8G8B8A8	R_GRAPHICS_IMAGE_InitR8G8B8A8
R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8	R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8
R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8	R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8
R_GRAPHICS_IMAGE_GetProperties	R_GRAPHICS_IMAGE_GetProperties
R_GRAPHICS_PATTERN_Initialize	R_GRAPHICS_PATTERN_Initialize
R_WINDOW_SURFACES_InitConst	R_WINDOW_SURFACES_InitConst
R_WINDOW_SURFACES_Initialize	R_WINDOW_SURFACES_Initialize
R_WINDOW_SURFACES_Finalize	R_WINDOW_SURFACES_Finalize
---	R_WINDOW_SURFACES_Dispose
R_WINDOW_SURFACES_CONFIG_SetEmpty	---
R_WINDOW_SURFACES_GetLayerFrameBuffer	R_WINDOW_SURFACES_GetLayerFrameBuffer
R_WINDOW_SURFACES_GetLayerCount	R_WINDOW_SURFACES_GetLayerCount
R_WINDOW_SURFACES_SwapBuffers	R_WINDOW_SURFACES_SwapBuffers
---	R_WINDOW_SURFACES_SwapBuffersStart
---	R_WINDOW_SURFACES_WaitForVSync
R_WINDOW_SURFACES_DoMessageLoop	R_WINDOW_SURFACES_DoMessageLoop
R_WINDOW_SURFACES_SetLayerAttributes *1	R_WINDOW_SURFACES_AccessLayerAttributes
R_LAYER_ATTRIBUTES_SetEmpty	
R_WINDOW_SURFACES_AllocOffscreenStack	R_WINDOW_SURFACES_AllocOffscreenStack
R_WINDOW_SURFACES_FreeOffscreenStack	R_WINDOW_SURFACES_FreeOffscreenStack
---	R_VIDEO_INPUT_InitConst
---	R_VIDEO_INPUT_Initialize
---	R_VIDEO_INPUT_Finalize
R_RGA_BitPerPixelType_To_BytePerPixelType	R_RGA_BitPerPixelType_To_BytePerPixelType
R_RGA_BytePerPixelType_To_BitPerPixelType	R_RGA_BytePerPixelType_To_BitPerPixelType
R_BYTE_PER_PIXEL_IsInteger	R_BYTE_PER_PIXEL_IsInteger
---	R_V_SYNC_Initialize
---	R_V_SYNC_Finalize
---	R_V_SYNC_Wait
---	R_V_SYNC_WaitStart
---	R_V_SYNC_OnInterrupting
---	R_V_SYNC_GetAsyncStatus
---	R_VRAM_STACK_Initialize
---	R_VRAM_STACK_AllocateMemory
---	R_VRAM_STACK_FreeMemory
---	R_VRAM_EX_STACK_Initialize
---	R_VRAM_EX_STACK_Alloc
---	R_VRAM_EX_STACK_Free

R_Get_AnimationTimingFunction	R_Get_AnimationTimingFunction
R_ANIMATION_TIMING_FUNCTION_GetValue	R_ANIMATION_TIMING_FUNCTION_GetValue
R_RGA_Get_R8G8B8A8	R_RGA_Get_R8G8B8A8
---	R_RGA_CalcWorkBufferSize
R_RGA_CalcWorkBufferB_Size	R_RGA_CalcWorkBufferB_Size
---	(RGA C++ API)
r_co_function_t	---

*1 The specification was changed.

4.2.2 Error Code

If an error was raised, "R_ERROR_Set" function is called with not 0 argument as "errnum_t" type. Also, the function returning "errnum_t" type returns not 0.

The code raising error is found by searching the symbol in source files or set conditional break point of the argument of "errnum_t" type in "R_ERROR_Set" function.

You can search the number of error code in HTML file attached with the project.

4.2.3 Figure of byte_per_pixel_t

This is a type of number of bytes per pixel. See Table 1-2.

When one pixel is less than one byte (BitPerPixel < 8), a value shifted from the number of bits per pixel is set.

When 1 pixel is 1 byte or more:

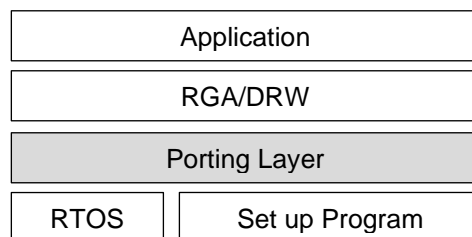
	15	8	7	0
0	0	Number of bytes		

When 1 pixel is less than 1 byte:

	15	8	7	0
0	Number of bits		0	

4.3 Porting Guide

If you change RTOS and set up program from RGA running environment, it is necessary to change the code of the porting layer.



The porting layer is defined in the following file.

- r_drw_pl.c

RGA calls the following porting layer functions. It is necessary to change the define of functions according to a kind of RTOS and set up program.

Kind	Function Name	Description
Initiali- zation	R_DRW_OnInitialize	Callback function called at initializing DRW
	R_DRW_OnFinalize	Callback function called at finalizing DRW
Interrupt	R_DRW_EnableInterrupt	Enables DRW interrupt
	R_DRW_DisableInterrupt	Disables DRW interrupt
RTOS thread	R_DRW_WaitForInterruptEvent	Waits for DRW interrupt
	R_DRW_SetInterruptEvent	Notifies from DRW interrupt to the thread
	R_DRW_CheckThread	Checks if the current thread is locked thread
Memory	R_DRW_FlushCache	Flushes level 1 CPU cache
	R_DRW_ToPhysicalAddress	Changes to physical address
	R_DRW_ToCachedAddress	Checks if the address is in the L1 cache area or changes to L1 cache area
	R_DRW_ToUncachedAddress	Checks if the address is in L1 uncached area or changes to L1 uncached area

The porting layer must call the following API functions.

Function Name	Description
R_DRW_OnInterrupting	Execute processing when DRW interrupt was raised

For details of the function, refer to the HTML file attached to the project. For diagrams not written in the HTML file, it is described in this manual.

4.4 Supplementary Explanation

4.4.1 Correspondence to Canvas 2D and Correspondence to Hardware Acceleration

In the hardware column, ✓ : Hardware is used, x: No hardware is used, —: Not applicable

Canvas2D API	RGA - C language API	Hardware
CanvasRenderingContext2D interface.	graphics_t	—
getContext()	R_GRAPHICS_Initialize	—
context.canvas	-	
CSS currentColor	-	
.save ()	R_GRAPHICS_Save	—
.restore()	R_GRAPHICS_Restore	—
.scale(x, y)	R_GRAPHICS_ScaleMatrix	✓
.rotate(angle)	R_GRAPHICS_RotateMatrixRadi an	✓
.translate(x, y)	R_GRAPHICS_TranslateMatrix	✓
.transform(a, b, c, d, e, f)	R_GRAPHICS_TransformMatrix	✓
.setTransform(a, b, c, d, e, f)	R_GRAPHICS_SetMatrix_2x3	✓
.lineWidth	-	
.lineCap, .lineJoin, .miterLimit	-	
.moveTo()	-	
.closePath()	-	
.lineTo()	-	
.quadraticCurveTo()	-	
.bezierCurveTo()	-	
.arcTo()	-	
.arc()	-	
.rect()	R_GRAPHICS_Rect	✓
.fillStyle single-color	R_GRAPHICS_SetFillColor	✓
.fillStyle gradation	-	
.fillStyle pattern	R_GRAPHICS_SetFillPattern	✓
.strokeStyle single-color	-	
.strokeStyle gradation	-	
.strokeStyle pattern	-	
.createLinearGradient()	-	
.createRadialGradient()	-	
CanvasGradient.addColorSto p()	-	
.createPattern()	R_GRAPHICS_PATTERN_Initiali ze	—
.beginPath()	R_GRAPHICS_BeginPath	—

.fill()	-	
.stroke()	-	
.drawSystemFocusRing()	-	
.drawCustomFocusRing()	-	
.scrollPathIntoView()	-	
.clip()	R_GRAPHICS_Clip (Only a single rectangle)	✓
.isPointInPath()	-	
.clearRect()	R_GRAPHICS_Clear	✓
.fillRect()	R_GRAPHICS_FillRect	✓
.strokeRect()	-	
	-	
.drawImage(dx, dy)	R_GRAPHICS_DrawImage	✓
.drawImage(dx, dy, dw, dh)	R_GRAPHICS_DrawImageResized	✓
.drawImage(sx, sy, sw, sh, dx, dy, dw, dh)	R_GRAPHICS_DrawImageChild	✓
.createImageData(Width, Height)	R_GRAPHICS_IMAGE_- InitR8G8B8A8	—
.createImageData(ImageData)	R_GRAPHICS_IMAGE_- InitSameSizeR8G8B8A8	—
ImageData.width	R_GRAPHICS_IMAGE_GetProperties	—
ImageData.height	R_GRAPHICS_IMAGE_GetProperties	—
ImageData.data	R_GRAPHICS_IMAGE_GetProperties	—
.getImageData()	R_GRAPHICS_IMAGE_- InitCopyFrameBufferR8G8B8A8	x
.putImageData()	R_GRAPHICS_DrawImage R_GRAPHICS_DrawImageChild	x
.globalAlpha	R_GRAPHICS_SetGlobalAlpha	✓
.globalCompositeOperation	R_GRAPHICS_- SetGlobalCompositeOperation	✓

4.4.2 Internal operation in initialize function

The function call tree related to RGA initialization is shown.

```
R_GRAPHICS_Initialize
  R_GRAPHICS_OnInitialize
    R_GRAPHICS_H_Initialize // Initialize the hardware
      R_DRW_OnInitialize
    R_GRAPHICS_OnInitialized
```

4.4.3 Identifying Image Format

The first few bytes of the `graphics_image_t`-type structure to be specified for the argument of the `R_GRAPHICS_DrawImage`, `R_GRAPHICS_DrawImageResized`, or `R_GRAPHICS_DrawImageChild` function are used to identify image formats. Therefore, JPEG file data or PNG file data can be specified directly for the `graphics_image_t`-type argument of the `R_GRAPHICS_DrawImage` function.

Image Format	First Few Bytes	Note
JPEG	0xFF 0xD8	SOI segment
PNG	0x89 0x50 0x4E 0x47	PNG header
Raw	Others	<code>graphics_image_t</code> -type structure + Raw data

4.4.4 Defaultable Flags

Defaultable flags are the type that can select three options: setting to ON, setting to OFF, and no setting (unchanged) for each array element when setting logic-type array variables. Symbols for setting to ON and setting to OFF are defined. When these symbols are not used, settings remain unchanged.

The variable type that stores defaultable flags is a 32-bit integer type. These flags are defined as a flag that sets the lower 16 bits to ON and a flag that sets the upper 16 bits to OFF. Components of the upper 16 bits are the same as those of the lower 16 bits.

Flag to be set to OFF (upper 16 bits)	Flag to be set to ON (lower 16 bits)
--	---

```
typedef BitField DefaultableFlagsType; /* Flags of DefaultableFlagType */
enum DefaultableFlagType {
    /* Set to "ON" */
    ENABLE_SAMPLE_FLAG_A    = 0x0001,
    ENABLE_SAMPLE_FLAG_B    = 0x0002,
    ENABLE_SAMPLE_FLAG_C    = 0x0004,

    /* Set to "OFF" */
    DISABLE_SAMPLE_FLAG_A   = ENABLE_SAMPLE_FLAG_A << 16,
    DISABLE_SAMPLE_FLAG_B   = ENABLE_SAMPLE_FLAG_B << 16,
    DISABLE_SAMPLE_FLAG_C   = ENABLE_SAMPLE_FLAG_C << 16,
};
```

When the argument of the function to set flags (SampleClass_setDefaultableFlags below) is a defaultable flag, only flags to be set to "ON" or "OFF" are connected by "|". Settings remain unchanged for flags that are not set to "ON" or "OFF." For initialization functions, default values are used.

```
errnum_t main()
{
    DefaultableFlagsType flags;

    e= SampleClass_setDefaultableFlags( object,
        ENABLE_SAMPLE_FLAG_A | DISABLE_SAMPLE_FLAG_B ); IF(e)goto fin;
    /* SAMPLE_FLAG_C is not modified. */

    e= SampleClass_getDefaultableFlags( object, &flags ); IF(e)goto fin;
    if ( flags & ENABLE_SAMPLE_FLAG_A ) { ... }
    if ( flags & DISABLE_SAMPLE_FLAG_B ) { ... }
    if ( flags & ENABLE_SAMPLE_FLAG_C ) { ... }
}

errnum_t SampleClass_setDefaultableFlags( SampleClass* self,
DefaultableFlagsType Flags )
{
    self->Flags = self->Flags | ( Flags & 0x0000FFFF );
    self->Flags = self->Flags & ~( Flags >> 16 );
}

errnum_t SampleClass_getDefaultableFlags( SampleClass* self,
DefaultableFlagsType* out_Flags )
{
    BitField flags = ( self->Flags & 0x0000FFFF );
    *out_Flags = flags | ~( flags << 16 );
}
```

When the argument of the function to acquire the current flag (SampleClass_getDefaultableFlags above) is a defaultable flag, the upper 16 bits of an acquirable value are an inverted value of the lower 16 bits. For this reason, both symbols for setting to ON and symbols for setting to OFF can be used for decision statements. Since the internal variable that retains the current flag is the internal specification, there is no problem with specifications in which upper 16 bits are disabled.

Inverted value of lower 16 bits (upper 16 bits)	Current flag (lower 16 bits)
--	---------------------------------

4.4.5 Function to Initialize Internal Variables with Constants (*_initConst Function)

In the C language class that includes the finalizing function (*_finalize function), the *_initConst function must be called first (before calling the *_initialize function). This is to prevent any exception from occurring even if the *_finalize function is called when an error occurs from another object before initialization. Before calling the function that may cause an error first among functions, call the *_initConst function first at a time for all objects that only exist in functions (i.e. created and deleted in functions).

Even if the *_initConst function is called, many functions (methods) are not made available until the *_initialize function is called.

In the case of the C++ language API of this library, the *_initConst function responds to the constructor. The *_initialize function responds to object creation functions (such as R_RGA_New_Canvas2D_ContextClass function). The *_finalize function responds to the destroy member function. For example, when an error occurs before an object creation function is called after the constructor was called by the variable declaration, no exception occurs even when the destroy member function is called.

4.4.6 Finalize (*_Finalize Function)

The function that last name of is "_Finalize" does the operation of finalize. Finalize is close operation of file, the operation in the destructor of C++ language, the operation called from finally block of Java language and others.

Outline	Does finalize operation.	
Declaration	errnum_t *_Finalize(type* self, errnum_t e); type* self Target object to finalize	
Argument	errnum_t e	Error code raised from before. There is different whether to do the rollback operation whether error was raised depending on a function's specification.
Return value	Error code or e 0 = successful and e = 0 If argument e is not 0, returns the value of e. If argument e is 0, returns an error code of this finalize operation.	

Pass error code raised from before to argument "e" like the following code.

```
errnum_t  Func()
{
    errnum_t  e;

    CLASS_InitConst( &sample );
    e= CLASS_Initialize( &sample ); IF(e){goto fin;}

    e=0;
fin:
    e= CLASS_Finalize( &sample, e );
    return  e;
}
```

If the finalize operation was success, the target object becomes Uninitialized state.

If error was raised in the finalize operation, the function returns error code not 0 and may become the following state.

State after finalizing	Description and expected behavior
Uninitialized	<p>The target object can be deleted.</p> <p>In this case, all internal objects are finalized, even if errors were raised internally and the program can be resumed from error state.</p> <p>Uninitialized internal objects are deleted.</p> <p>Internal objects which returned to a previous state is not deleted until resource management object deletes it, or the system was reset. Otherwise, the object may keep lock state. A callback function related internal object which returned to a previous state may be called.</p> <p>Notify the error information to the end user or make logs after finalize function.</p>
Reset	<p>Calls "R_OSPL_RaiseUnrecoverable" function from the finalize function.</p> <p>"R_OSPL_RaiseUnrecoverable" function resets the system or exits the process.</p> <p>In this case, finalize function is not returned.</p>
State before finalizing	<p>The object which returned to previous state cannot be deleted.</p> <p>Call finalize operation which usually returns to previous state from out of finally block (fin: in above code) and retry the finalize operation.</p> <p>If the finalize operation is called from finally block, the object will not be deleted.</p> <p>Adjust anything before calling the finalize function.</p> <p>An object which integrates not deleted object sometimes becomes to Uninitialized state. In this case, see the description of "Uninitialized" in this table.</p>

5. Functions and Methods

5.1 Functions Equivalent to graphics_t Class Member Function

5.1.1 List of Functions

Section	Function Name	Description
5.1.2	R_GRAPHICS_STATIC_GetVersion	Gets version number of RGA
5.1.3	R_GRAPHICS_PrintRegisters	Prints registers
5.1.4	R_GRAPHICS_Start	Starts buffered graphics hardware rendering.
5.1.5	R_GRAPHICS_FinishPreviousFrame	GPU finishes drawing to "frame_buffer_t::draw_buffer_index".
5.1.6	R_GRAPHICS_StartPreviousFrame	Starts hardware rendering about previous frame
5.1.7	R_GRAPHICS_GetHasFramePipeline	Returns on/off mode of frame pipeline mode
5.1.8	R_GRAPHICS_BeginSoftwareRendering2	Notifies the graphics library of the start of software rendering
5.1.9	R_GRAPHICS_BeginSoftwareRenderingA	Notifies the graphics library of the start of software rendering
5.1.10	R_GRAPHICS_CONFIG_SetEmpty	Initializes all member variables to the value meaning empty
5.1.11	R_GRAPHICS_InitConst	Initializes the constant part.
5.1.12	R_GRAPHICS_Initialize	Initializes the graphics drawing context object.
5.1.13	R_GRAPHICS_Finalize	Finalizes the graphics drawing context object.
5.1.14	R_GRAPHICS_SetFrameBuffer	Changes the drawing target.
5.1.15	R_GRAPHICS_GetFrameBuffer	Gets the drawing target.
5.1.16	R_GRAPHICS_Finish	Waits until drawing is completed.
5.1.17	R_GRAPHICS_Save	Saves the set value of context to the specified status structure.
5.1.18	R_GRAPHICS_Restore	Returns the status structure content to context.
5.1.19	R_GRAPHICS_ResetMatrix	Resets the target matrix of the matrix calculation function to the unit matrix.
5.1.20	R_GRAPHICS_SetMatrix_2x3	Sets each element of the matrix. (2x3)
5.1.21	R_GRAPHICS_SetMatrix_3x3	Sets each element of the matrix. (3x3)
5.1.22	R_GRAPHICS_GetMatrix_3x3	Acquires each element of the matrix.
5.1.23	R_GRAPHICS_TranslateMatrixI	Translates matrix from the current matrix. (integer type specified)
5.1.24	R_GRAPHICS_TranslateMatrix	Translate matrix from the current matrix. (floating-point type specified)
5.1.25	R_GRAPHICS_ScaleMatrix	Enlarges or reduces matrix from the current matrix.
5.1.26	R_GRAPHICS_RotateMatrixDegree	Rotates matrix from the current matrix. Rotation center: (0,0)
5.1.27	R_GRAPHICS_ShearMatrix	Makes shear deformation from the current matrix.
5.1.28	R_GRAPHICS_TransformMatrix	Multiplies the current matrix by the specified 2x3 matrix.
5.1.29	R_GRAPHICS_MultiplyMatrix	Multiplies the current matrix by the specified 3x3 matrix.
5.1.30	R_GRAPHICS_GetProjectiveMatrix	Acquires a matrix that deforms a random profile quadrangle to a random profile quadrangle.
5.1.31	R_GRAPHICS_SetBackgroundColor	Sets the background color.
5.1.32	R_GRAPHICS_GetBackgroundColor	Acquires the background color.
5.1.33	R_GRAPHICS_GetClearColor	Acquires the color used for R_GRAPHICS_Clear.
5.1.34	R_GRAPHICS_Clear	Clears rectangle area.
5.1.35	R_GRAPHICS_DrawImage	Draws an image.
5.1.36	R_GRAPHICS_DrawImageResized	Enlarges or reduces an image and draws it in a rectangle.

5.1.37	R_GRAPHICS_DrawImageChild	Draws a part of an image.
5.1.38	R_GRAPHICS_FillRect	Fills a rectangle.
5.1.39	R_GRAPHICS_SetFillColor	Sets the color used for single-color fill for the current fill paint object.
5.1.40	R_GRAPHICS_SetFillPattern	Sets a pattern for the current fill paint object.
5.1.41	R_GRAPHICS_BeginPath	Resets the default path content to null.
5.1.42	R_GRAPHICS_Rect	Adds a rectangle to the default path.
5.1.43	R_GRAPHICS_Clip	Sets the shape of the current default path to a clipping area.
5.1.44	R_GRAPHICS_SetGlobalAlpha	Sets an alpha value (opacity) to be multiplied by all drawings.
5.1.45	R_GRAPHICS_GetGlobalAlpha	Acquires an alpha value (opacity) to be multiplied by all drawings.
5.1.46	R_GRAPHICS_SetGlobalCompositeOperation	Sets the calculation method for alpha blend.
5.1.47	R_GRAPHICS_GetGlobalCompositeOperation	Acquires the calculation method for alpha blend.
5.1.48	R_GRAPHICS_SetQualityFlags	Sets the drawing quality.
5.1.49	R_GRAPHICS_GetQualityFlags	Acquires the current drawing quality.
5.1.50	R_GRAPHICS_SetStrokeColor	Sets the color used for single-color fill for the paint object of the current border.
5.1.51	R_GRAPHICS_StrokeRect	Draws sides of rectangle.
5.1.52	R_GRAPHICS_BeginSoftwareRendering	Notifies the graphics library of the start of software rendering.
5.1.53	R_GRAPHICS_EndSoftwareRendering	Notifies the graphics library of the end of software rendering.
5.1.54	R_GRAPHICS_EndRenderingInFin	Calls this function from the end of the function that performs software rendering.

5.1.2 R_GRAPHICS_STATIC_GetVersion

Outline	Gets version number of RGA.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_STATIC_GetVersion(uint32_t* out_Version);	
Description	If "out_Version == 310", version number is 3.10.	
Argument	out_Version	Output: version number of RGA
Return value	Error code. If there is no error, the return value is 0.	

5.1.3 R_GRAPHICS_PrintRegisters

Outline	Prints registers	
Header	RGA.h	
Declaration	void R_GRAPHICS_PrintRegisters(void);	
Description		
Argument	None	
Return value	None	

5.1.4 R_GRAPHICS_Start

Outline	Starts buffered graphics hardware rendering.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Start(graphics_t* self);	
Description	This function can be called, if frame pipeline mode has been off. The operation by CPU and graphics hardware run parallely after calling this function.	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.5 R_GRAPHICS_FinishPreviousFrame

Outline	GPU finishes drawing to "frame_buffer_t::draw_buffer_index".	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_FinishPreviousFrame(graphics_t* self);	
Description	This function can be called, if <graphics_config_t.has_frame_pipeline> = true. This function must be called before swapping frame buffers. Call <R_GRAPHICS_StartPreviousFrame> to start drawing next frame after showing previous frame completed to drawing by this function to the display.	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.6 R_GRAPHICS_StartPreviousFrame

Outline	Starts hardware rendering about previous frame	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_StartPreviousFrame(graphics_t* self);	
Description	This function can be called, if <graphics_config_t.has_frame_pipeline> = true. This draws the frame buffer of <frame_buffer_t.draw_buffer_index>. This function must be called after setting to draw next frame by <R_GRAPHICS_SetFrameBuffer> function and swapping frame buffers.	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.7 R_GRAPHICS_GetHasFramePipeline

Outline	Returns on/off mode of frame pipeline mode	
Header	RGA.h	
Declaration	bool_t R_GRAPHICS_GetHasFramePipeline(graphics_t* self);	
Description	To set on/off mode of frame pipeline mode, set to "graphics_config_t::has_frame_pipeline".	
Argument	graphics_t* self	Graphics drawing context
Return value	on/off mode of frame pipeline mode	

5.1.8 R_GRAPHICS_BeginSoftwareRendering2

Outline	Notifies the graphics library of the start of software rendering	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_BeginSoftwareRendering2(graphics_t* self);	
Description	This function is for portability.	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.9 R_GRAPHICS_BeginSoftwareRenderingA

Outline	Notifies the graphics library of the start of software rendering	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_BeginSoftwareRenderingA(graphics_t* self, void* in_Address);	
Description	This function is for portability.	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.10 R_GRAPHICS_CONFIG_SetEmpty

Outline	Initializes all member variables to the value meaning empty	
Header	RGA.h	
Declaration	void R_GRAPHICS_CONFIG_SetEmpty(graphics_config_t* out);	
Description	See <Section_ValueMeaningEmptyAndFlaggedParameter> about the value meaning empty.	
Argument	graphics_config_t* out	A structure initializing the member variable
Return value	None	

5.1.11 R_GRAPHICS_InitConst

Outline	Initializes the constant part used by the RGA.	
Header	RGA.h	
Declaration	void R_GRAPHICS_InitConst(graphics_t* self);	
Description	Refer to section 6.11.9, Function to Initialize Internal Variables with Constants (*_initConst Function).	
Argument	graphics_t* self	Graphics drawing context
Return value	None	

5.1.12 R_GRAPHICS_Initialize

Outline	Initializes the graphics drawing context object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Initialize (graphics_t* self, graphics_config_t* config);	
Description	<p>Initializes internal variables.</p> <p>Initializes Renesas Graphics Processor for OpenVG™ (R-GPVG2) and JPEG Codec Unit (JCU).</p> <p>For the built-in version, register the interrupt handler before calling this function.</p> <p>When "self" is no longer be used, call R_GRAPHICS_Finalize.</p> <p>There is only one context. When two or more frame buffers was drawn, change the frame buffer by "R_GRAPHICS_SetFrameBuffer" function.</p> <p>See section 4.4.2. Internal operation in initialize function.</p>	
Argument	graphics_t* self	Graphics drawing context
	graphics_config_t* config	
Return value	Error code. If there is no error, the return value is 0.	

5.1.13 R_GRAPHICS_Finalize

Outline	Finalizes the graphics drawing context object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Finalize(graphics_t* self, errnum_t e);	
Description	Internal object is deleted, even if error was raised.	
Argument	graphics_t* self	Graphics drawing context
	errnum_t e	Errors that have occurred. No error = 0.
Return value	Error code or e. 0 = "successful and e = 0".	

5.1.14 R_GRAPHICS_SetFrameBuffer

Outline	Changes the drawing target.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetFrameBuffer(graphics_t* self, frame_buffer_t* frame_buffer);	
Description	<p>This changes the drawing target to VRAM area at "frame_buffer->buffer_address [frame_buffer->draw_bufferIndex]".</p> <p>This function does the same operation as "R_GRAPHICS_Finish" function, too.</p>	
Argument	graphics_t* self	Graphics drawing context
	frame_buffer_t* frame_buffer	Frame buffer to be drawn
Return value	Error code. If there is no error, the return value is 0.	

5.1.15 R_GRAPHICS_GetFrameBuffer

Outline Gets the drawing target.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_GetFrameBuffer(graphics_t* self, frame_buffer_t** out_frame_buffer);`

Description

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>frame_buffer_t** out_frame_buffer</code>	(Output) Frame buffer to be drawn
Return value	Error code. If there is no error, the return value is 0.	

5.1.16 R_GRAPHICS_Finish

Outline Waits until drawing is completed.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_Finish(graphics_t* self);`
 Description This waits for finish drawing by RGA or by the application directly to the frame buffer and changes to the state to able to show the frame buffer.
 When the CPU directly reads or writes frame buffer data in the cache area or non-cache area after the RGA API is called, enclose by calling `R_GRAPHICS_BeginSoftwareRendering` to `R_GRAPHICS_EndSoftwareRendering`. These functions wait hardware rendering and flush the CPU cache only if necessary. If the process waits for finish drawing asynchronously, call "R_GRAPHICS_FinishStart" function.

Argument	<code>graphics_t* self</code>	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.17 R_GRAPHICS_Save

Outline Saves the set value of context to the specified status structure.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_Save(graphics_t* self, graphics_status_t* out_status);`
 Description Example:

```

graphics_status_t status = {0};
e= R_GRAPHICS_Save( &graphics, &status ); IF(e){goto fin;}
:
fin:
e= R_GRAPHICS_Restore( &graphics, &status, e );

```

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>graphics_status_t* out_status</code>	(Output) Set value of context
Return value	Error code. If there is no error, the return value is 0.	

5.1.18 R_GRAPHICS_Restore

Outline	Returns the status structure content to context.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Restore(graphics_t* self, graphics_status_t* status, errnum_t e);	
Description		
Argument	graphics_t* self	Graphics drawing context
	graphics_status_t* status	Set value of context
	errnum_t e	Errors that have occurred. No error = 0.
Return value	Error code or e 0 = successful and e = 0	

5.1.19 R_GRAPHICS_ResetMatrix

Outline	Resets the matrix to be a matrix calculation function target to the unit matrix.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_ResetMatrix(graphics_t* self);	
Description		
Argument	graphics_t* self	Graphics drawing context
	Error code. If there is no error, the return value is 0.	

5.1.20 R_GRAPHICS_SetMatrix_2x3

Outline	Sets each element of the current matrix. (2x3)	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetMatrix_2x3(graphics_t* self, graphics_matrix_float_t sx, graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty);	
Description	When the library was ported, take care computing error.	
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t sx, graphics_matrix_float_t ky, graphics_matrix_float_t kx, graphics_matrix_float_t sy, graphics_matrix_float_t tx, graphics_matrix_float_t ty	2x3 matrix $\begin{pmatrix} sx & kx & tx \\ ky & sy & ty \\ 0 & 0 & 1 \end{pmatrix}$
Return value	Error code. If there is no error, the return value is 0.	

5.1.21 R_GRAPHICS_SetMatrix_3x3

Outline	Sets each element of the current matrix (Matrix[]) (3x3).	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetMatrix_3x3(graphics_t* self, graphics_matrix_float_t * matrix);	
Description	When the library was ported, take care computing error.	
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t * matrix	3x3 matrix (array) $\begin{pmatrix} Matrix[0] & Matrix[3] & Matrix[6] \\ Matrix[1] & Matrix[4] & Matrix[7] \\ Matrix[2] & Matrix[5] & Matrix[8] \end{pmatrix}$
Return value	Error code. If there is no error, the return value is 0.	

5.1.22 R_GRAPHICS_GetMatrix_3x3

Outline	Acquires each element of the current matrix (Matrix[]).	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_GetMatrix_3x3(graphics_t* self, graphics_matrix_float_t * out_matrix);	
Description		
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t * out_matrix	(Output) 3x3 matrix (array) $\begin{pmatrix} \text{Matrix}[0] & \text{Matrix}[3] & \text{Matrix}[6] \\ \text{Matrix}[1] & \text{Matrix}[4] & \text{Matrix}[7] \\ \text{Matrix}[2] & \text{Matrix}[5] & \text{Matrix}[8] \end{pmatrix}$
Return value	Error code. If there is no error, the return value is 0.	

5.1.23 R_GRAPHICS_TranslateMatrixI

Outline	Translates the current matrix (M). (Integer type specified)	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_TranslateMatrixI(graphics_t* self, int_t tx, int_t ty);	
Description	$M=M \cdot \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$	
Argument	graphics_t* self	Graphics drawing context
	int_t tx, int_t ty	Displacement (When the origin is at the upper left, plus of X means right direction and plus of Y means downward direction.)
Return value	Error code. If there is no error, the return value is 0.	

5.1.24 R_GRAPHICS_TranslateMatrix

Outline	Translates the current matrix (M). (Floating-point type specified)	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_TranslateMatrix(graphics_t* self, graphics_matrix_float_t tx, graphics_matrix_float_t ty);	
Description	$M=M \cdot \begin{pmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{pmatrix}$	
Argument	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t tx graphics_matrix_float_t ty	Displacement (When the origin is at the upper left, plus of X means right direction and plus of Y means downward direction.)
Return value	Error code. If there is no error, the return value is 0.	

5.1.25 R_GRAPHICS_ScaleMatrix

Outline Enlarges or reduces the current matrix (M).
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_ScaleMatrix(graphics_t* self,
 graphics_matrix_float_t sx, graphics_matrix_float_t sy);`

Description

$$M = M \cdot \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

When the library was ported, take care computing error.

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>graphics_matrix_float_t tx</code> <code>graphics_matrix_float_t ty</code>	Magnification (Enlargement/reduction center: Origin)
Return value	Error code. If there is no error, the return value is 0.	

5.1.26 R_GRAPHICS_RotateMatrixDegree

Outline Rotates the current matrix (M). The center coordinates of rotation are (0,0).
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_RotateMatrixDegree(graphics_t* self,
 graphics_matrix_float_t angle);`

Description

$$M = M \cdot \begin{pmatrix} \cos(\text{angle}) & -\sin(\text{angle}) & 0 \\ \sin(\text{angle}) & \cos(\text{angle}) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

When the library was ported, take care computing error.

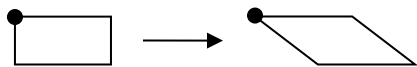
Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>graphics_matrix_float_t angle</code>	Rotation angle (degrees) (When the origin is at the upper left, plus means clockwise direction.)
Return value	Error code. If there is no error, the return value is 0.	

5.1.27 R_GRAPHICS_ShearMatrix

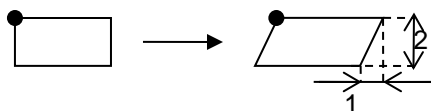
Outline	Makes shear deformation of the current matrix (M).
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_ShearMatrix(graphics_t* self, graphics_matrix_float_t shx, graphics_matrix_float_t shy);
Description	$M=M \cdot \begin{pmatrix} 1 & shx & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

When (shx, shy) = (1.0, 0.0), a parallelogram is generated with perpendicular sides tilted 45 degrees.

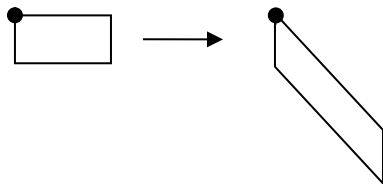
Note, however, that the matrix is shifted unless the origin is at the upper left of the rectangle.



When (shx, shy) = (-0.5, 0.0), a parallelogram is generated with hypotenuses of a triangle (base : height = 1 : 2).



When (shx, shy) = (0.0, 1.0), a parallelogram is generated with horizontal sides tilted 45 degrees.



Argument	When the library was ported, take care computing error.	
	graphics_t* self	Graphics drawing context
	graphics_matrix_float_t shx graphics_matrix_float_t shy	Rate of shear (Shear center: Origin)
Return value	Error code. If there is no error, the return value is 0.	

5.1.28 R_GRAPHICS_TransformMatrix

Outline Multiplies the current matrix (M) by the specified 2x3 matrix.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_TransformMatrix(graphics_t* self,
 graphics_matrix_float_t sx, graphics_matrix_float_t ky,
 graphics_matrix_float_t kx, graphics_matrix_float_t sy,
 graphics_matrix_float_t tx, graphics_matrix_float_t ty);`

Description

$$M=M \cdot \begin{pmatrix} sx & kx & tx \\ ky & sy & ty \\ 0 & 0 & 1 \end{pmatrix}$$

When the library was ported, take care computing error.

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>graphics_matrix_float_t sx</code> <code>graphics_matrix_float_t ky</code> <code>graphics_matrix_float_t kx</code> <code>graphics_matrix_float_t sy</code> <code>graphics_matrix_float_t tx</code> <code>graphics_matrix_float_t ty</code>	2x3 matrix to be multiplied
Return value	Error code. If there is no error, the return value is 0.	

5.1.29 R_GRAPHICS_MultiplyMatrix

Outline Multiplies the current matrix (M) by the specified 3x3 matrix (Matrix[]).
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_MultiplyMatrix(graphics_t* self, graphics_matrix_float_t *
 matrix);`

Description

$$M=M \cdot \begin{pmatrix} Matrix[0] & Matrix[3] & Matrix[6] \\ Matrix[1] & Matrix[4] & Matrix[7] \\ Matrix[2] & Matrix[5] & Matrix[8] \end{pmatrix}$$

When the library was ported, take care computing error.

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>graphics_matrix_float_t * matrix</code>	3x3 matrix to be multiplied (array with nine elements)
Return value	Error code. If there is no error, the return value is 0.	

5.1.30 R_GRAPHICS_GetProjectiveMatrix

Outline	Acquires a matrix that deforms a random profile quadrangle to a random profile quadrangle.	
Header	RGA.h	
Declaration	<pre>errnum_t R_GRAPHICS_GetProjectiveMatrix(graphics_matrix_float_t source_top_left_x, graphics_matrix_float_t source_top_left_y, graphics_matrix_float_t source_top_right_x, graphics_matrix_float_t source_top_right_y, graphics_matrix_float_t source_bottom_left_x, graphics_matrix_float_t source_bottom_left_y, graphics_matrix_float_t source_bottom_right_x, graphics_matrix_float_t source_bottom_right_y, graphics_matrix_float_t destination_top_left_x, graphics_matrix_float_t destination_top_left_y, graphics_matrix_float_t destination_top_right_x, graphics_matrix_float_t destination_top_right_y, graphics_matrix_float_t destination_bottom_left_x, graphics_matrix_float_t destination_bottom_left_y, graphics_matrix_float_t destination_bottom_right_x, graphics_matrix_float_t destination_bottom_right_y, graphics_matrix_float_t * out_matrix);</pre>	
Description	When the library was ported, take care computing error.	
Argument	graphics_matrix_float_t source*	Four-point coordinates before conversion
	graphics_matrix_float_t destination*	Four-point coordinates after conversion
	graphics_matrix_float_t * out_matrix	(Output) 3x3 matrix (array with nine elements)
Return value	Error code. If there is no error, the return value is 0.	

5.1.31 R_GRAPHICS_SetBackgroundColor

Outline	Sets the background color.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetBackgroundColor(graphics_t* self, r8g8b8a8_t color);	
Description	<p>When a frame buffer without alpha is to be drawn, the background color is the same as the clear color. The default background color is white.</p> <p>When a frame buffer with alpha is to be drawn, the background color is specified color and the clear color drawing to the target is transparent black even if any color specified. Set the background color for the back layer for correct look.</p>	
Argument	graphics_t* self	Graphics drawing context
	r8g8b8a8_t color	Use R_RGA_Get_R8G8B8A8() to acquire the background color.
Return value	Error code. If there is no error, the return value is 0.	

5.1.32 R_GRAPHICS_GetBackgroundColor

Outline Acquires the background color.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_GetBackgroundColor(graphics_t* self, r8g8b8a8_t* out_color);`

Description

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>r8g8b8a8_t* out_color</code>	(Output) Background color
Return value	Error code. If there is no error, the return value is 0.	

5.1.33 R_GRAPHICS_GetClearColor

Outline Acquires the color used for `R_GRAPHICS_Clear`.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_GetClearColor(graphics_t* self, r8g8b8a8_t* out_color);`
 Description Use `R_GRAPHICS_SetBackgroundColor()` to set the clear color.

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>r8g8b8a8_t* out_color</code>	(Output) Clear color
Return value	Error code. If there is no error, the return value is 0.	

5.1.34 R_GRAPHICS_Clear

Outline Clears the rectangle area.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_Clear(graphics_t* self, int_fast32_t min_x, int_fast32_t min_y, int_fast32_t width, int_fast32_t height);`
 Description Use `R_GRAPHICS_SetBackgroundColor()` to set the clear color.
 When a double-buffer is used, the drawing buffer is cleared. Therefore, the cleared content is not displayed only by calling this function. Use `R_WINDOW_SURFACES_SwapBuffers()` to apply display.
 This function is affected by clipping and matrix.

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>int_fast32_t min_x,</code> <code>int_fast32_t min_y,</code> <code>int_fast32_t width,</code> <code>int_fast32_t height</code>	Rectangle area
Return value	Error code. If there is no error, the return value is 0.	

5.1.35 R_GRAPHICS_DrawImage

Outline	Draws an image at coordinates (min_x, min_y) with the same size.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_DrawImage(graphics_t* self, graphics_image_t* image, int_fast32_t min_x, int_fast32_t min_y);
Description	Image data to be specified for the image argument can be created by using section 7.1, Image Format Conversion by ImagePackager. JPEG data can be directly specified for arguments.

This function is affected by R_GRAPHICS_SetGlobalAlpha.

This function is also affected by the current matrix and clipping.

When the alpha component is included in the image and is not included in the drawing target frame buffer, RGB components to be drawn in the frame buffer are blended to values that have been multiplied by the alpha component. When the alpha component is included in the drawing target frame buffer, RGB components are blended to values that have not been multiplied by the alpha component.

An example of pixel format including the alpha component:

ARGB8888, ARGB4444

An example of pixel format without alpha component:

XRGB8888, RGB565

When drawing characters of raster font, specify the image of A8 (Alpha only) format to "image" argument. Also, the image of A8 format can be zoomed.

Specify a CLUT-format image as same bit count as the frame buffer for drawing in the CLUT format frame buffer. Only min_x = 0 and min_y = 0 can be specified as a drawing position. If source image's width was not byte unit, an error is raised. If CLUT color in the video controller was fit with drawing image, set CLUT of graphics_image_properties_t to the video controller.

when the image data specified for the image argument exists in the array variables prepared by the application, flushing is required before calling "R_GRAPHICS_DrawImage" function. However, flushing is not required, if there is the image data in ROM.

When performing flush, directly flush the CPU cache or enclose the image data read/write processing by R_GRAPHICS_BeginSoftwareRendering to R_GRAPHICS_EndSoftwareRendering.

Argument	graphics_t* self	Graphics drawing context
	graphics_image_t* image	Image
	int_fast32_t min_x, int_fast32_t min_y	Minimum X and Y coordinates
Return value	Error code. If there is no error, the return value is 0.	

5.1.36 R_GRAPHICS_DrawImageResized

Outline	Enlarges or reduces an image and draws it in the specified rectangle.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_DrawImageResized(graphics_t* self, graphics_image_t* image, int_fast32_t min_x, int_fast32_t min_y, int_fast32_t width, int_fast32_t height);	
Description	See the description on the R_GRAPHICS_DrawImage function. See 5.1.35. JPEG data cannot be specified to image argument.	
Argument	graphics_t* self	Graphics drawing context
	graphics_image_t* image	Image
	int_fast32_t min_x, int_fast32_t min_y	Minimum X and Y coordinates
	int_fast32_t width, int_fast32_t height	Width and height of the drawing target
Return value	Error code. If there is no error, the return value is 0.	

5.1.37 R_GRAPHICS_DrawImageChild

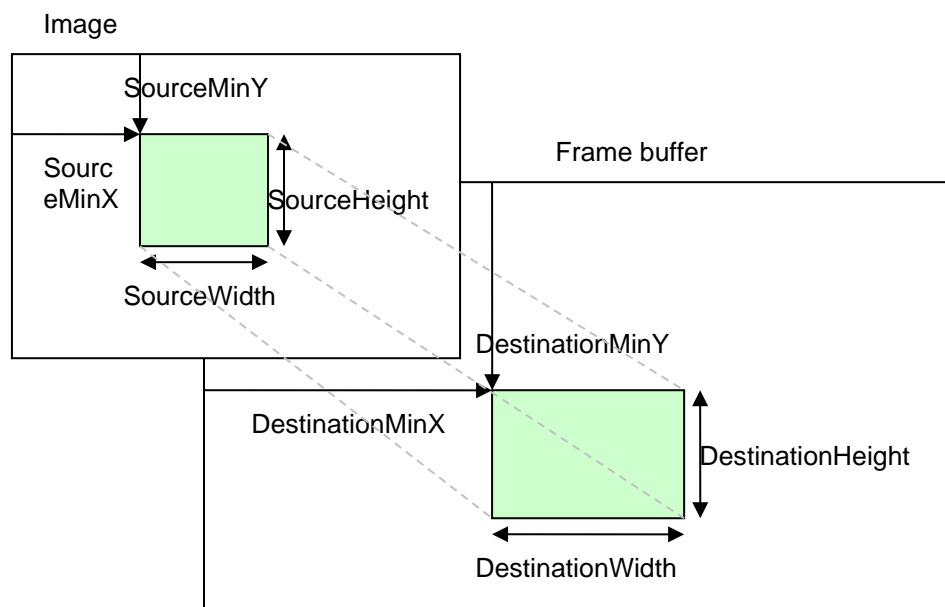
Outline Draws a part of an image.

Header RGA.h

Declaration

```
errnum_t R_GRAPHICS_DrawImageChild( graphics_t* self, graphics_image_t*
image,
int_fast32_t source_min_x, int_fast32_t source_min_y,
int_fast32_t source_width, int_fast32_t source_height,
int_fast32_t destination_min_x, int_fast32_t destination_min_y,
int_fast32_t destination_width, int_fast32_t destination_height );
```

Description



When $\text{source_width} \neq \text{destination_width}$ or $\text{source_height} \neq \text{destination_height}$, images are enlarged or reduced.

A part of image can be drawn with matrix of translate and zoom. But it cannot be drawn with another matrix.

See the description on the R_GRAPHICS_DrawImage function. See 5.1.35.

Argument

<code>graphics_t* self</code>	Graphics drawing context
<code>graphics_image_t* image</code>	Image
<code>int_fast32_t source_min_x</code> <code>int_fast32_t source_min_y</code>	Minimum X and Y coordinates in the image
<code>int_fast32_t source_width</code> <code>int_fast32_t source_height</code>	Width and height in the image
<code>int_fast32_t destination_min_x</code> <code>int_fast32_t destination_min_y</code>	Minimum X and Y coordinates of the drawing target
<code>int_fast32_t destination_width</code> <code>int_fast32_t destination_height</code>	Width and height of the drawing target
Return value	Error code. If there is no error, the return value is 0.

5.1.38 R_GRAPHICS_FillRect

Outline	Fills the rectangle area specified by the argument.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_FillRect(graphics_t* self, int_fast32_t min_x, int_fast32_t min_y, int_fast32_t width, int_fast32_t height);	
Description	This function is affected by the current matrix, the current fill, and clipping. No border is drawn. This function is affected by R_GRAPHICS_SetGlobalAlpha.	
Argument	graphics_t* self	Graphics drawing context
	int_fast32_t min_x, int_fast32_t min_y	Minimum X and Y coordinates of rectangle
	int_fast32_t width, int_fast32_t height	Width and height of rectangle
Return value	Error code. If there is no error, the return value is 0.	

5.1.39 R_GRAPHICS_SetFillColor

Outline	Changes the paint object of the current fill to single-color fill and sets the fill color.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetFillColor(graphics_t* self, r8g8b8a8_t Color);	
Description	The initial value is opaque black.	
Argument	graphics_t* self	Graphics drawing context
	r8g8b8a8_t color	Fill color. Use R_RGA_Get_R8G8B8A8() for the fill color setting.
Return value	Error code. If there is no error, the return value is 0.	

5.1.40 R_GRAPHICS_SetFillPattern

Outline	Sets the pattern for the current fill paint object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetFillPattern(graphics_t* self, graphics_pattern_t* pattern);	
Description	Use R_GRAPHICS_FillRect for drawing.	
Argument	graphics_t* self	Graphics drawing context
	graphics_pattern_t* pattern	Use R_GRAPHICS_PATTERN_Initialize() to initialize the pattern object.
Return value	Error code. If there is no error, the return value is 0.	

5.1.41 R_GRAPHICS_BeginPath

Outline	Resets the default path content to null.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_BeginPath(graphics_t* self);	
Description		
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.42 R_GRAPHICS_Rect

Outline	Adds a rectangle to the default path.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Rect(graphics_t* self, int_fast32_t min_x, int_fast32_t min_y, int_fast32_t width, int_fast32_t height);	
Description	This function is used to set the clipping area.	
Argument	graphics_t* self	Graphics drawing context
	int_fast32_t min_x, int_fast32_t min_y	Minimum X and Y coordinates of rectangle
	int_fast32_t width, int_fast32_t height	Width and height of rectangle
Return value	Error code. If there is no error, the return value is 0.	

5.1.43 R_GRAPHICS_Clip

Outline	Sets the shape of the current default path to a clipping area.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_Clip(graphics_t* self);	
Description	<p>When the current default path is empty or is not a rectangle, an error occurs.</p> <p>When the current default path is empty, drawing is disabled in any area.</p> <p>If this function is called when the current clipping area is a part of the frame buffer, the area common to the previous clipping area and the default path becomes a new clipping area.</p> <p>To restore entire drawing, call R_GRAPHICS_Restore.</p>	
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.44 R_GRAPHICS_SetGlobalAlpha

Outline	Sets an alpha value (opacity) to be multiplied by all drawings.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_SetGlobalAlpha(graphics_t* self, uint8_t alpha_value);	
Description	<p>The default value is 255.</p> <p>This function affects the following drawing functions.</p> <p>Figure fill functions such as R_GRAPHICS_FillRect</p> <p>Pattern drawing function R_GRAPHICS_FillRect</p> <p>Border drawing functions such as R_GRAPHICS_StrokeRect</p> <p>Image drawing functions such as R_GRAPHICS_DrawImage</p> <p>This function does not affect the following drawing function.</p> <p>R_GRAPHICS_Clear</p>	
Argument	graphics_t* self	Graphics drawing context
	uint8_t alpha_value	Alpha value (0 to 255). A smaller value makes drawing light.
Return value	Error code. If there is no error, the return value is 0.	

5.1.45 R_GRAPHICS_GetGlobalAlpha

Outline Acquires an alpha value (opacity) to be multiplied by all drawings.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_GetGlobalAlpha(graphics_t* self, uint8_t* out_alpha_value);`

Description

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>uint8_t* out_alpha_value</code>	(Output) Alpha value (0 to 255). A smaller value makes drawing light.
Return value	Error code. If there is no error, the return value is 0.	

5.1.46 R_GRAPHICS_SetGlobalCompositeOperation

Outline Sets the calculation method for alpha blend.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_SetGlobalCompositeOperation(graphics_t* self, graphics_composite_operation_t composite_operation);`
 Description The following case other than `GRAPHICS_SOURCE_OVER` can be set. Image drawing such as `R_GRAPHICS_DrawImage`

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>graphics_composite_operation_t composite_operation</code>	Calculation method
Return value	Error code. If there is no error, the return value is 0.	

5.1.47 R_GRAPHICS_GetGlobalCompositeOperation

Outline Acquires the calculation method for alpha blend.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_GetGlobalCompositeOperation(graphics_t* self, graphics_composite_operation_t* out_composite_operation);`

Description

Argument	<code>graphics_t* self</code>	Graphics drawing context
	<code>graphics_composite_operation_t* out_composite_operation</code>	(Output) Calculation method
Return value	Error code. If there is no error, the return value is 0.	

5.1.48 R_GRAPHICS_SetQualityFlags

Outline Sets the drawing quality.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_SetQualityFlags(graphics_t* self, graphics_quality_flags_t qualities);`

Description

Argument	<code>graphics_t* self,</code>	Graphics drawing context
	<code>graphics_quality_flags_t qualities</code>	
Return value	Error code. If there is no error, the return value is 0.	

5.1.49 R_GRAPHICS_GetQualityFlags

Outline Acquires the current drawing quality.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_GetQualityFlags(graphics_t* self,
 graphics_quality_flags_t* out_qualities);`

Description

Argument	<code>graphics_t* self,</code>	Graphics drawing context
	<code>graphics_quality_flags_t* out_qualities</code>	
Return value	Error code. If there is no error, the return value is 0.	

5.1.50 R_GRAPHICS_SetStrokeColor

Outline Sets the color used for single-color fill for the paint object of the current border.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_SetStrokeColor(graphics_t* self, r8g8b8a8_t color);`

Description

Argument	<code>graphics_t* self,</code>	Graphics drawing context
	<code>r8g8b8a8_t color</code>	Border color
Return value	Error code. If there is no error, the return value is 0.	

5.1.51 R_GRAPHICS_StrokeRect

Outline Draws sides of rectangle.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_StrokeRect(graphics_t* self,
 int_fast32_t min_x, int_fast32_t min_y, int_fast32_t width, int_fast32_t height);`
 Description This function is affected by line width, border color, and clipping.
 Fill is not made.
 When the current matrix is not the unit matrix, an error occurs.
 This function is affected by R_GRAPHICS_SetGlobalAlpha.

Argument	<code>graphics_t* self,</code>	Graphics drawing context
	<code>int_fast32_t min_x, int_fast32_t min_y</code>	Minimum X and Y coordinates of rectangle
	<code>int_fast32_t width, int_fast32_t height</code>	Width and height of rectangle
Return value	Error code. If there is no error, the return value is 0.	

5.1.52 R_GRAPHICS_BeginSoftwareRendering

Outline Notifies the graphics library of the start of software rendering.
 Header RGA.h
 Declaration `errnum_t R_GRAPHICS_BeginSoftwareRendering(graphics_t* self);`
 Description This function is used before software rendering to the frame buffer in On-Chip RAM.
 Depending on a condition, this function waits for finishing drawing like
 "R_GRAPHICS_Finish" function.

Argument	<code>graphics_t* self,</code>	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.53 R_GRAPHICS_EndSoftwareRendering

Outline	Notifies the graphics library of the end of software rendering.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_EndSoftwareRendering(graphics_t* self);	
Description	This function is used before software rendering to the frame buffer in On-Chip RAM. To call this function from functions that support error processing, also call R_GRAPHICS_EndRenderingInFin at the end of the function.	
Argument	graphics_t* self,	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.1.54 R_GRAPHICS_EndRenderingInFin

Outline	Call this function from the end of the function that performs software rendering.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_EndRenderingInFin(graphics_t* self, errnum_t e);	
Description	This function is used before software rendering to the frame buffer in On-Chip RAM. To call this function from functions that support error processing, also call R_GRAPHICS_EndRenderingInFin at the end of the function.	
Argument	graphics_t* self,	Graphics drawing context
	errnum_t e	Errors that have occurred. No error = 0.
Return value	Error code. If there is no error, the return value is 0.	

5.2 Functions Equivalent to graphics_image_t Class Member Function

5.2.1 List of Functions

Section	Function Name	Description
5.2.2	R_GRAPHICS_IMAGE_InitByShareFrameBuffer	Initializes the frame buffer data as an image.
5.2.3	R_GRAPHICS_IMAGE_InitR8G8B8A8	Initializes the r8g8b8a8_t image object.
5.2.4	R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8	Initializes the image object to the same width and height.
5.2.5	R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8	Initializes the image object to which a part of frame buffer being displayed is copied.
5.2.6	R_GRAPHICS_IMAGE_GetProperties	Get properties of the image.
5.2.7	R_GRAPHICS_IMAGE_InitByShareFrameBufferEx	Initializes the frame buffer data as an image
5.2.8	R_GRAPHICS_IMAGE_GetImageFormat	Get the image format.

5.2.2 R_GRAPHICS_IMAGE_InitByShareFrameBuffer

Outline	Initializes the frame buffer data as an image.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_IMAGE_InitByShareFrameBuffer(graphics_image_t* self, frame_buffer_t* frame_buffer);	
Description	<p>Initializes internal variables.</p> <p>The VRAM area indicated by</p> <pre>frame_buffer->buffer_address [frame_buffer->show_buffer_index]</pre> <p>at the time when this function is called is shared with image (self).</p> <p>[Condition] This function cannot initialize CLUT format image. RGA supports static CLUT format image made from ImagePackager.</p> <p>[Condition] When wrong stride alignment error was raised, change width of the buffer which becomes to fit stride alignment and copy the image to the buffer. RGA cannot copy at this condition. Also, draw the source image (a part of the buffer) by calling "R_GRAPHICS_DrawImageChild" function.</p>	
Argument	graphics_image_t* self	Image
	frame_buffer_t* frame_buffer	Frame buffer that contains the image
Return value	Error code. If there is no error, the return value is 0.	

5.2.3 R_GRAPHICS_IMAGE_InitR8G8B8A8

Outline	Initializes the r8g8b8a8_t image object.	
Header	RGA.h	
Declaration	<pre>errnum_t R_GRAPHICS_IMAGE_InitR8G8B8A8(graphics_image_t* self, void* image_data_array, size_t image_data_array_size, int_fast32_t width, int_fast32_t height);</pre>	
Description	<p>Initializes internal variables.</p> <p>Acquirable image data is arranged in a uint8_t-type array in the order of Red, Green, Blue, and Alpha from the upper-left pixel to the lower-right pixel.</p> <p>R8G8B8A8 format is not same as ARGB8888 format (Figure 1-4). R8G8B8A8 format is drawn by software rendering.</p> <p>To create an image of another pixel format, call 5.2.2. R_GRAPHICS_IMAGE_InitByShareFrameBuffer.</p>	
Argument	graphics_image_t* self	Image
	void* image_data_array	Starting address of an array to be a memory area that stores images
	size_t image_data_array_size	Size (bytes) of the memory area indicated by image_data_array
	int_fast32_t width, int_fast32_t height	Width and height of image
Return value	Error code. If there is no error, the return value is 0.	

5.2.4 R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8

Outline	Initializes the image object to the same width and height.	
Header	RGA.h	
Declaration	<pre>errnum_t R_GRAPHICS_IMAGE_InitSameSizeR8G8B8A8(graphics_image_t* self, void* image_data_array, size_t image_data_array_size, graphics_image_t* same_size_image);</pre>	
Description	<p>Initializes internal variables.</p> <p>R8G8B8A8 format is not same as ARGB8888 format (Figure 1-4). R8G8B8A8 format is drawn by software rendering.</p> <p>To create an image of another pixel format, call 5.2.2. R_GRAPHICS_IMAGE_InitByShareFrameBuffer.</p>	
Argument	graphics_image_t* self	Image
	void* image_data_array	Starting address of an array to be a memory area that stores images
	size_t image_data_array_size	Size (bytes) of the memory area indicated by image_data_array
	graphics_image_t* same_size_image	Image object that references width and height
Return value	Error code. If there is no error, the return value is 0.	

5.2.5 R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8

Outline	Initializes the image object to which a part of frame buffer being displayed is copied.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_IMAGE_InitCopyFrameBufferR8G8B8A8(graphics_image_t* self, void* image_data_array, size_t image_data_array_size, graphics_t* context, int_fast32_t min_x, int_fast32_t min_y, int_fast32_t width, int_fast32_t height);
Description	Initializes internal variables. Acquirable image data is arranged in a uint8_t-type array in the order of Red, Green, Blue, and Alpha from the upper-left pixel to the lower-right pixel. R8G8B8A8 format is not same as ARGB8888 format. R8G8B8A8 format is drawn by software rendering.

To create an image of another pixel format, call 5.2.2.

R_GRAPHICS_IMAGE_InitByShareFrameBuffer.

Argument	graphics_image_t* self	Image
	void* image_data_array	Starting address of an array to be a memory area that stores images
	size_t image_data_array_size	Size (bytes) of the memory area indicated by "image_data_array" argument
	graphics_t* context	Context with copy source frame buffer to be drawn
	int_fast32_t min_x, int_fast32_t min_y	Minimum X and Y coordinates (frame buffer coordinates) of the range to be acquired
	int_fast32_t width, int_fast32_t height	Width and height of the range to be acquired
Return value	Error code. If there is no error, the return value is 0.	

5.2.6 R_GRAPHICS_IMAGE_GetProperties

Outline	Get image properties.
Header	RGA.h
Declaration	errnum_t R_GRAPHICS_IMAGE_GetProperties(graphics_image_t* self, graphics_image_properties_t* out_properties);
Description	Flush operation must be done before the array pointed by "out_properties->address" is read or written. The data in the ROM does not have to be flushed. When you want to flush, flush CPU cache directly, call "R_GRAPHICS_Finish" or sandwich reading or writing the image data between "R_GRAPHICS_BeginSoftwareRendering" and "R_GRAPHICS_EndSoftwareRendering".

Argument	graphics_image_t* self	Image
	graphics_image_properties_t* out_properties	(Output) Image properties
Return value	Error code. If there is no error, the return value is 0.	

5.2.7 R_GRAPHICS_IMAGE_InitByShareFrameBufferEx

Outline	Initializes the frame buffer data as an image	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_IMAGE_InitByShareFrameBufferEx(graphics_image_t* self, frame_buffer_t* in_FrameBuffer, graphics_t* in_GraphicsContext);	
Description	Regardless of pipeline mode, this function executes correctly. The function is same as <R_GRAPHICS_IMAGE_InitByShareFrameBuffer>.	
Argument	graphics_image_t* self	Image to initialize
	frame_buffer_t* in_FrameBuffer	Frame buffer that contains the image
	graphics_t* in_GraphicsContext	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.2.8 R_GRAPHICS_IMAGE_GetImageFormat

Outline	Get the image format.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_IMAGE_GetImageFormat(const graphics_image_t* self, pixel_format_t* out_Format);	
Description		
Argument	graphics_image_t* self	Image to be initialized
	pixel_format_t* out_Format	Frame buffer containing the contents of the image
Return value	Error code. If there is no error, the return value is 0.	

5.3 Functions Equivalent to graphics_pattern_t Class Member Function**5.3.1 List of Functions**

Section	Function Name	Description
5.3.2	R_GRAPHICS_PATTERN_Initialize	Initializes the pattern object.

5.3.2 R_GRAPHICS_PATTERN_Initialize

Outline	Initializes the pattern object.	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_PATTERN_Initialize(graphics_pattern_t* self, graphics_image_t* image, repetition_t repetition, graphics_t* context);	
Description	Initializes internal variables. Set the object of GraphicsPatternClass to R_GRAPHICS_SetFillPattern.	
Argument	graphics_pattern_t* self	Pattern object
	graphics_image_t* image	Pattern component image
	repetition_t repetition	Repetition setting (normally GRAPHICS_REPEAT)
	graphics_t* context	Belonging context
Return value	Error code. If there is no error, the return value is 0.	

5.4 Functions Equivalent to window_surfaces_t Class Member Functions

5.4.1 List of Functions

Section	Function Name	Description
5.4.2	R_WINDOW_SURFACES_InitConst	Initializes internal variables with constants.
5.4.3	R_WINDOW_SURFACES_Initialize	Initializes the display device or window and starts displaying graphics.
5.4.4	R_WINDOW_SURFACES_Finalize	Finalizes the display device.
5.4.5	R_WINDOW_SURFACES_GetLayerFrameBuffer	Acquires the pointer to the frame buffer structure of the specified layer.
5.4.6	R_WINDOW_SURFACES_GetLayerCount	Acquires the number of layers.
5.4.7	R_WINDOW_SURFACES_SwapBuffers	Swaps the buffer of the specified layer and displays the drawn content.
5.4.8	R_WINDOW_SURFACES_DoMessageLoop	Enters the message loop.
5.4.9	R_WINDOW_SURFACES_AccessLayerAttributes	Accesses attributes of the specified display layer.
5.4.10	R_WINDOW_SURFACES_AllocOffscreenStack	Allocates the off-screen buffer from VRAM stack.
5.4.11	R_WINDOW_SURFACES_FreeOffscreenStack	Frees the off-screen buffer to VRAM stack.
5.4.12	R_WINDOW_SURFACES_SetLayerAttributes	Sets attributes of the specified display layer.
5.4.13	R_WINDOW_SURFACES_CONFIG_SetEmpty	Initializes the values of all member variables with values that mean empty.
5.4.14	R_LAYER_ATTRIBUTES_SetEmpty	Initializes the values of all member variables with values that mean empty.

5.4.2 R_WINDOW_SURFACES_InitConst

Outline	Initializes internal variables with constants.	
Header	RGA.h, window_surfaces.h	
Declaration	void R_WINDOW_SURFACES_InitConst(window_surfaces_t* self);	
Description		
Argument	window_surfaces_t* self	Frame buffers and screen display
Return value	None	

5.4.3 R_WINDOW_SURFACES_Initialize

Outline	Initializes the display device or window.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_Initialize(window_surfaces_t* self, window_surfaces_config_t* in_out_config);	
Description	To control display independently, directly use the frame_buffer_t structure instead of the window_surfaces_t class. Initializes internal variables. The entire screen becomes black after initialization. After calling "R_WINDOW_SURFACES_SwapBuffers" function, to show is started.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	window_surfaces_config_t* in_out_config	
Return value	Error code. If there is no error, the return value is 0.	

5.4.4 R_WINDOW_SURFACES_Finalize

Outline	Finalizes the display device.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_Finalize (window_surfaces_t* self, errnum_t e);	
Description		
Argument	window_surfaces_t* self	Frame buffers and screen display
	errnum_t e	Errors that have occurred. No error = 0.
Return value	Error code or e. 0 = "successful and e = 0".	

5.4.5 R_WINDOW_SURFACES_GetLayerFrameBuffer

Outline	Acquires the pointer to the frame buffer structure of the specified layer.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_GetLayerFrameBuffer(window_surfaces_t* self, int_t layer_num, frame_buffer_t** out_frame_buffer);	
Description	When the attribute of frame buffer is changed, call "R_WINDOW_SURFACES_AccessLayerAttributes" function.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_t layer_num	The layer number 0: Innermost, +1: Next to the innermost layer
	frame_buffer_t** out_frame_buffer	(Output) Frame buffer structure
Return value	Error code. If there is no error, the return value is 0.	

5.4.6 R_WINDOW_SURFACES_GetLayerCount

Outline	Acquires the number of layers.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_GetLayerCount(window_surfaces_t* self, int_t* out_layer_count);	
Description		
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_t* out_layer_count	(Output) Number of layers
Return value	Error code. If there is no error, the return value is 0.	

5.4.7 R_WINDOW_SURFACES_SwapBuffers

Outline	Swaps the buffer of the specified layer and displays the drawn content.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_SwapBuffers(window_surfaces_t* self, int_t layer_num, graphics_t graphics);	
Description	If buffer was single buffer, buffer is not swapped. The drawn content is displayed without calling this function, but the progress of the drawing is displayed instead. If buffer was double buffer, the content after swapping is front of 2 frames. Call "R_GRAPHICS_Clear" function or call "R_GRAPHICS_IMAGE_InitByShareFrameBuffer" function with showing frame buffer and copy front of 1 frame by calling "R_GRAPHICS_DrawImage" function. When Graphics = NULL is specified, completion of drawing is not waited before the buffer is swapped.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_t layer_num	The layer number 0: Innermost, +1: Next to the innermost layer
	graphics_t graphics	The drawn graphics context, NULL enabled
Return value	Error code. If there is no error, the return value is 0.	

5.4.8 R_WINDOW_SURFACES_DoMessageLoop

Outline	Enters the message loop.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_DoMessageLoop(window_surfaces_t* self);	
Description	Upon completion of the application, the processing returns from this function. For the terminating method, see the sub-class specifications.	
Argument	window_surfaces_t* self	Frame buffers and screen display
Return value	Error code. If there is no error, the return value is 0.	

5.4.9 R_WINDOW_SURFACES_AccessLayerAttributes

Outline	Accesses attributes of the specified display layer.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_AccessLayerAttributes(window_surfaces_t* self, layer_attributes_t* in_out_attributes);	
Description	Not all attributes are available. It depends on the device and support status.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	layer_attributes_t* in_out_attributes	(Input/output) Attributes of layer
Return value	Error code. If there is no error, the return value is 0.	

5.4.10 R_WINDOW_SURFACES_AllocOffscreenStack

Outline	Allocates the off-screen buffer from VRAM stack.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_AllocOffscreenStack(window_surfaces_t* self, frame_bufer_t* in_out_frame_buffer);	
Description	Input member variable in "frame_buffer_t" is "stride", "height", "buffer_count". Output member variable in "frame_buffer_t" is all element of "buffer_address" array. If the memory was few, E_FEW_ARRAY error is returned. Allocated off-screen buffer is freed by calling "R_WINDOW_SURFACES_Finalize" function.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	frame_bufer_t*	(Input/output) The off-screen buffer
	in_out_frame_buffer	
Return value	Error code. If there is no error, the return value is 0.	

5.4.11 R_WINDOW_SURFACES_FreeOffscreenStack

Outline	Frees the off-screen buffer to VRAM stack.	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_FreeOffscreenStack(window_surfaces_t* self, frame_bufer_t* frame_buffer);	
Description	Input member variable in "frame_buffer_t" is "buffer_count", "buffer_address". If freeing order is not reverse of allocating order, E_ACCESS_DENIED error is returned.	
Argument	window_surfaces_t* self	Frame buffers and screen display
	frame_bufer_t*	(Input/output) The freeing off-screen buffer
	in_out_frame_buffer	
Return value	Error code. If there is no error, the return value is 0.	

5.4.12 R_WINDOW_SURFACES_SetLayerAttributes

Outline	Sets attributes of the specified display layer	
Header	RGA.h, window_surfaces.h	
Declaration	errnum_t R_WINDOW_SURFACES_SetLayerAttributes(window_surfaces_t* self, int_fast32_t const in_LayerNum, layer_attributes_t* in_Attributes);	
Description		
Argument	window_surfaces_t* self	Frame buffers and screen display
	int_fast32_t const in_LayerNum	Target display layer index number to set
	layer_attributes_t* in_Attributes	Attributes of layer
Return value	Error code. If there is no error, the return value is 0.	

5.4.13 R_WINDOW_SURFACES_CONFIG_SetEmpty

Outline	Initializes all member variables to the value meaning empty	
Header	RGA.h, window_surfaces.h	
Declaration	void R_WINDOW_SURFACES_CONFIG_SetEmpty(window_surfaces_config_t* out);	
Description	See <Section_ValueMeaningEmptyAndFlaggedParameter> about the value meaning empty.	
Argument	window_surfaces_config_t* out	A structure initializing the member variable
Return value	None	

5.4.14 R_LAYER_ATTRIBUTES_SetEmpty

Outline	Initializes all member variables to the value meaning empty	
Header	RGA.h, window_surfaces.h	
Declaration	void R_LAYER_ATTRIBUTES_SetEmpty(layer_attributes_t* out);	
Description	See <Section_ValueMeaningEmptyAndFlaggedParameter> about the value meaning empty.	
Argument	layer_attributes_t* out	A structure initializing the member variable
Return value	None	

5.5 Functions Related to byte_per_pixel_t Class

5.5.1 List of Functions

Section	Function Name	Description
5.5.2	R_RGA_BitPerPixelType_To_BytePerPixelType	Converts the number of bits per pixel to the number of bytes per pixel (with decimal part).
5.5.3	R_RGA_BytePerPixelType_To_BitPerPixelType	Converts the number of bytes per pixel (with decimal part) to the number of bits per pixel.
5.5.4	R_BYTE_PER_PIXEL_IsInteger	Returns information on whether the number of bytes per pixel is an integer.
5.5.5	R_BIT_PER_PIXEL_GetBytePerPixel	Convert from byte per pixel to bit per pixel
5.5.6	R_BYTE_PER_PIXEL_GetBitPerPixel	Convert from bit per pixel to byte per pixel
5.5.7	R_RGA_BYTE_PER_PIXEL_IsInteger	Returns information on whether the number of bytes per pixel is integer

5.5.2 R_RGA_BitPerPixelType_To_BytePerPixelType

Outline	Converts the number of bits per pixel to the number of bytes per pixel (with decimal part).	
Header	RGA.h	
Declaration	byte_per_pixel_t R_RGA_BitPerPixelType_To_BytePerPixelType(int_t bit_per_pixel);	
Description		
Argument	int_t bit_per_pixel	Number of bits per pixel
Return value	Number of bytes per pixel (with decimal part)	

5.5.3 R_RGA_BytePerPixelType_To_BitPerPixelType

Outline	Converts the number of bytes per pixel (with decimal part) to the number of bits per pixel.	
Header	RGA.h	
Declaration	int_t R_RGA_BytePerPixelType_To_BitPerPixelType(byte_per_pixel_t byte_per_pixel);	
Description		
Argument	byte_per_pixel_t byte_per_pixel	Number of bytes per pixel (with decimal part)
Return value	Number of bits per pixel	

5.5.4 R_BYTE_PER_PIXEL_IsInteger

Outline	Returns information on whether the number of bytes per pixel is an integer.	
Header	RGA.h	
Declaration	bool_t R_BYTE_PER_PIXEL_IsInteger(byte_per_pixel_t byte_per_pixel);	
Description		
Argument	byte_per_pixel_t byte_per_pixel	Number of bytes per pixel (with decimal part)
Return value	Information on whether the number of bits per pixel is an integer	

5.5.5 R_BIT_PER_PIXEL_GetBytePerPixel

Outline Convert from byte per pixel to bit per pixel
 Header RGA.h
 Declaration `byte_per_pixel_t R_BIT_PER_PIXEL_GetBytePerPixel(int_fast32_t in_BitPerPixel);`

Description

Argument

in_BitPerPixel	
----------------	--

Return value

The value as byte_per_pixel_t

5.5.6 R_BYTE_PER_PIXEL_GetBitPerPixel

Outline Convert from bit per pixel to byte per pixel
 Header RGA.h
 Declaration `int_fast32_t R_BYTE_PER_PIXEL_GetBitPerPixel(byte_per_pixel_t in_BytePerPixel);`

Description

Argument

in_BytePerPixel	
-----------------	--

Return value

BitPerPixel

5.5.7 R_RGA_BYTE_PER_PIXEL_IsInteger

Outline Returns information on whether the number of bytes per pixel is integer.
 Header RGA.h
 Declaration `bool_t R_RGA_BYTE_PER_PIXEL_IsInteger(byte_per_pixel_t in_BytePerPixel);`

Description

Argument

in_BytePerPixel	Number of bytes per pixel (with decimal part)
-----------------	---

Return value

Information on whether the number of bits per pixel is integer
--

5.6 Functions Related to animation_timing_function_t Class

5.6.1 List of Functions

Section	Function Name	Description
5.6.2	R_Get_AnimationTimingFunction	Gets defined animation timing.
5.6.3	R_ANIMATION_TIMING_FUNCTION_GetValue	Calculates an attribute value at the specified elapsed time from the time starting animation.

5.6.2 R_Get_AnimationTimingFunction

Outline Gets defined animation timing.

Header RGA.h

Declaration `errnum_t R_Get_AnimationTimingFunction(char* timing_name, animation_timing_function_t** out_timing);`

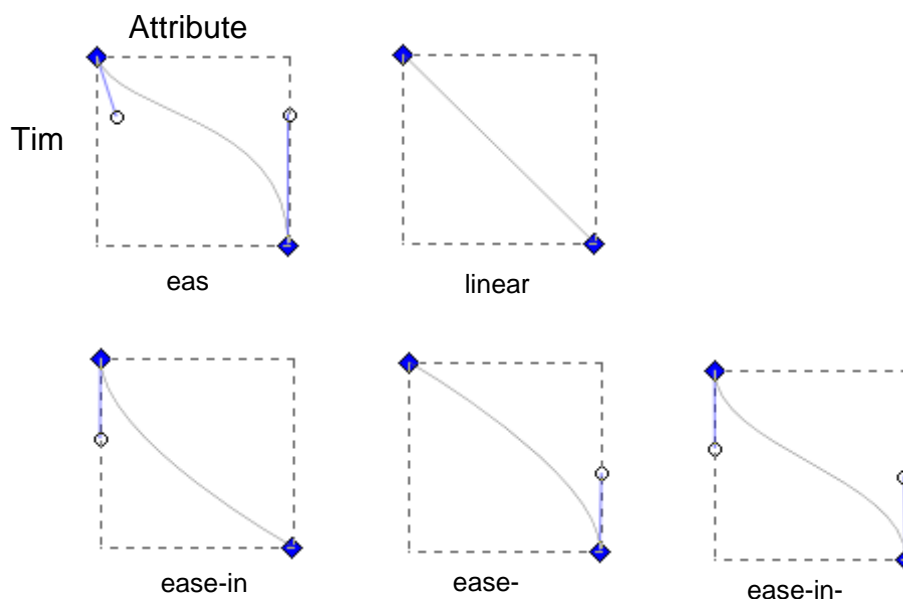
Description

Argument

<code>char* timing_name</code>	The name of animation timing. The following name can be passed: "ease", "linear", "ease_in", "ease_out", "ease_in_out"
<code>animation_timing_function_t** out_timing</code>	(Output) The address of animation timing object
Return value	Error code. If there is no error, the return value is 0.

5.6.3 R_ANIMATION_TIMING_FUNCTION_GetValue

Outline	Calculates an attribute value at the specified elapsed time from the time starting animation.
Header	RGA.h
Declaration	float32_t R_ANIMATION_TIMING_FUNCTION_GetValue(animation_timing_function_t* self, float32_t clamp_time, float32_t value_of_previous_keyframe, float32_t value_of_next_keyframe);
Description	Attribute value is user defined position value, color value or other value changed by time



Example:

```
timing_name="linear", value_of_previous_keyframe=10,  
value_of_next_keyframe=20, clamp_time=0.5
```

This case returns 15.

Argument	animation_timing_function_t* self	The animation timing object
	float32_t clamp_time	The percentage of time from previous key frame (clamp_time=0.0) to next key frame (clamp_time=1.0). (decimal from 0.0 to 1.0)
	float32_t value_of_previous_keyframe	The value of attribute at the previous key frame
	float32_t value_of_next_keyframe	The value of attribute at the next key frame
Return value	The value of attribute at the time of "clamp_time" argument	

5.7 OS Porting Layer of JCU driver

5.7.1 List of Functions

Section	Function Name	Description
5.7.2	R_GRAPHICS_OnInitialize	Callback function setting a default value of <graphics_config_t>
5.7.3	R_GRAPHICS_OnFinalize	Callback function that releases memory allocated in <R_GRAPHICS_OnInitialize> and so on
5.7.4	R_GRAPHICS_OnInitialized	Callback function called from the end of <R_GRAPHICS_Initialize>
5.7.5	R_GRAPHICS_JCU_ClearCodecEvent	Clears JCU interrupt signaled event
5.7.6	R_GRAPHICS_JCU_SetCodecEvent	Notifies from end of JCU interrupt service to the thread
5.7.7	R_GRAPHICS_JCU_WaitForCodecEvent	Waits for JCU interrupt
5.7.8	R_GRAPHICS_JCU_ClearTerminateEvent	Clears JCU terminated event
5.7.9	R_GRAPHICS_JCU_SetTerminateEvent	Notifies from JCU terminated event to the thread
5.7.10	R_GRAPHICS_JCU_WaitForTerminateEvent	Waits for JCU terminated event

5.7.2 R_GRAPHICS_OnInitialize

Outline	Callback function setting a default value of <graphics_config_t>	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_OnInitialize(graphics_t* self, graphics_config_t* in_out_Config, void** out_CalleeDefined);	
Description	The following list is an example of "out_CalleeDefined" argument. - an address of memory allocated in this function	
Argument	graphics_t* self	Address of the object (non-initialized) at which initialization starts
	graphics_config_t* in_out_Config,	Target of setting default value
	void** out_CalleeDefined	A value defined by this function.
Return value	Error code. If there is no error, the return value is 0.	

5.7.3 R_GRAPHICS_OnFinalize

Outline	Callback function that releases memory allocated in <R_GRAPHICS_OnInitialize> and so on	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_OnFinalize(graphics_t* self, void* in_CalleeDefined, errnum_t e);	
Description		
Argument	graphics_t* self	Address of the object that has been finalized
	void* in_CalleeDefined	Output value of "out_CalleeDefined" argument of <R_GRAPHICS_OnInitialize>
	errnum_t e	Errors that have occurred. No error = 0
Return value	Error code. If there is no error, the return value is 0.	

5.7.4 R_GRAPHICS_OnInitialized

Outline	Callback function called from the end of <R_GRAPHICS_Initialize>	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_OnInitialized(graphics_t* self);	
Description		
Argument	graphics_t* self	Graphics drawing context
Return value	Error code. If there is no error, the return value is 0.	

5.7.5 R_GRAPHICS_JCU_ClearCodecEvent

Outline	Clears JCU interrupt signaled event	
Header	RGA.h	
Declaration	void R_GRAPHICS_JCU_ClearCodecEvent(void);	
Description	This function clears the event that is set by <R_GRAPHICS_JCU_SetCodecEvent>.	
Argument	None	
Return value	None	

5.7.6 R_GRAPHICS_JCU_SetCodecEvent

Outline	Notifies from end of JCU interrupt service to the thread	
Header	RGA.h	
Declaration	void R_GRAPHICS_JCU_SetCodecEvent(void);	
Description	See <R_DRW_SetInterruptEvent>.	
Argument	None	
Return value	None	

5.7.7 R_GRAPHICS_JCU_WaitForCodecEvent

Outline	Waits for JCU interrupt	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_JCU_WaitForCodecEvent(void);	
Description	See <R_DRW_WaitForInterruptEvent>.	
Argument	None	
Return value	None	

5.7.8 R_GRAPHICS_JCU_ClearTerminateEvent

Outline	Clears JCU terminated event	
Header	RGA.h	
Declaration	void R_GRAPHICS_JCU_ClearTerminateEvent(void);	
Description	This function clears the event that is set by <R_GRAPHICS_JCU_SetTerminateEvent>.	
Argument	None	
Return value	None	

5.7.9 R_GRAPHICS_JCU_SetTerminateEvent

Outline	Notifies from JCU terminated event to the thread	
Header	RGA.h	
Declaration	void R_GRAPHICS_JCU_SetTerminateEvent(void);	
Description		
Argument	None	
Return value	None	

5.7.10 R_GRAPHICS_JCU_WaitForTerminateEvent

Outline	Waits for JCU terminated event	
Header	RGA.h	
Declaration	errnum_t R_GRAPHICS_JCU_WaitForTerminateEvent(void);	
Description	See <R_DRW_WaitForInterruptEvent>.	
Argument	None	
Return value	Error code. If there is no error, the return value is 0.	

5.8 OS Porting Layer of DRW driver

5.8.1 List of Functions

Section	Function Name	Description
5.8.2	R_DRW_OnInitialize	Callback function called at initializing DRW
5.8.3	R_DRW_OnFinalize	Callback function called at finalizing DRW
5.8.4	R_DRW_EnableInterrupt	Enables DRW interrupt
5.8.5	R_DRW_DisableInterrupt	Disables DRW interrupt
5.8.6	R_DRW_WaitForInterruptEvent	Waits for DRW interrupt
5.8.7	R_DRW_SetInterruptEvent	Notifies from DRW interrupt to the thread
5.8.8	R_DRW_CheckThread	Checks if the current thread is locked thread
5.8.9	R_DRW_FlushCache	Flushes level 1 CPU cache
5.8.10	R_DRW_ToPhysicalAddress	Changes to physical address
5.8.11	R_DRW_ToCachedAddress	Checks if the address is in the L1 cache area or changes to L1 cache area
5.8.12	R_DRW_ToUncachedAddress	Checks if the address is in L1 uncached area or changes to L1 uncached area
5.8.13	R_DRW_OnInterrupting	Execute processing when DRW interrupt was raised

5.8.2 R_DRW_OnInitialize

Outline	Callback function called at initializing DRW	
Header	r_drw_pl.h	
Declaration	errnum_t R_DRW_OnInitialize(void);	
Description	DRW driver expects this function to do the following operation. <ul style="list-style-type: none"> - Starts to supply DRW clock - Registers DRW interrupt callback function calling <R_DRW_OnInterrupting> function - Sets DRW interrupt priority 	
Argument	None	
Return value	Error code. If there is no error, the return value is 0.	

5.8.3 R_DRW_OnFinalize

Outline	Callback function called at finalizing DRW	
Header	r_drw_pl.h	
Declaration	void R_DRW_OnFinalize(void);	
Description	DRW driver expects this function to do the following operation. <ul style="list-style-type: none"> - Stop to supply DRW clock 	
Argument	None	
Return value	None	

5.8.4 R_DRW_EnableInterrupt

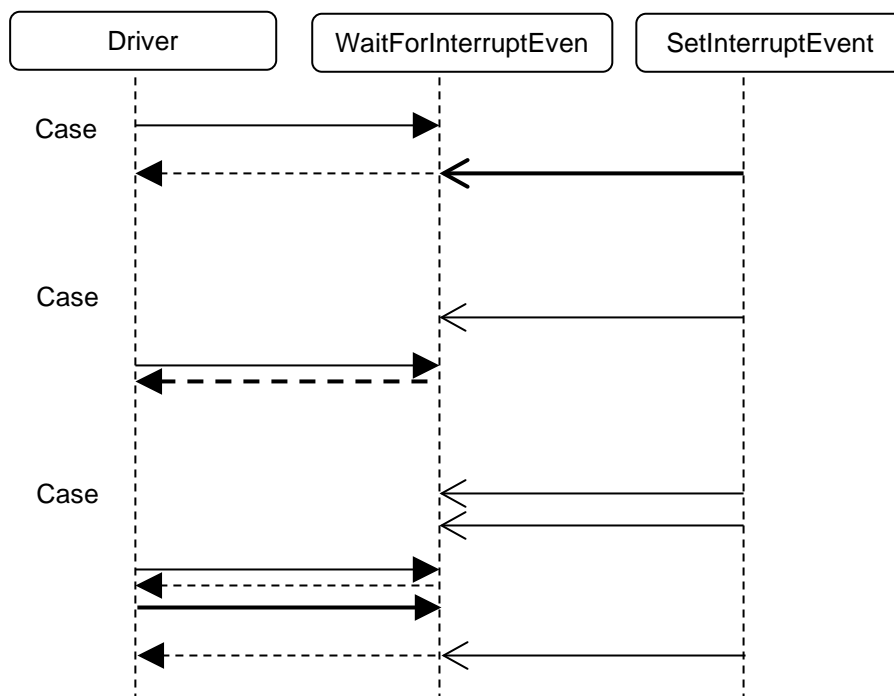
Outline	Enables DRW interrupt	
Header	r_drw_pl.h	
Declaration	void R_DRW_EnableInterrupt(void);	
Description		
Argument	None	
Return value	None	

5.8.5 R_DRW_DisableInterrupt

Outline	Disables DRW interrupt	
Header	r_drw_pl.h	
Declaration	bool_t R_DRW_DisableInterrupt(void);	
Description		
Argument	None	
Return value	Whether DRW interrupt have been enabled until now	

5.8.6 R_DRW_WaitForInterruptEvent

Outline	Waits for DRW interrupt
Header	r_drw_pl.h
Declaration	void R_DRW_WaitForInterruptEvent(void);
Description	<p>In the OS-less environment, this function does the polling inside.</p> <p>In RTOS environment, change this function define to the code of calling binary semaphore, event group or thread attached event API.</p> <p>The thread calling this function is same as the thread calling <R_DRW_OnInitialize> function.</p> <p>This function returns after calling <R_DRW_SetInterruptEvent> (Case1 of <Figure_SequenceOfWaitForInterruptEvent>).</p> <p>If <R_DRW_SetInterruptEvent> function was already called, this function returns immediately (Case2 of <Figure_SequenceOfWaitForInterruptEvent>).</p> <p>If <R_DRW_SetInterruptEvent> function was called only before previous calling this function, even if <R_DRW_SetInterruptEvent> function was called many times, this function does not return immediately. If the specification is not satisfied, when two or more factors occur consecutively, it will not wait for the next wait. (Case3 of <Figure_SequenceOfWaitForInterruptEvent>).</p>



Argument	None
Return value	None

5.8.7 R_DRW_SetInterruptEvent

Outline	Notifies from DRW interrupt to the thread	
Header	r_drw_pl.h	
Declaration	void R_DRW_SetInterruptEvent(void);	
Description	<p>This function notifies to waiting RTOS thread calling <R_DRW_WaitForInterruptEvent> function.</p> <p>If this function is called from interrupt, this function will be called from the inside of <R_DRW_OnInterrupting> function.</p> <p>Not only interrupt function but also any thread calls this function.</p> <p>Even if you call this function continuously, <R_DRW_WaitForInterruptEvent> will not return continuously.</p>	
Argument	None	
Return value	None	

5.8.8 R_DRW_CheckThread

Outline	Checks if the current thread is locked thread	
Header	r_drw_pl.h	
Declaration	void R_DRW_CheckThread(void);	
Description	<p>If the thread calling this function is not same as the thread calling <R_DRW_OnInitialize> function (=the thread locking DRW), there may be a problem related exclusive control.</p> <p>Calling RGA API must be from one thread by using thread communication for example.</p> <p>If your environment is OS less, this function can return without any operations.</p>	
Argument	None	
Return value	None	

5.8.9 R_DRW_FlushCache

Outline	Flushes level 1 CPU cache	
Header	r_drw_pl.h	
Declaration	void R_DRW_FlushCache(void* in_StartAddress, uint32_t in_Size);	
Description	<p>Flush operation of this function is to writeback and to discard.</p> <p>Arguments of this function can be ignored, if all cache lines are flushed.</p>	
Argument	void* in_StartAddress	Start address of to flush
Return value	uint32_t in_Size	Size (byte) to flush
	None	

5.8.10 R_DRW_ToPhysicalAddress

Outline	Changes to physical address	
Header	r_drw_pl.h	
Declaration	errnum_t R_DRW_ToPhysicalAddress(void* in_Address, uintptr_t* out_PhysicalAddress);	
Description	Example: <pre>> // Case that physical address is integer type: > uintptr_t physical_address; > e= R_DRW_ToPhysicalAddress(address, &physical_address); if(e){goto fin;} > > // Case that physical address is pointer type: > uintptr_t physical_address; > void* pointer; > e= R_DRW_ToPhysicalAddress(address, &physical_address); if(e){goto fin;} > pointer = (void*) physical_address;</pre>	
Argument	void* in_Address	Virtual address
Return value	uintptr_t* out_PhysicalAddress	Output: Virtual address for cached area
	Error code. If there is no error, the return value is 0.	

5.8.11 R_DRW_ToCachedAddress

Outline	Checks if the address is in the L1 cache area or changes to L1 cache area	
Header	r_drw_pl.h	
Declaration	errnum_t R_DRW_ToCachedAddress(void* in_Address, void* out_CachedAddress);	
Description	The define of this function must be modified to same setting with MMU setting. "out_CachedAddress" argument must be passed the address of pointer. If the target environment did not have mirror area and "in_Address" argument was passed uncached address, an error is raised. If an error was raised, you may know the variable by looking at value of "in_Address" argument and map file. If the environment did not have any mirror area, it is not necessary to call the function to change the address. But, the function can be used by checking whether passed address is in cached area or uncached area.	
Argument	void* in_Address	Virtual address
Return value	void* out_CachedAddress	Output: Virtual address for cached area
	Error code. If there is no error, the return value is 0.	

5.8.12 R_DRW_ToUncachedAddress

Outline	Checks if the address is in L1 uncached area or changes to L1 uncached area
Header	r_drw_pl.h
Declaration	errnum_t R_DRW_ToUncachedAddress(void* in_Address, void* out_UncachedAddress);
Description	<p>The define of this function must be modified to same setting with MMU setting.</p> <p>"out_UncachedAddress" argument is passed the address of pointer.</p> <p>If the target environment did not have mirror area and "in_Address" argument was passed cached address, an error is raised.</p> <p>If an error was raised, you may know the variable by looking at value of "in_Address" argument and map file.</p> <p>If the environment did not have any mirror area, it is not necessary to call the function to change the address. But, the function can be used by checking whether passed address is in cached area or uncached area.</p>

Argument	void* in_Address	Virtual address
Return value	void* out_UncachedAddress	Output: Virtual address for uncached area
	Error code. If there is no error, the return value is 0.	

5.8.13 R_DRW_OnInterrupting

Outline	Execute processing when DRW interrupt was raised
Header	r_drw_pl.h
Declaration	void R_DRW_OnInterrupting(void);
Description	The porting layer must be modified to call this function when DRW interrupt was raised.

Argument	None
Return value	None

5.9 Other Functions

5.9.1 List of Functions

Section	Function Name	Description
5.9.2	R_RGA_Get_R8G8B8A8	Returns the R8G8B8A8 color value.
5.9.3	R_RGA_CalcWorkBufferB_Size	Calculates the size required for the work buffer B.

5.9.2 R_RGA_Get_R8G8B8A8

Outline	Returns the R8G8B8A8 color value.	
Header	RGA.h	
Declaration	r8g8b8a8_t R_RGA_Get_R8G8B8A8(int_t red, int_t green, int_t blue, int_t alpha);	
Description		
Argument	int_t red, int_t green, int_t blue, int_t alpha	Each color component 0 to 255
Return value	R8G8B8A8 color value	

5.9.3 R_RGA_CalcWorkBufferB_Size

Outline	Calculates the size required for the work buffer B.
Header	RGA.h
Declaration	size_t R_RGA_CalcWorkBufferB_Size(int_t MaxWidthOfJPEG, int_t MaxHeightOfJPEG, int_t MaxBytePerPixelOfFrameBuffer);
Description	<p>Parameters may change in the future.</p> <p>R_RGA_CalcWorkBufferB_Size is the #define macro.</p> <p>Requested size for work buffer B:</p> $\text{ceil_16(MaxWidthOfJPEG)} * \text{ceil_16(MaxHeightOfJPEG)} * \text{MaxBytePerPixelOfFrameBuffer [Bytes]}$ <p>ceil_16: round up to multiples of 16</p>

It is possible to set the size of work buffer B to 0, if JPEG/PNG is not drawn.

When PNG image was decoded, the work area (size is set to work_size_for_libPNG member variable) of LibPNG is allocated in work buffer B. But R_RGA_CalcWorkBufferB_Size macro is not added this size.

The work buffer B is not requested, if all following condition were fulfilled (JPEG Codec Unit (JCU) in RZ/A2M can draw directly) or any JPEG images are not drawn. The drawing target address of left up of JPEG image can be divided by 8. The size of JPEG image is multiples of MCU (Minimum Coded Unit). The size is depended on the pixel format of the JPEG data.

- 16pixels x 8lines (JPEG image is YCbCr422 format)
- 16pixels x 16lines (JPEG image is YCbCr420 format)
- 8pixels x 8lines (JPEG image is YCbCr444 format)
- 32pixels x 8lines (JPEG image is YCbCr411 format)

The matrix is unit matrix or translation only.

The return value is set in "graphics_config_t" type.

Argument	int_t MaxWidthOfJPEG	The maximum width of JPEG image
	int_t MaxHeightOfJPEG	The maximum height of JPEG image
	int_t MaxBytePerPixelOfFrameBuffer	<p>The maximum bytes of the drawing target frame buffer.</p> <p>If the matrix is not unit matrix and not translation only, this argument is 4</p>
Return value	Size (bytes) required for the work buffer	

6. Tools

6.1 Image Format Conversion by ImagePackager

The ImagePackager packs multiple image files into a single binary file (for the target board) or a source file (for the target board and PC). When image files are packed, they can also be converted to the raw format of any pixel formats. The integrated file is used as a source of binary data or C language data. In an XML file, specify image files to be packed. File names and extension name are not ignored case-insensitive comparison.

ImagePackager is one of commands in "RGA_Tools.vbs". It calls vbs files and exe files internally.

6.1.1 Operational Procedure

1. Make .image.xml file ().
2. Double click RGA_Tools.vbs file at \${RGA}\src\renesas\application\tool, select "ImagePackager" command in opened window, specify .image.xml file. This makes header file and binary file or C language data source.

```
-----
RGA Tools - Copyright(c) 2012-2015 Renesas Electronics Corporation
  1. Convert image format [RunImagePackager]
  2. Search error information [SearchErrorInformation]
Number or command >1
-----
((( [RunImagePackager_sth] )))
Renesas Image Packager - Copyright(c) 2012-2015 Renesas Electronics
Corporation
Make one binary file from many image files and other files.
Enter only : Open sample folder.
Setting file(ImagePackagerConfig.xml) >
```

3. Write #include directive with a generated header file and pass a symbol of image defined in the header file to "R_GRAPHICS_DrawImage" function or other functions.
4. If binary file was generated, write it at the address written in .image.xml file and start drawing operations. The address is written at /ImagePackager/OutputBinary/OutputHeader/@address in .image.xml file or generated header file.

ImagePackager can be started by the following command in the command prompt.

```
>cd src\renesas\application\RGA_Canvas2D\Images
>cscript //nologo ..\..\RGA_Tools.vbs RunImagePackager
BinaryImageConfig.image.xml
```

6.1.2 List of files

File	Description
RGA_Tools.vbs	Script file which contains ImagePackager
*.image.xml	Setting file of ImagePackager
(*.bmp, *.jpg, *.png)	Input image file
(Output: binary file)	A file which contains images and files to download to the target board
(Output: source file)	A source file which contains image data and file data to put in the program

6.1.3 Sample

BinaryImageConfig.image.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<ImagePackager>

<OutputBinary path="BinaryImage_RZ_A1H.c" language="C"
symbol="g_RGA_Sample_BinaryImage"
    source_template="${ImagePackagerLib}\SourceTemplate.xml#default"
    super_class="${ImagePackagerLib}\SuperClass.xml#default">
<OutputHeader path="BinaryImage_RZ_A1H.h"
include_define="BINARYIMAGE_RZ_A1H_H"/>
</OutputBinary>

<InputFiles>
<File path="BinaryHeader.txt" type="char[]" symbol="g_BinaryHeader"/>
<Image path="picture.bmp" type="graphics_image_t*" symbol="g_Picture_bmp"
output_format="RGB565"/>
<Image path="smile32.bmp" type="graphics_image_t*" symbol="g_Smile_bmp"
output_format="ARGB4444"/>
<File path="JPEG.jpg" type="graphics_image_t*" symbol="g_JPEG_jpg"/>
</InputFiles>

</ImagePackager>

```

Refer 6.1.8 regarding the basic forms of writing XML and XPath.

The folder that contains the XML file is the reference of relative paths.

A folder path or wild card can be specified for Image/@path.

Providing a fixed-value header at the top allows checking whether binary data is contained or not.

6.1.4 Types of Output Binary File (Language)

- Binary format (external reference symbol of C language) + C language header
- Binary format or S-record format (direct addressing such as in flush) + C language header
- C language source + C language header

Specified for /ImagePackager/OutputBinary/@language and /ImagePackager/OutputHeader/@address

6.1.5 File Formats in the Output Binary File

Format	Description
Offset table	The array of 4byte integer type in the first of the binary file that indicates the offset value to image data and binary data.
Raw-format image	Specified for /ImagePackager/InputFiles/Image/@output_format "ARGB8888", "XRGB8888"(X component is 0x00), "RGB565", "ARGB1555", "ARGB4444", "YUV422", "CLUT8", "CLUT4", "CLUT1", "A8", "A4", "A1" See following "The format of Raw-format image"
Binary format	When expanding JPEG data on the target board, the file is stored as it is. Specified for /ImagePackager/InputFiles/File

The format of Raw-format image:

Offset	Size	Description
0x00	4byte	The bit flags described the type of image [bit0] 1fixed [bit1] (reserved) [bit2] 1=Premultiplied alpha, 0=not Premultiplied alpha [bit3-bit15] (reserved) [bit16-bit31] 0 This endian is depended on the setting of "@endian".
0x04	4byte	The offset value to the image data from the first of Raw-format image. This endian is depended on the setting of "@endian".
0x08	4byte	(reserved)
0x0C	2byte	The image width (pixels) This endian is depended on the setting of "@endian".
0x0E	2byte	The image height (pixels) This endian is depended on the setting of "@endian".
0x10	1byte	The pixel format 0=RGB565, 2=ARGB1555, 3=ARGB4444, 6=CLUT8, 7=CLUT4, 8=CLUT1, 9=XRGB8888, 10=ARGB8888, 11=YUV422, 20=A8, 21=A4, 22=A1
0x11	7byte	(reserved)
*		Image data If the pixel format was RGB components, this endian is depended on the setting of "@endian". If the pixel format was YUV components, this endian is not depended on. The offset value of this image data is depended on the setting of "@raw_image_alignment". The byte count par one line (stride) is depended on the setting of "@raw_stride_alignment" and so on. If this image data had CLUT (Color Look Up Table, Palette), there are image data after CLUT. The elements of CLUT are ARGB8888 format depended on the endian by "@endian" setting. A value is fixed to 0xFF. The count of elements is 256 (CLUT8), 16 (CLUT4) or 2(CLUT1). Limitation for RGA: The starting address of image data drawn by RGA must be a multiple of 32. The padding at the right end of each line must not be inserted.

6.1.6 Input Formats

- BMP format: 32 bits or 24 bits (256, 16, or 2 colors for CLUT)
- PNG format: Supports data with alpha
- JPEG format

See the description on @path of "/ImagePackager/InputFiles/Image".

6.1.7 Parameters That Can Be Described in BinaryImageConfig.image.xml

/ImagePackager/OutputBinary: One or more parameters (Two or more parameters are used to output both "Binary" @language and "C" @language.)

@path	Path of the binary file output destination (Essential)
@symbol	C language's external reference symbol (global variable name) that supports all binary files (Essential) When outputting multiple binary files, do not change this parameter in each binary file. When the /ImagePackager/OutputHeader/@address was set, this parameter is a macro name having the address value.
@language	Programming language of output binary data (Optional) "Binary" and "C", "SRec"(S-record) can be set. "Binary" is set by default. Default of @table_format is changed, if the @language setting was changed. The source file which refers to image data must be recompiled.
@section	Specified section is embedded in the output source. Example: section = "BinaryImage" --- BinaryImage section is created. This is filled at \${Section} in /ImagePackager/SourceTemplate.
@endian	Endian (Optional) "LittleEndian" or "BigEndian" can be selected. @endian pointed from @super_class is set by default.
@raw_image_alignment	Alignment (bytes) of the starting address of raw-format image data (Optional) Only 2 ⁿ can be set. @raw_image_alignment pointed from @super_class is set by default. When outputting multiple binary files, do not change this parameter in each binary file. The raw-format header is not aligned by this setting.
@raw_image_alignment_symbol	The #define symbol name containing the @raw_image_alignment value (Optional) The #define symbol name is output to the header file to be generated. Compile the program using binary files by using the header. Example: raw_image_alignment_symbol = "GRAPHICS_RAW_IMAGE_ALIGNMENT" The code filled in the header file: <pre>#define RAW_IMAGE_ALIGNMENT 1</pre> The #define sentence of the symbol name is not output by default. When outputting multiple binary files, do not change this parameter in each binary file.
@raw_stride_alignment	Alignment of the number of bytes up to the line immediately below in an image to be output (Optional). The number of bytes is carried to a multiple of the value specified for (value). @raw_stride_alignment pointed from @super_class is set by default. Example: raw_stride_alignment = 1 Alignment of the number in hardware rendering RGA was changed from 32 to 1. 1 means not to add padding at the last of line.
@raw_stride_alignment_4	Pixel format of output image in which the number of bytes up to the line immediately below is 4. Two or more CSV formats can be specified. (Optional) This setting takes precedence over the @raw_stride_alignment setting.

	<p>@raw_stride_alignment_4 pointed from @super_class is set by default.</p> <p>Example: raw_stride_alignment_4 = YUV422</p>
@alpha_raw_image_width	<p>Width (pixels) when outputting A8, A4, A1 Raw format images. (optional)</p> <p>This option is to meet the limitation of RGA that the width of A8, A4, A1 Raw format source images needs to match the width of the drawing target. Outputs with the specified width regardless of the width of the input image. The default is to make the input image width the same as the output image width. An error will occur if you specify a width smaller than the input image. This option is ignored for formats other than A8, A4, A1.</p>
@table_format	<p>The format of the table. (Optional)</p> <ul style="list-style-type: none"> • "Offset": Fills not only the body but also the offset table to the body in the binary file, because re-compile does not have to be done, even if the size of the image or the file was changed. Even if XML tag was changed, it is not necessary to re-compile variable above than changing XML tag. • "Embed": Fills images and files normally. <p>Default value is the following value.</p> <ul style="list-style-type: none"> • If @language="Binary", @table_format="Offset" • If @language="C", @table_format="Embed"
@source_template	<p>ID of the template of the output source file in the case of @language="C". (Optional)</p> <p>Set the value of "/ImagePackager/SourceTemplate/@id".</p> <p>It is possible to set the reference to the other XML file.</p> <p>\${ImagePackagerLib} is replaced to the folder path containing ImagePackagerLib.vbs</p> <p>Example:</p> <pre>source_template="\${ImagePackagerLib}\SourceTemplate.xml#default"</pre>
@super_class	<p>ID of the super class attached to the binary file. (Optional)</p> <p>Set the value of "/ImagePackager/SuperClass/@id".</p> <p>It is possible to set the reference to the other XML file.</p> <p>\${ImagePackagerLib} is replaced to the folder path containing ImagePackagerLib.vbs</p> <p>Example:</p> <pre>super_class="\${ImagePackagerLib}\SuperClass.xml#default"</pre>

/ImagePackager/OutputBinary/OutputHeader: One or more elements

/ImagePackager/OutputHeader: Zero or one element

@path	Path of the header file output destination (Essential)
@include_define	The #define symbol name to prevent the header file from being included twice (Optional) Default name: "__BINARY_IMAGE__"
@address	Memory address to allocate binary files (Optional) Specify the memory address in the 0x00000000 format. This parameter is used when allocating addresses directly in the flash memory in addition to the program image to be linked. Since the set address is output to the header file to be created, compile the program that references binary data by using the header. If this parameter is omitted, the C language's external reference symbol (global variable name) specified for /ImagePackager/OutputBinary/@symbol is used.

/ImagePackager/InputFiles: Only one element

@base_folder	A path to be the reference of /ImagePackager/InputFiles/Image/@path (Optional) The reference of relative paths is the folder that contains the BinaryImageConfig.image.xml file. Default path: "."
--------------	---

/ImagePackager/InputFiles/Image: Zero, one or more elements

The data converted to Raw-format are embedded in the binary file. "File" tag should be specified, when JPEG and PNG data will be decoded in the target board.

@path	Path of image files to be input (Essential) As the reference of relative paths, the path of the /ImagePackager/InputFiles/@base_folder folder and the sub-folder are also input when a wild card is specified. In the case of "png" or "jpg" extension, incompressibly expanded files are output to binary files. When expanding a file on the target board, specify JPEG file for /ImagePackager/InputFiles/File. For images with no alpha channel or images whose all alpha values are 0xFF, information that shows that alpha blending is not required is embedded in the output data. This information may allow fast drawing.
@type	Type of symbols described in the header file (Essential) @symbol type. Specify "graphics_image_t*" in usually case. Specify "uint8_t[]", if you want to cut a header that length is 0x18 byte before image data.
@symbol	A symbol described in the header file (Essential) The #define symbol in the global scope. @type type When "\${...}" is used, the symbol is replaced with a file name. Example: g_\${BaseName}_\${Extension}_\${Format} - > g_Sample_jpg_ARGB8888

@output_format	<p>Format of files to be output (Essential)</p> <p>If input file was 24bit/32bit Window bitmap file, JPEG file, PNG file (with alpha/without alpha), values "ARGB8888", "XRGB8888"(X component is 0x00), "RGB565", "ARGB1555", "ARGB4444", "YUV422", "A8", "A4" and "A1" can be specified.</p> <p>When "CLUT8" is specified, input a 256-color Windows bitmap file.</p> <p>When "CLUT4" is specified, input a 16-color Windows bitmap file.</p> <p>When "CLUT1" is specified, input a monochrome Windows bitmap file.</p> <p>The following table describes alpha component of output image. If input image was 256, 16 colors or monochrome Windows bitmap file, see the input color as CLUT referenced color at the following table. Notice: PNG format created by Paint Brush for Windows 7 always has alpha component.</p> <table><tr><th>Input Image</th><th>Output Image</th><th>Alpha component of output</th></tr><tr><td>With A</td><td>Every format</td><td>A component of the input image</td></tr><tr><td rowspan="2">Without A</td><td>ARGB</td><td>0xFF</td></tr><tr><td>A component only</td><td>Y component (luminance) converted input image from RGB to YCbCr</td></tr></table> <p>Two or more values can be specified in the CSV format. In that case, however, include "\${Format}" in @symbol. "\${Format}" will be replaced to the name of pixel format.</p>	Input Image	Output Image	Alpha component of output	With A	Every format	A component of the input image	Without A	ARGB	0xFF	A component only	Y component (luminance) converted input image from RGB to YCbCr
Input Image	Output Image	Alpha component of output										
With A	Every format	A component of the input image										
Without A	ARGB	0xFF										
	A component only	Y component (luminance) converted input image from RGB to YCbCr										
@premultiplied_alpha	<p>This parameter shows whether to multiply RGB components by the alpha component. (Optional)</p> <ul style="list-style-type: none">● "no": Not multiplied (default)● "yes": RGB components are converted to multiplied values before they are output. This choice is available only when the alpha component is not contained in the drawing destination.● "already_yes": The input image has already been multiplied. This choice is available only when the alpha component is not contained in the drawing destination.											

/ImagePackager/InputFiles/File: Zero, one or more elements

Not converted data are embedded in the binary file.

@path	Path of files to be input (Essential) Reference of relative paths: /ImagePackager/InputFiles/@base_folder Files in sub folder are input, if the value had wildcard.
@type	Type of symbols described in the header file (Optional) @symbol type. When the file content is an array, attach [] to the end of the type name. uint8_t[] is set by default.
@symbol	A symbol described in the header file (Essential) The #define symbol in the global scope. uint8_t[] type.
@alignment	The address alignment of the file. (Optional) @alignment pointed from /ImagePackager/OutputBinary/@super_class is set by default.

/ImagePackager/InputFiles/Var: Zero, one or more elements

The values written in XML file are embedded in the binary file.

@type	Type of the symbol written in the header file. (Essential) int32_t, uint32_t, int16_t, uint16_t, int8_t, uint8_t, void**, void* can be specified. Specify a position (offset) of target of the pointer at @value. 4-byte value is filled in the binary, even if any type of the pointer. A symbol specified at @symbol is a value of address in C language.
@symbol	The symbol name written in the header file. (Essential) This name is a name of global variable initialized the value embedded in the binary file.
@value	The value embedded in the binary file. (Essential) If binary was written in ROM, the value is constant. If binary was written in RAM, the value is in the variable. Integer or the following special format can be specified. <ul style="list-style-type: none"> ● Example of integer: "10", "-10", "0xFF" ● Special format "(new Image('file_path')).width" : Width of image named by file_path ● Special format "(new Image('file_path')).height" : Height of image named by file_path ● Special format "symbol.offset" : Position of the symbol "file_path" is the path of image file. "symbol" is @symbol in "Image", "File", "Var" tag. Endian is depended on @endian.

/ImagePackager/SourceTemplate/ : Zero, one or more elements

@id	ID of SourceTemplate tag. (Essential) This value is referred from /ImagePackager/OutputBinary/@source_template.
Source/text()	Template of source file. (Essential) The following tags can be filled in the template. <ul style="list-style-type: none"> ● \${Section} : Section name. The value of /ImagePackager/OutputBinary/@section ● \${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol ● \${Size} : Size of the binary file (bytes) ● \${BinaryData} : Binary data
SourceWithSection/text()	Template of source file, if a section was specified. (Optional) If SourceWithSection tag was not specified, the content specified by Source tag is used, even if a section was specified.
Header/text()	Template of header file. (Essential) The following tags can be filled in the template. <ul style="list-style-type: none"> ● \${include_define} : Macro name avoiding double include. The value of /ImagePackager/OutputBinary/OutputHeader/@include_define ● \${DeclareBinaryImageSymbol} : Declares of symbols in the binary data. The content of DeclareVariable/text() or DeclareAddress/text() and #define by @raw_image_alignment_symbol ● \${Variables} : #define list as variables ● \${Section} : Section name. The value of /ImagePackager/OutputBinary/@section ● \${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol ● \${Size} : Size of the binary file (bytes) ● \${StartAddress} : Start address of the binary data ● \${LastAddress} : Last address of the binary data
HeaderWithSection/text()	Template of header file, if a section was specified. (Optional) If HeaderWithSection tag was not specified, the content specified by Header tag is used, even if a section was specified.
DeclareVariable/text()	If @language="C", template filled at \${DeclareBinaryImageSymbol}. The following tags can be filled in the template. <ul style="list-style-type: none"> ● \${Section} : Section name. The value of /ImagePackager/OutputBinary/@section ● \${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol ● \${Size} : Size of the binary file (bytes)
DeclareAddress/text()	If @language="Binary", template filled at \${DeclareBinaryImageSymbol}. The following tags can be filled in the template. <ul style="list-style-type: none"> ● \${Section} : Section name. The value of /ImagePackager/OutputBinary/@section ● \${Symbol} : Variable name. The value of /ImagePackager/OutputBinary/@symbol ● \${StartAddress} : Start address of the binary data ● \${LastAddress} : Last address of the binary data

/ImagePackager/SuperClass: Zero, one or more elements

@id	ID of SuperClass tag. (Essential) This value is referred from /ImagePackager/OutputBinary/@super_class.
-----	---

/ImagePackager/SuperClass/OutputBinary : Zero or one

@endian	Default value of /ImagePackager/OutputBinary/@endian. This default value is "LittleEndian".
@raw_image_alignment	Default value of /ImagePackager/OutputBinary/@raw_image_alignment. This default value is 4.
@raw_stride_alignment	Default value of /ImagePackager/OutputBinary/@raw_stride_alignment. This default value is 1.
@raw_stride_alignment_4	Default value of /ImagePackager/OutputBinary/@raw_stride_alignment_4. This default value is "".

/ImagePackager/SuperClass/InputFiles/File : Zero, one or more elements

@path	Path of target file attached with SuperClass. Wildcard can be specified. (Essential) Example: path="*.jpg"
@alignment	Default value of /ImagePackager/InputFiles/File/@alignment. This default value is 4.

6.1.8 Basic forms of writing XML

This section describes basic forms of writing XML, XPath and # fragment that is data format specified with tools.

(1) XML

XML file is one of text file. You can edit it by text editor. This section describes basic forms of writing XML only.

XML declaration is written at the head of XML file like the following text. Character code set is generally encoding="UTF-8", if XML declaration was omitted. The following text is an example.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

A markup construct that begins with < and ends with >. Start-tags must be paired with end-tags. End-tag is appended with slash at head of start-tag's name. Target program ignores not supported tags and does not warn spelling mistakes. Difference between upper case and lower case is treated as different name.

```
<Root>
</Root>
```

XML tags must have tree structure. Also, there must be one root end tag. Data passing to target program are not changed, even if return character was replaced space or tab character or deleted between tags.

```
<Root>
    <LeafA></LeafA>
    <LeafB></LeafB>
</Root>
```

Start-tag and end-tag can be replaced to one tag that has a name appended slash at the tail of tag name. Data passing to target program are not changed, even if they were replaced like it.

```
<Root>
    <LeafA/>
    <LeafB/>
</Root>
```

Text can be written between start-tag and end-tag. The text is specified value depending on tag specification defined by target program. It is depended on tag specification whether different text between upper case and lower case is treated as same data or not same data.

```
<Root>
    <LeafA>ABC</LeafA>
    <LeafB>DEF</LeafB>
</Root>
```

Specified value can be written as attribute's value in start-tag. The attribute's value (right of equal) must be enclosed between " " or ' '. It is no different between " " and ' '. Specified value cannot be written in end-tag. Target program ignores not supported attributes and does not warn spelling mistakes. Difference between upper case and lower case is treated as different attribute's name. It is depended on target program whether different attribute's value between upper case and lower case is treated as same data or not same data.

```
<Root>
    <Leaf attribute="1" other="no"/>
</Root>
```

Same name tags can be written depended on tag's specification. 2nd tag and more tags are ignored, if specification of the tag can be allowed one tag only. It is depended on target program whether only one tag was allowed or not.

```

<Root>
    <Leaf attribute="1" other="no" />
    <Leaf attribute="2" other="no" />
    <Leaf attribute="3" other="yes" />
</Root>

```

(2) XPath

XPath is one of address that can point to a part of XML text. XPath is written by the forms like file path. This section describes basic forms only.

When there was XML text like the following text:

```

<Root>
    <LeafA>ABC</LeafA>
    <LeafB attribute="1" other="no" />
</Root>

```

The following text is an XPath that points to the position at the text value "ABC". Between XML nodes are split by slash character. "text()" points a text between tags. It is necessary to write "()" at the tail of "text". Difference between upper case and lower case is treated as different name. The following text is an example.

```
/Root/LeafA/text()
```

The following text is an XPath that points to the position at the attribute's value "1". It is necessary to write slash and "@" at the head of attribute's name.

```
/Root/LeafA/@attribute
```

It is step path that the head of XPath is not slash.

```
LeafB/@attribute
@attribute
```

(3) # fragment

and ID (fragment) can be written after specified file name or path depended on the specification of target program. The following text is an example.

```
Folder\File.xml#Leaf1
```

Fragment is compared with id attributes from all XML tags in the specified file. Difference between upper case and lower case is treated as different name. Path attached fragment points an XML tag that has matched id attribute's value. For example, "Folder\File.xml#Leaf1" points following "LeafA" tag in "Folder\File.xml" file.

```

<Root>
    <LeafA id="Leaf1">ABC</LeafA>
    <LeafB id="Leaf2">ABC</LeafB>
</Root>

```

6.2 Converting binary by ConvertBin

ConvertBin converts from binary file to C language array or S-record format (Motorola S-record).

Command	Description
BinToC	Converts from binary file to C language array. Variable name can be specified.
BinToSRec	Converts from binary file to S-record format. Comment, load address and execute address can be specified. Execute address is usually 0 for data binary.
SRecToBin	Converts from S-record format to binary file.

Window after RGA_Tools.vbs is double-clicked:

```
RGA Tools - Copyright(c) 2012-2015 Renesas Electronics Corporation
  1. Convert image format [RunImagePackager]
  2. Search error information [SearchErrorInformation]
  3. Convert to binary [ConvertBin]
Number or command >3
-----
((( [ConvertBin] )))
1. Binary to C language [BinToC]
2. Binary to S-record format [BinToSRec]
3. S-record format to binary [SRecToBin]
Number or command >
```

6.3 Creating image file by RawToBmp

RawToBmp creates a BMP image file from Raw data in frame buffer.

Window after RGA_Tools.vbs is double-clicked:

```

RGA Tools - Copyright(c) 2012-2015 Renesas Electronics Corporation

1. Convert image format [RunImagePackager]
2. Search error information [SearchErrorInformation]
3. Convert to binary [ConvertBin]
4. Convert to BMP file [RawToBmp]

Number or command >4
-----
Path of setting file >C:\Folder\RawToBmp.ini

```

Example of setting file:

```

RawPath = Image.bin
OutBmpPath = Image.bmp
Stride = 1600
Format = RGB565

```

Description of setting file:

Attribute Name	Description
RawPath	File path saved Raw data loaded from frame buffer
OutBmpPath	File path of output BMP file
Stride	Number of bytes of pixels having the same x coordinate in the previously below line.
Format	Pixel format. Any one of ARGB8888, RGB565, ARGB1555, ARGB4444, YCbCr422, A8, A4, A1. See Table 1-3.
ReadOffset	List of offsets that means byte order of reading 8 bytes. e.g.) 01234567(Little endian), 76543210(Big endian) "ReadOffset" cannot specify with YCbCr422 format.

7. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware

The latest version can be downloaded from the Renesas Electronics website.

RTK7921053C00000BE (RZ/A2M CPU board) User's Manual

The latest version can be downloaded from the Renesas Electronics website.

RTK79210XXB00000BE (RZ/A2M SUB board) User's Manual

The latest version can be downloaded from the Renesas Electronics website.

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C

The latest version can be downloaded from the ARM website.

ARM Cortex™-A9 Technical Reference Manual Revision: r4p1

The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification - Architecture version 2.0

The latest version can be downloaded from the ARM website.

ARM CoreLink™ Level 2 Cache Controller L2C-310 Technical Reference Manual Revision: r3p3

The latest version can be downloaded from the ARM website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

Integrated development environment e2studio User's Manual can be downloaded from the Renesas Electronics website.

The latest version can be downloaded from the Renesas Electronics website.

7.1 Reference Symbols

In the source file and the HTML document referring to this document, symbols (reference symbols) are written instead of numbers for identifying sections / figures / tables. The correspondence between reference symbols and sections / figures / tables is shown below.

Reference Symbols	Sections / figures / tables
Section_RGA_Initialize	4.4.2 Internal operation in initialize function
Section_RGA_Finalize	4.4.6 Finalize (*_Finalize Function)
Section_RGA_ImageFormat	6.1.5 File Formats in the Output Binary File
Section_byte_per_pixel_t	4.2.3 Figure of byte_per_pixel_t
Figure_RGA_PixelFormat	Figure 1-4 Structure of Pixel Format
Section_DefaultableFlags	4.4.4 Defaultable Flags
Section_RGA_ImagePackager	6.1 Image Format Conversion by ImagePackager
Section_RGA_IdentifyingImageFormat	4.4.3 Identifying Image Format
Figure_RGA_DrawImageChild	5.1.37 R_GRAPHICS_DrawImageChild

Revision History

Rev.	Date	Description	
		Page	Summary
1.11	Sep. 30, 2020	p16, p18	Figure 4.2 Layout correction.
		p43	Correction of errors. 2x3 matrix -> 3x3 matrix.
		p46	Correction of mistakes deleted. (6.6.5.14).
		p58	Add description to "5.4.1 List of Functions 5.4.12 - 14".
		p81	Added description of "@alpha_raw_image_width".
		p1	Added Restrictions This library supports only affine transformation.
1.10	May. 17, 2019	p13	Table 7.1. Operation Confirmation Conditions (1/2) Remove compiler option "-mthumb-interwork"
1.03	Apr. 15, 2019	p10	Modified Supported JPEG format target sentence from RZ/A1 to RZ/A2M
		p12	Changed the folder tree
		p12	Changed the way of installing RGA library to user project from copying to using
		p13	Updated the confirmed version of integrated development environment
		p46	Modified pixel format not previously supported
		p75	Modified the sentence from RZ/A1H to RZ/A2M
1.02	Dec.28, 2018	-	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity.

Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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.

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

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

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

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

Contact information

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