RZ/A2M Group

Video Display Controller and Sprite Engine Sample Driver

Introduction

This document describes the functional specification of Video Display Controller (VDC) and Sprite Engine driver for a RZ/A series RZ/A2M group MCU.

Target Device

RZ/A2M

Contents

1. Specifications

This driver uses the VDC built in RZ/A2M group to control video input and display.

The functions that this driver supports are listed below.

Table 1-1 VDC Driver Functions

Item	Function
Input video image specification	8-bit input conforming to ITU-R BT.656 standard (27 MHz, interlace signal) 8-bit input conforming to ITU-R BT.656 extended standard (27 MHz, progressive signal) * 8-bit input conforming to ITU-R BT.601 extended standard (27 MHz, interlace signal) * 8-bit input conforming to ITU-R BT.601 extended standard (54 MHz, progressive signal) * 16-bit input conforming to ITU-R BT.601 extended standard (13.5 MHz, interlace signal) * Digital pin input: YCbCr422, YCbCr444, RGB888, RGB666, and RGB565 video image
Video image recording	Storing the video image in the YCbCr422/YCbCr444/RGB565/RGB888 format at a rate of 1/1, 1/2, 1/4, or 1/8 field.
Video image quality adjustment	Contrast adjustment, brightness adjustment, horizontal noise reduction, black stretch, LTI/sharpness
Video image scaling/rotation	Vertical/horizontal scaling: 1/8 to 8 times 0, 90, 180, and 270 degree rotations and horizontal mirroring
Graphics planes	Graphics planes: 4 planes Supported pixel formats: RGB565, RGB888, ARGB1555, ARGB4444, ARGB8888, RGBA5551, RGBA8888, CLUT8, CLUT4, CLUT1, YCbCr422, YCbCr444
Graphics functions	Alpha blending in rectangular area (fade-in and fade-out functions are available.) Chroma-key Alpha blending in one pixel units
Output video image size	Video output size examples: XGA (1024x768), SVGA (800x600), WVGA (800x480), VGA (640x480), WQVGA (480x240), QVGA landscape (320x240), QVGA portrait (240x320)
Output video image format	Progressive video output • RGB888 (24-bit parallel output) • RGB666 (18-bit parallel output) • RGB565 (16-bit parallel output) • RGB888 (8-bit serial output)
Panel output adjustment	Panel brightness/contrast adjustment, RGB gamma correction, dither processing, output format conversion

Note: The ITU-R BT.656 and 601 standards do not include the description regarding the progressive signal. The ITU-R BT.601 standard does not include the description regarding the connection interface.

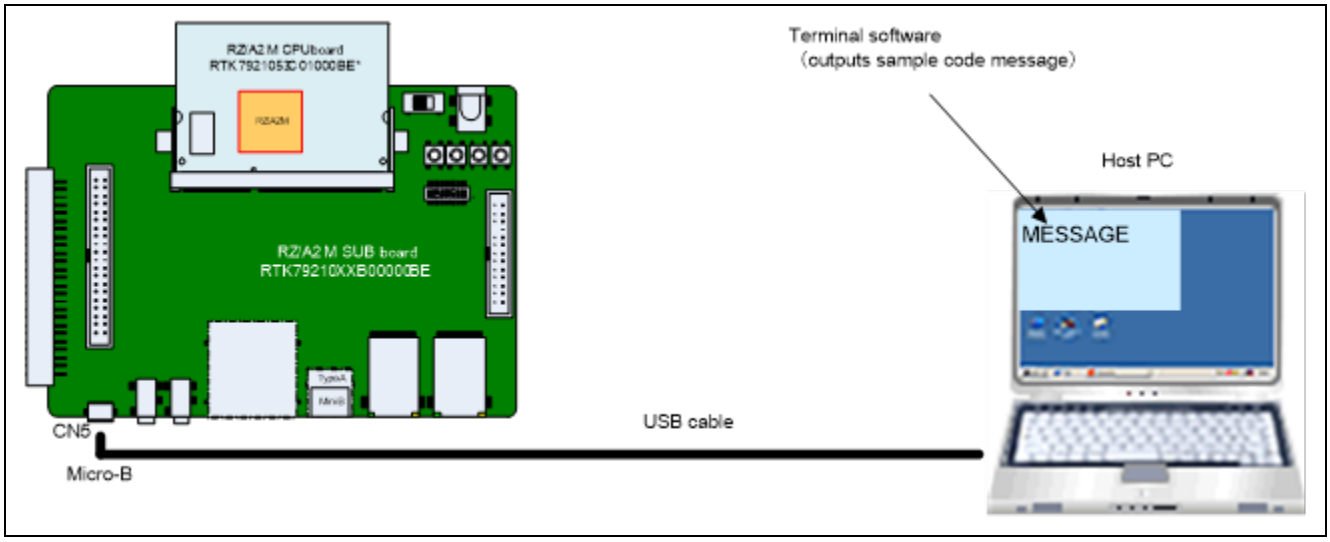


Figure 1.1 Operation check conditions

2. Operation Confirmation Conditions

The sample code of this application note has been confirmed to operate under the following conditions.

Table 2.1 Peripheral device used(1/2)

Peripheral device	Usage
MCU Used	RZ/A2M
Operating frequency[MHz] (Note)	CPU Clock (Iφ) : 528MHz Image processing clock (Gφ) : 264MHz Internal Bus Clock (Bφ) : 132MHz Peripheral Clock 1 (P1φ) : 66MHz Peripheral Clock 0 (P0φ) : 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	e2 studio V7.4.0
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 2.2 Peripheral device used(2/2)

Operation mode	Boot mode 3 (Serial Flash boot 3.3V)
Terminal software communication settings	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)	Serial flash memory allocated to SPI multi-I/O bus space (channel 0) Manufacturer : Macronix Inc. Model Name : MX25L51245GXD RL78/G1C (This device communications the host PC by convert USB Communication and Serial Communication.)

3. Reference Application Notes

The application notes related to this application note are shown below.

Nothing.

4. Hardware Description

4.1 Hardware Configuration

This driver is supposed to be an evaluation board for RZ / A2M

4.2 List of Pins to Be Used

Table 4-1 lists the pins to be used and describes their functionalities.

Table 4-1 Pins to Be Used and Their Function

Symbol	I/O	Function	RZ/A2M Evaluation Board connection
DV0_CLK	Input	External input clock 0	NC
DV0_VSYNC	Input	External input Vsync 0	NC
DV0_HSYNC	Input	External input Hsync 0	NC
DV0_DATA23~0	Input	External input video image data 0	NC
LCD0_CLK	Output	Panel clock 0	PJ_6
LCD0_DATA23~0	Output	Video image data 0 for panel	PB_5-0, PA_7-0, P8_0, PF_7-0, PH_2
LCD0_TCON6~0	Output	Control signal 0 for panel	PC_3(TCON4), PC_4(TCON3), P7_7(TCON0)
LCD0_EXTCLK	Input	Panel clock source 0	PJ_6
TXCLKOUTM/P	Output	LVDS clock output pins	P4_7, P4_6
TXOUT2M/P	Output	LVDS data output pins	P4_5, P4_4
TXOUT1M/P	Output	LVDS data output pins	P4_3, P4_2
TXOUT0M/P	Output	LVDS data output pins	P4_1, P4_0

5. Software Description

5.1 File Configuration

Table 5-1, Table 5-2 lists the files that make up this driver.

Table 5-1 VDC Driver File Configuration

File Name	Description
r_vdc.c	VDC driver API function Source file defining the VDC driver's API functions
r_vdc.h	VDC driver API definitions Header file describing the prototypes of the VDC driver API functions and the parameters that are defined as APIs
r_vdc_check_parameter.c	VDC driver parameter check processing Source file describing the VDC driver's parameter check processing
r_vdc_check_parameter.h	VDC driver parameter check definitions Header file describing the prototypes of the VDC driver's parameter check functions
r_vdc_interrupt.c	VDC driver interrupt related processing Source file describing the VDC interrupt related setup processing and interrupt service routines
r_vdc_register.c	VDC driver register setup processing Source file describing the VDC register setup processing
r_vdc_register.h	VDC driver register setup definitions Header file describing the prototypes of the VDC register setup processing functions and the structures of the register address tables
r_vdc_register_address.c	VDC driver register address table File describing the table containing the VDC's register addresses
r_vdc_shared_param.c	VDC driver shared parameter processing Source file describing the setup and retrieval processing for the parameters that are shared inside the VDC driver
r_vdc_shared_param.h	VDC driver shared parameter definitions Header file describing the prototypes of VDC driver shared parameter setup/retrieval processing functions
r_vdc_user.h	VDC driver user-defined header Header file defining compile switches and constants that can by statically edited by the user

Table 5-2 SPEA Driver File Configuration

ファイル名	説明
r_spea.c	SPEA driver API function Source file defining the VDC driver's API functions
r_spea.h	SPEA driver API definitions Header file describing the prototypes of the VDC driver API functions and the parameters that are defined as APIs
r_spea_check_parameter.c	SPEA driver parameter check processing Source file describing the VDC driver's parameter check processing
r_spea_check_parameter.h	SPEA driver parameter check definitions Header file describing the prototypes of the VDC driver's parameter check functions
r_spea_register.c	SPEA driver register setup processing Source file describing the VDC register setup processing

r_spea_register.h	SPEA driver register setup definitions Header file describing the prototypes of the VDC register setup processing functions and the structures of the register address tables
r_spea_register_address.c	SPEA driver register address table File describing the table containing the SPEA's register addresses
r_spea_user.h	SPEA driver user-defined header Header file defining compile switches and constants that can by statically edited by the user

This driver also references the following external files.

Table 5-3 External Files Referenced by the VDC and SPEA Driver

File Name	Description
r_typedefs.h	Basic type definition header Header file defining the basic types
iodefine.h	I/O definition header Header file containing the I/O definitions

5.2 Enumeration Type Definitions

The enumeration type definitions are given below. See 5.4 for the error codes.

(1) vdc_channel_t

vdc_channel_t is an enumeration type for representing the VDC channels.

```
typedef enum
{
    VDC_CHANNEL_0 = 0,
    VDC_CHANNEL_NUM
} vdc_channel_t;
```

Enumeration constant	Value	Description
VDC_CHANNEL_0	0	Channel 0
VDC_CHANNEL_NUM	1	Number of channels

(2) vdc_onoff_t

vdc_onoff_t is an enumeration type for representing ON or OFF.

```
typedef enum
{
    VDC_OFF    = 0,
    VDC_ON     = 1
} vdc_onoff_t;
```

Enumeration constant	Value	Description
VDC_OFF	0	OFF
VDC_ON	1	ON

(3) vdc_edge_t

vdc_edge_t is an enumeration type for representing the edge of a signal.

```
typedef enum
{
    VDC_EDGE_RISING    = 0,
    VDC_EDGE_FALLING   = 1
} vdc_edge_t;
```

Enumeration constant	Value	Description
VDC_EDGE_RISING	0	Rising edge
VDC_EDGE_FALLING	1	Falling edge

(4) vdc_sig_pol_t

vdc_sig_pol_t is an enumeration type for representing the polarity of a signal.

```
typedef enum
{
    VDC_SIG_POL_NOT_INVERTED   = 0,
    VDC_SIG_POL_INVERTED       = 1
} vdc_sig_pol_t;
```

Enumeration constant	Value	Description
VDC_SIG_POL_NOT_INVERTED	0	Not inverted
VDC_SIG_POL_INVERTED	1	Inverted

(5) vdc_scaling_type_t

vdc_scaling_type_t is an enumeration type for representing the types of scalers.

```
typedef enum
{
    VDC_SC_TYPE_SC0 = 0,
    VDC_SC_TYPE_NUM
} vdc_scaling_type_t;
```

Enumeration constant	Value	Description
VDC_SC_TYPE_SC0	0	Scaler 0
VDC_SC_TYPE_NUM	1	Number of scaler types

(6) vdc_graphics_type_t

vdc_graphics_type_t is an enumeration type for representing the types of graphics.

```
typedef enum
{
    VDC_GR_TYPE_GR0 = 0,
    VDC_GR_TYPE_GR2,
    VDC_GR_TYPE_GR3,
    VDC_GR_TYPE_NUM
} vdc_graphics_type_t;
```

Enumeration constant	Value	Description
VDC_GR_TYPE_GR0	0	Graphics 0
VDC_GR_TYPE_GR2	1	Graphics 2
VDC_GR_TYPE_GR3	2	Graphics 3
VDC_GR_TYPE_NUM	3	Number of graphics

(7) vdc_layer_id_t

vdc_layer_id_t is an enumeration type for representing the layer ID.

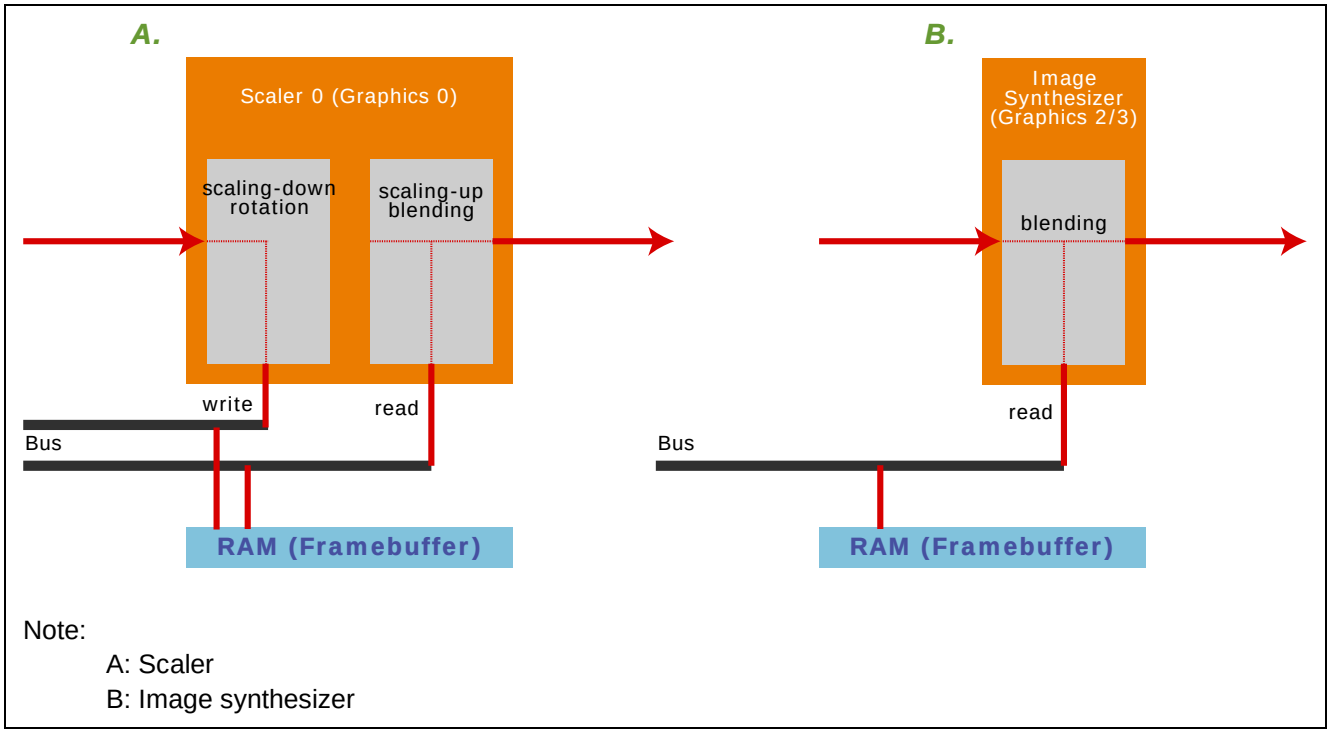
```
typedef enum
{
    VDC_LAYER_ID_ALL       = -1,
    VDC_LAYER_ID_0_WR      = (VDC_SC_TYPE_SC0 + 0),
    VDC_LAYER_ID_0_RD      = (VDC_SC_TYPE_NUM + VDC_GR_TYPE_GR0),
    VDC_LAYER_ID_2_RD      = (VDC_SC_TYPE_NUM + VDC_GR_TYPE_GR2),
    VDC_LAYER_ID_3_RD      = (VDC_SC_TYPE_NUM + VDC_GR_TYPE_GR3),
    VDC_LAYER_ID_NUM       = (VDC_SC_TYPE_NUM + VDC_GR_TYPE_NUM)
} vdc_layer_id_t;
```

Enumeration constant	Value	Description
VDC_LAYER_ID_ALL	-1	All layers
VDC_LAYER_ID_0_WR	0	Write process for layer 0
VDC_LAYER_ID_0_RD	1	Read process for layer 0
VDC_LAYER_ID_2_RD	2	Read process for layer 2
VDC_LAYER_ID_3_RD	3	Read process for layer 3
VDC_LAYER_ID_NUM	4	Number of layer IDs

The VDC has three layers (graphics 0 (layer 0), graphics 2 (layer 2), and graphics 3 (layer 3)). Of the VDC's internal blocks, scaler 0 corresponds to graphics 0, and the two image synthesizer blocks correspond to graphics 2 and graphics 3, respectively.

The scaler can be divided into the former stage for writing the input data and the latter stage for reading data from memory (see A in Figure 5-1). The former stage of the scaler performs scale-down and rotation processing on the input image data and writes the results into memory. The latter stage of the scaler performs scale-up processing on the data that is read from memory. Different layer IDs, which are defined in the enumeration type vdc_layer_id_t, are assigned to the memory write processing and read processing for the same layer.

The image data read from memory and the image data from the lower layer can be blended in the image synthesizer (see B in Figure 5-1).



Note:
A: Scaler
B: Image synthesizer

Figure 5-1 Memory Write/Read Processing

The results of image synthesis among layers look like as shown in Figure 5-2. Layer 0 is the bottom layer and layer 3 is the top layer.

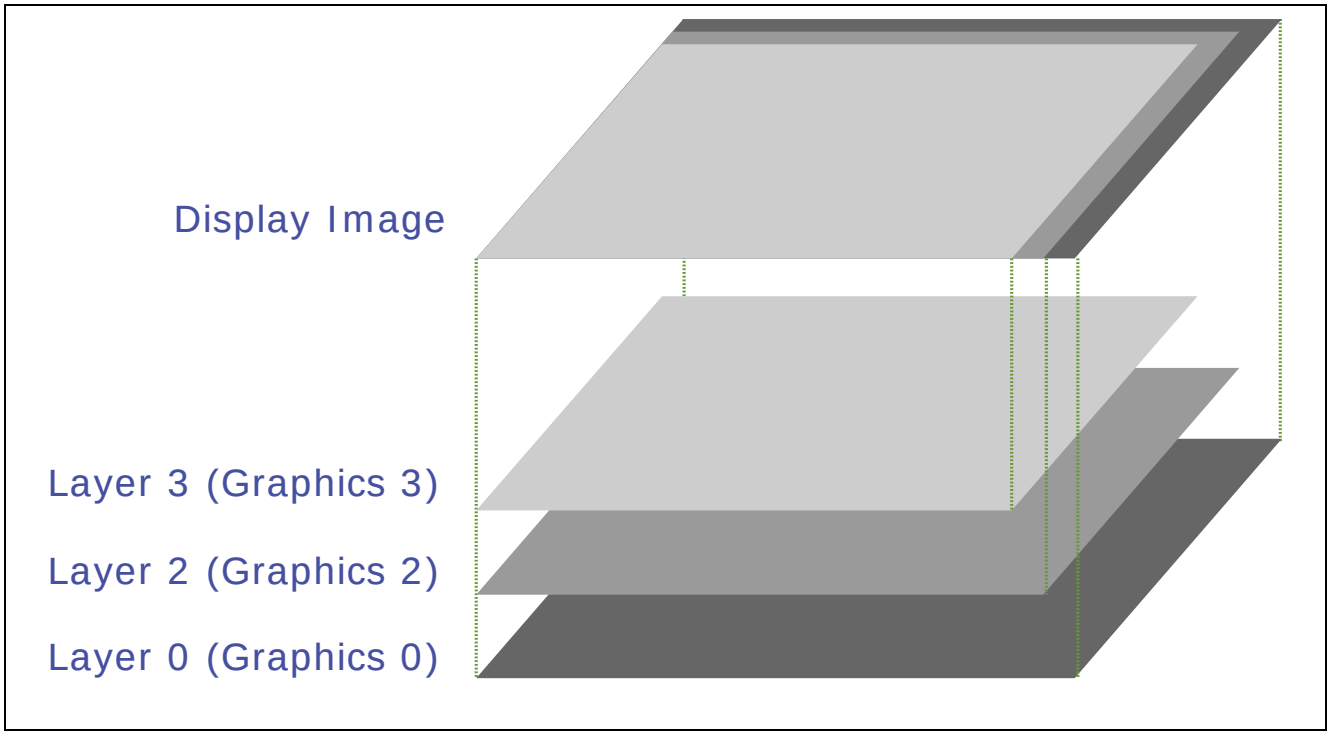


Figure 5-2 Layers and Image Synthesis

(8) vdc_int_type_t

vdc_int_type_t is an enumeration type for representing the types of VDC interrupts.

```
typedef enum
{
    VDC_INT_TYPE_S0_VI_VSYNC = 0,
    VDC_INT_TYPE_S0_LO_VSYNC,
    VDC_INT_TYPE_S0_VSYNCERR,
    VDC_INT_TYPE_VLINE,
    VDC_INT_TYPE_S0_VFIELD,
    VDC_INT_TYPE_IV1_VBUFERR,
    VDC_INT_TYPE_IV3_VBUFERR,
    VDC_INT_TYPE_IV5_VBUFERR,
    VDC_INT_TYPE_IV6_VBUFERR,
    VDC_INT_TYPE_S0_WLINE,
    VDC_INT_TYPE_NUM
} vdc_int_type_t;
```

Enumeration constant	Value	Description
VDC_INT_TYPE_S0_VI_VSYNC	0	Vsync signal input to scaler 0
VDC_INT_TYPE_S0_LO_VSYNC	1	Vsync signal output from scaler 0
VDC_INT_TYPE_S0_VSYNCERR	2	Missing Vsync signal for scaler 0
VDC_INT_TYPE_VLINE	3	Specified line signal for panel output in graphics 3
VDC_INT_TYPE_S0_VFIELD	4	Field end signal for recording function in scaler 0
VDC_INT_TYPE_IV1_VBUFERR	5	Frame buffer write overflow signal for scaler 0
VDC_INT_TYPE_IV3_VBUFERR	6	Frame buffer read underflow signal for graphics 0
VDC_INT_TYPE_IV5_VBUFERR	7	Frame buffer read underflow signal for graphics 2
VDC_INT_TYPE_IV6_VBUFERR	8	Frame buffer read underflow signal for graphics 3
VDC_INT_TYPE_S0_WLINE	9	Write specification line signal input to scaling-down control block in scaler 0
VDC_INT_TYPE_NUM	10	Number of VDC interrupt types

(9) vdc_gr_disp_sel_t

vdc_gr_disp_sel_t is an enumeration type for representing the graphics display modes.

```
typedef enum
{
    VDC_DISPSEL_IGNORED    = -1,
    VDC_DISPSEL_BACK       = 0,
    VDC_DISPSEL_LOWER      = 1,
    VDC_DISPSEL_CURRENT    = 2,
    VDC_DISPSEL_BLEND      = 3,
    VDC_DISPSEL_NUM        = 4
} vdc_gr_disp_sel_t;
```

Enumeration constant	Value	Description
VDC_DISPSEL_IGNORED	-1	Ignored, no change made
VDC_DISPSEL_BACK	0	Background color display
VDC_DISPSEL_LOWER	1	Lower-layer graphics display
VDC_DISPSEL_CURRENT	2	Current graphics display

VDC_DISPSEL_BLEND	3	Blended display of lower-layer graphics and current graphics
VDC_DISPSEL_NUM	4	Number of graphics display modes

(10) vdc_imgimprv_id_t

vdc_imgimprv_id_t is an enumeration type for representing the image quality improvers.

```
typedef enum
{
    VDC_IMG_IMPRV_0 = 0,
    VDC_IMG_IMPRV_NUM
} vdc_imgimprv_id_t;
```

Enumeration constant	Value	Description
VDC_IMG_IMPRV_0	0	Image quality improver 0
VDC_IMG_IMPRV_NUM	1	Number of image quality improvers

(11) spea_onoff_t

spea_onoff_t is an enumeration type for representing ON or OFF. It is used to set ON / OFF of SPEA's Window.

```
typedef enum
{
    SPEA_OFF    = 0,
    SPEA_ON     = 1
} spea_onoff_t;
```

Enumeration constant	Value	Description
SPEA_OFF	0	OFF
SPEA_ON	1	ON

(12) rle_onoff_t

rle_onoff_t is an enumeration type for representing ON or OFF. It is used to set ON / OFF of RLE function.

```
typedef enum
{
    RLE_OFF    = 0,
    RLE_ON     = 1
} rle_onoff_t;
```

Enumeration constant	Value	Description
RLE_OFF	0	OFF
RLE_ON	1	ON

(13) spea_window_id_t

spea_window_id_t is an enumerated type representing SPEA's Window ID. A maximum of 15 windows can be created per SPEA layer.

```
typedef enum
{
    WINDOW_00 = 0,
    WINDOW_01,
    WINDOW_02,
    WINDOW_03,
    WINDOW_04,
    WINDOW_05,
    WINDOW_06,
    WINDOW_07,
    WINDOW_08,
    WINDOW_09,
    WINDOW_10,
    WINDOW_11,
    WINDOW_12,
    WINDOW_13,
    WINDOW_14,
    WINDOW_15,
    WINDOW_NUM
} spea_window_id_t;
```

Enumeration constant	Value	Description
WINDOW_00	0	SPEA の Window ID (Lower layer)
WINDOW_01	1	SPEA の Window ID
WINDOW_02	2	SPEA の Window ID
WINDOW_03	3	SPEA の Window ID
WINDOW_04	4	SPEA の Window ID
WINDOW_05	5	SPEA の Window ID
WINDOW_06	6	SPEA の Window ID
WINDOW_07	7	SPEA の Window ID
WINDOW_08	8	SPEA の Window ID
WINDOW_09	9	SPEA の Window ID
WINDOW_10	10	SPEA の Window ID
WINDOW_11	11	SPEA の Window ID
WINDOW_12	12	SPEA の Window ID
WINDOW_13	13	SPEA の Window ID
WINDOW_14	14	SPEA の Window ID
WINDOW_15	15	SPEA の Window ID (Top layer)

5.3 Common Structure Definitions

(1) vdc_period_rect_t

vdc_period_rect_t is a structure for representing the horizontal/vertical timing of the VDC signals.

```
typedef struct
{
    uint16_t    vs;
    uint16_t    vw;
    uint16_t    hs;
    uint16_t    hw;
} vdc_period_rect_t;
```

Type Member Name	Description
uint16_t vs	Vertical signal start position from the reference signal (lines)
uint16_t vw	Vertical signal width (lines)
uint16_t hs	Horizontal signal start position from the reference signal (clock cycles)
uint16_t hw	Horizontal signal width (clock cycles)

The horizontal/vertical timings in the vdc_period_rect_t structure are represented as a rectangle area as shown in Figure 5-3.

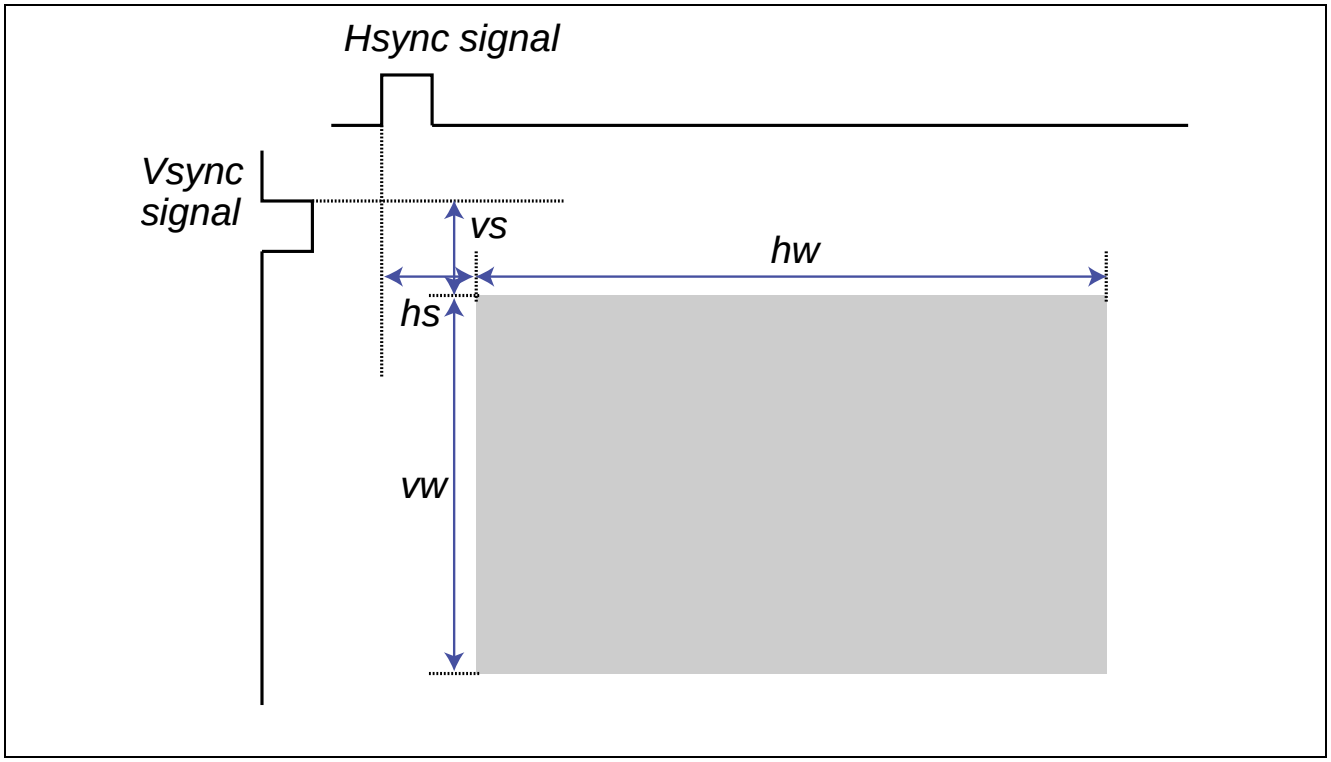


Figure 5-3 Rectangle Representing the Horizontal and Vertical Timings

5.4 Error Codes

Table 5-4 shows a list of error codes of the VDC driver.

Table 5-4 List of VDC Driver Error Codes

Error Code	Value	Description (Error Type)
VDC_OK	0	Normal termination
VDC_ERR_PARAM_CHANNEL	1	Invalid channel error (parameter error) An illegal channel is specified.
VDC_ERR_PARAM_LAYER_ID	2	Invalid layer ID error (parameter error) An illegal layer ID is specified.
VDC_ERR_PARAM_NULL	3	NULL specification error (parameter error) NULL is specified for a required parameter.
VDC_ERR_PARAM_BIT_WIDTH	4	Bit width error (parameter error) A value exceeding the possible bit width is specified.
VDC_ERR_PARAM_UNDEFINED	5	Undefined parameter error (parameter error) A value that is not defined in the specification is specified.
VDC_ERR_PARAM_EXCEED_RANGE	6	Out-of-value-range error (parameter error) The specified parameter value is beyond the value range defined in the specification.
VDC_ERR_PARAM_CONDITION	7	Unauthorized condition error (parameter error) A parameter is specified under conditions that are not authorized by the specification.
VDC_ERR_IF_CONDITION	8	Interface condition error (interface error) An API function is called under unauthorized conditions.
VDC_ERR_RESOURCE_CLK	9	Clock resource error (resource error) No panel clock is set up.
VDC_ERR_RESOURCE_VSYNC	10	Vertical sync signal resource error (resource error) No vertical sync signal is set up.
VDC_ERR_RESOURCE_INPUT	11	Input signal resource error (resource error) No video image input is set up.
VDC_ERR_RESOURCE_OUTPUT	12	Output resource error (resource error) No display output is set up.
VDC_ERR_RESOURCE_LVDS_CLK	13	LVDS clock resource error (resource error) No LVDS clock is set up
VDC_ERR_RESOURCE_LAYER	14	Layer resource error (resource error) The specified layer is under unavailable conditions.
VDC_ERR_NUM	15	VDC driver error code number

Table 5-5 shows a list of error codes of the VDC driver.

Table 5-5 List of SPEA Driver Error Codes

Error Code	Value	Description (Error Type)
SPEA_OK	0	Normal termination
SPEA_ERR_PARAM_LAYER_ID	1	Invalid layer ID error (parameter error) An illegal layer ID is specified.
SPEA_ERR_PARAM_NULL	2	NULL specification error (parameter error) NULL is specified for a required parameter.
SPEA_ERR_PARAM	3	Unauthorized condition error (parameter error) A parameter is specified under conditions that are not authorized by the specification.

5.5 User Custom Parameters

Parameters that can statically be changed by the user are defined in "r_vdc_user.h" for this driver.

(1) Enumeration type vdc_colcnv_rgb_ycbcr_t

vdc_colcnv_rgb_ycbcr_t is an enumeration type for representing the color matrix values. It is referenced by the VDC driver when converting GBR signals to YCbCr signals. The default values are the standard values that are described in the hardware manual.

```
typedef enum
{
    VDC_COLORCONV_Y_R  = (77u),
    VDC_COLORCONV_Y_G  = (150u),
    VDC_COLORCONV_Y_B  = (29u),
    VDC_COLORCONV_CB_R = (2005u),
    VDC_COLORCONV_CB_G = (1963u),
    VDC_COLORCONV_CB_B = (128u),
    VDC_COLORCONV_CR_R = (128u),
    VDC_COLORCONV_CR_G = (1941u),
    VDC_COLORCONV_CR_B = (2027u)
} vdc_colcnv_rgb_ycbcr_t;
```

Enumeration constant	Value	Description
VDC_COLORCONV_Y_R	77u	Cr/R Signal Gain Adjustment for Y/G Signal Output(0.299)
VDC_COLORCONV_Y_G	150u	Y/G Signal Gain Adjustment for Y/G Signal Output(0.587)
VDC_COLORCONV_Y_B	29u	Cb/B Signal Gain Adjustment for Y/G Signal Output(0.114)
VDC_COLORCONV_CB_R	2005u	Cr/R Signal Gain Adjustment for Cb/B Signal Output(-0.169)
VDC_COLORCONV_CB_G	1963u	Y/G Signal Gain Adjustment for Cb/B Signal Output(-0.331)
VDC_COLORCONV_CB_B	128u	Cb/B Signal Gain Adjustment for Cb/B Signal Output(0.500)
VDC_COLORCONV_CR_R	128u	Cr/R Signal Gain Adjustment for Cr/R Signal Output(0.500)
VDC_COLORCONV_CR_G	1941u	Y/G Signal Gain Adjustment for Cr/R Signal Output(-0.419)
VDC_COLORCONV_CR_B	2027u	Cb/B Signal Gain Adjustment for Cr/R Signal Output(-0.081)

Note: The values are represented by 2's complement of the 11-bit values.
(-1024 ~ +1023[LSB]、256[LSB] = 1.0[time])

(2) Enumeration type vdc_colcnv_ycbcr_rgb_t

vdc_colcnv_ycbcr_rgb_t is an enumeration type for representing the color matrix values. It is referenced by the VDC driver when converting YCbCr signals to GBR signals. The default values are the standard values that are described in the hardware manual.

```
typedef enum
{
    VDC_COLORCONV_G_Y  = (256u),
    VDC_COLORCONV_G_CB = (1960u),
    VDC_COLORCONV_G_CR = (1865u),
    VDC_COLORCONV_B_Y  = (256u),
    VDC_COLORCONV_B_CB = (454u),
    VDC_COLORCONV_B_CR = (0u),
    VDC_COLORCONV_R_Y  = (256u),
    VDC_COLORCONV_R_CB = (0u),
    VDC_COLORCONV_R_CR = (359u)
} vdc_colcnv_ycbcr_rgb_t;
```

Enumeration constant	Value	Description
VDC_COLORCONV_G_Y	256u	Cr/R Signal Gain Adjustment for Y/G Signal Output(1.000)
VDC_COLORCONV_G_CB	1960u	Y/G Signal Gain Adjustment for Y/G Signal Output(-0.344)
VDC_COLORCONV_G_CR	1865u	Cb/B Signal Gain Adjustment for Y/G Signal Output(-0.714)
VDC_COLORCONV_B_Y	256u	Cr/R Signal Gain Adjustment for Cb/B Signal Output(1.000)
VDC_COLORCONV_B_CB	454u	Y/G Signal Gain Adjustment for Cb/B Signal Output(1.772)
VDC_COLORCONV_B_CR	0u	Cb/B Signal Gain Adjustment for Cb/B Signal Output(0.000)
VDC_COLORCONV_R_Y	256u	Cr/R Signal Gain Adjustment for Cr/R Signal Output(1.000)
VDC_COLORCONV_R_CB	0u	Y/G Signal Gain Adjustment for Cr/R Signal Output(0.000)
VDC_COLORCONV_R_CR	359u	Cb/B Signal Gain Adjustment for Cr/R Signal Output(1.402)

Note: The values are represented by 2's complement of the 11-bit values.
(-1024 ~ +1023[LSB]、256[LSB] = 1.0[time])

(3) Constant definitions

The constants are described below.

Constant	Value	Description
VDC_COLORCONV_DC_OFFSET	128u	Offset (DC) adjustment values for the Y/G, B, R signals in a color matrix Unsigned (0 (-128) to 255 (+127), 128[LSB] = 0) Referenced by the VDC driver when setting up the color matrix.
VDC_COLORCONV_1TIMES_GAIN	256u	1.0[time] gain value for the color matrix -1024 to +1023[LSB], 256[LSB] = 1.0[time] Referenced by the VDC driver when converting YCbCr signals to YCbCr signals and when converting GBR signals to GBR signals.
VDC_GAM_GAIN_ADJ_NUM	32u	The number of the gamma correction gain coefficient for each signal
VDC_GAM_START_TH_NUM	31u	The number of the gamma correction start threshold for each signal

5.6 Compile Switch

The following compile switch is defined in "r_vdc_user.h" for this driver.

Table 5-6 Compile Switch

Compile Switch	Description
R_VDC_CHECK_PARAMETERS	Enabling this definition causes the parameter check of the VDC driver API functions when they are called. If an error is found as the result of the parameter check, an error code indicating a parameter error is returned. See 5.4 for the error codes.

5.7 Restrictions

(1) Reserved words

The prefixes listed below are appended to the symbols such as function and variable names to be used for this driver to distinguish the driver program from other programs. Do not use in your program any names that begin with the following symbols, regardless of whether they are in upper or lower case:

- R_VDC
- VDC
- R_SPEA
- SPEA

(2) Register update

Any updates on the settings of most of the VDC registers are reflected on the rising edge of the vertical sync signal. Consequently, a time equivalent to up to 1 cycle of the vertical sync signal will be taken for the setting of a value to be reflected.

(3) Reentrancy

The APIs of this driver are not reentrant. The driver is likely to behave in an unexpected manner if one of its APIs is called by two or more tasks or interrupt processing routines asynchronously. Great care must be exercised with respect to the calling program of this driver and the call timing.

(4) Register accesses

This driver does not provide the user with any means of accessing all of the VDC registers. Some VDC registers are automatically set up by the driver itself.

5.8 VDC Internal Module Configuration

Figure 5-4 shows the internal modules of the VDC and shows data flow.

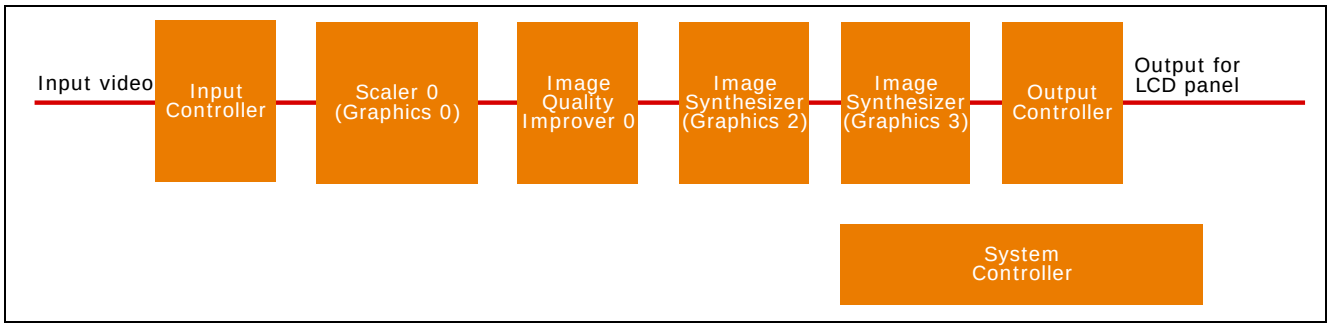


Figure 5-4 VDC Internal Module Configuration

The VDC in the RZ/A2M is made up of 6 blocks.

1. Input Controller:
   Subjects the signals to synchronization adjustment and adjusts the input image signals.
2. Scaler 0: Graphics 0
   Scaling and rotation of the input video image
3. Image Quality Improver 0:
   Image quality improvement, color conversion through color matrix function
4. Image Synthesizer: Graphics 2, graphics 3
   Synthesis of graphics planes
5. Output Controller:
   Output image adjustment, output format conversion, control signal output for TFT-LCD panel
6. System Controller:
   Interrupt control, panel clock control

6. Functions Reference

The API functions of the VDC driver are listed in Table 6-1.

Table 6-1 List of API Functions

Function name	Section	Outline
R_VDC_Initialize	6.1	VDC driver initialization
R_VDC_Terminate	6.2	VDC driver termination
R_VDC_VideoInput	6.3	Video input setup
R_VDC_SyncControl	6.4	Synchronization control setup
R_VDC_DisplayOutput	6.5	Display output setup
R_VDC_CallbackISR	6.6	Interrupt callback setup
R_VDC_WriteDataControl	6.7	Data write control processing
R_VDC_ChangeWriteProcess	6.8	Data write change processing
R_VDC_ReadDataControl	6.9	Data read control processing
R_VDC_ChangeReadProcess	6.10	Data read change processing
R_VDC_StartProcess	6.11	Data write/read start processing
R_VDC_StopProcess	6.12	Data write/read stop processing
R_VDC_ReleaseDataControl	6.13	Data write/read control release processing
R_VDC_VideoNoiseReduction	6.14	Noise reduction setup
R_VDC_ImageColorMatrix	6.15	Color matrix setup
R_VDC_ImageEnhancement	6.16	Image enhancement processing
R_VDC_ImageBlackStretch	6.17	Black stretch setup
R_VDC_AlphaBlending	6.18	Alpha blending setup
R_VDC_AlphaBlendingRect	6.19	Rectangle alpha blending setup
R_VDC_Chromakey	6.20	Chroma key setup
R_VDC_CLUT	6.21	CLUT setup
R_VDC_DisplayCalibration	6.22	Display calibration processing
R_VDC_GammaCorrection	6.23	Gamma correction setup
R_VDC_GetISR	6.24	Interrupt service routine acquisition processing

The API functions of the SPEA driver are listed in Table 6-2.

Table 6-2 List of API Functions

Function name	Section	Outline
R_SPEA_WindowOffset	6.25	Setting the coordinate offset of the window
R_SPEA_SetWindow	6.26	Setting Window Parameters
R_SPEA_WindowUpdate	6.27	Window parameter update request
R_RLE_SetWindow	6.28	Setting the RLE parameter
R_RLE_WindowUpdate	6.29	RLE parameter update request

6.1 R_VDC_Initialize

Synopsis	VDC driver initialization
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_Initialize(const vdc_channel_t ch, const vdc_init_t * const param, void (* const init_func)(uint32_t), const uint32_t user_num);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_init_t * param: Initialization parameter • void (* init_func)(uint32_t): Pointer to a user-defined function Specify the user-implemented function that is to be executed together with the VDC driver initialization processing. Within the API function R_VDC_Initialize, this function is called before the VDC registers are set up. user_num is used as the argument when the function is called. Specify '0' when this function is not required. • uint32_t user_num: User defined number Specify the argument to be passed to the user-defined function init_func. This parameter is ignored if '0' is specified as the user-defined function.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error

Details

(1) Function

This function initializes the VDC driver and performs the following associated processing:

- Initializes the VDC driver's internal variables.
- Calls the user-defined function specified in init_func.
- Sets up and enables the VDC's panel clock.
- Disables all the VDC interrupts.

(2) Use conditions

The following steps of processing need to be performed before the VDC driver is started:

- Supply of a clock to the VDC modules
- Setup of VDC-related interrupts (interrupt service routine, interrupt priority)
- Setup of VDC-related I/O ports
- Environment-specific setup necessary for the LCD panel and video input

Execute the above-listed steps except for setup of I/O ports for the LCD panel output before calling this function or implement a function that performs the above-listed steps and specify it in init_func as a user-defined function.

(3) Parameter details

The members of the vdc_init_t structure is described below.

```
typedef struct
{
    vdc_panel_clksel_t     panel_icksel;
    vdc_panel_clk_dcdr_t   panel_dcdr;
    const vdc_lvds_t     * lvds;
} vdc_init_t;
```

Type Member Name	Description
vdc_panel_clksel_t panel_icksel	Panel clock select • VDC_PANEL_ICKSEL_IMG_DV: Frequency-divided video image clock (DV_CLK) • VDC_PANEL_ICKSEL_EXT_0: Frequency-divided external clock 0 (LCD0_EXTCLK) • VDC_PANEL_ICKSEL_PERI: Frequency-divided peripheral clock 1 (P1φ)
vdc_panel_clk_dcdr_t panel_dcdr	Clock frequency division ratio • VDC_PANEL_CLKDIV_1_1: 1/1 • VDC_PANEL_CLKDIV_1_2: 1/2 • VDC_PANEL_CLKDIV_1_3: 1/3 • VDC_PANEL_CLKDIV_1_4: 1/4 • VDC_PANEL_CLKDIV_1_5: 1/5 • VDC_PANEL_CLKDIV_1_6: 1/6 • VDC_PANEL_CLKDIV_1_7: 1/7 • VDC_PANEL_CLKDIV_1_8: 1/8 • VDC_PANEL_CLKDIV_1_9: 1/9 • VDC_PANEL_CLKDIV_1_12: 1/12 • VDC_PANEL_CLKDIV_1_16: 1/16 • VDC_PANEL_CLKDIV_1_24: 1/24 • VDC_PANEL_CLKDIV_1_32: 1/32
const vdc_lvds_t * lvds	LVDS-related parameter Specify NULL if this parameter is not required.

The members of the vdc_lvds_t structure is described below.

```
typedef struct
{
    vdc_lvds_in_clk_sel_t   lvds_in_clk_sel;
    vdc_lvds_ndiv_t         lvds_idiv_set;    /* Not use */
    uint16_t                lvdspll_tst;      /* Not use */
    vdc_lvds_ndiv_t         lvds_odiv_set;
    vdc_channel_t           lvds_vdc_sel;
    uint16_t                lvdspll_fd;
    uint16_t                lvdspll_rd;
    vdc_lvds_pll_nod_t      lvdspll_od;       /* Not use */
} vdc_lvds_t;
```

Type Member Name	Description
vdc_lvds_in_clk_sel_t lvds_in_clk_sel	Clock input to frequency divider 1 • VDC_LVDS_INCLK_SEL_DV_0: DV0_CLK0 • VDC_LVDS_INCLK_SEL_EXT_0: LCD0_EXTCLK • VDC_LVDS_INCLK_SEL_PERI: P1φ
vdc_lvds_ndiv_t lvds_idiv_set	Frequency dividing value (NIDIV) for frequency divider 1(Not use) • VDC_LVDS_NDIV_1: NIDIV = 1 • VDC_LVDS_NDIV_2: NIDIV = 2 • VDC_LVDS_NDIV_4: NIDIV = 4
uint16_t lvdspll_tst	Internal parameter setting for LVDS PLL (Not use)
vdc_lvds_ndiv_t lvds_odiv_set	Frequency dividing value (NODIV) for frequency divider 2 • VDC_LVDS_NDIV_1: NODIV = 1 • VDC_LVDS_NDIV_2: NODIV = 2 • VDC_LVDS_NDIV_4: NODIV = 4
vdc_channel_t lvds_vdc_sel	Channel select in video display controller 5 whose data is to be output through • VDC_CHANNEL_0
uint16_t lvdspll_fd	Frequency dividing value (NFD) for the feedback frequency in the LVDS PLL NRD = lvdspll_fd + 1 NFD = lvdspll_fd (22 ~ 62)
uint16_t lvdspll_rd	Frequency dividing value (NRD) for the input frequency in the LVDS PLL NRD = lvdspll_rd + 1 lvdspll_rd (0 ~ 7)
vdc_lvds_pll_nod_t lvdspll_od	Frequency dividing value (NOD) for the output frequency in the LVDS PLL(Not use) • VDC_LVDS_PLL_NOD_1: NOD = 1 • VDC_LVDS_PLL_NOD_2: NOD = 2 • VDC_LVDS_PLL_NOD_4: NOD = 4 • VDC_LVDS_PLL_NOD_8: NOD = 8

6.2 R_VDC_Terminate

Synopsis	VDC driver termination
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_Terminate(const vdc_channel_t ch, void (* const quit_func)(uint32_t), const uint32_t user_num);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • void (* quit_func)(uint32_t): Pointer to a user-defined function Specify the user-implemented function that is to be executed together with the VDC driver termination processing. Within the API function R_VDC_Terminate, this function is called after the VDC registers are set up. user_num is used as the argument when the function is called. Specify '0' when this function is not required. • uint32_t user_num: User defined number Specify the argument to be passed to the user-defined function quit_func. This parameter is ignored if '0' is specified as the user-defined function.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error

Details

(1) Function

This function terminates the VDC driver and performs the following associated processing:

- Disables all the VDC interrupts.
- Disables the VDC panel clock.
- Calls the user-defined function specified in quit_func.

(2) Use conditions

There are no particular conditions with respect to the call of this function.

6.3 R_VDC_VideoInput

Synopsis	Video input setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_VideoInput(const vdc_channel_t ch, const vdc_input_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_input_t * param: Video input setup parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_PARAM_CONDITION: Unauthorized condition error

Details

(1) Function

This function performs the following processing on the video input:

- Sets up the phase timing of the input signals.
- Performs delay control on the sync signal for the video inputs.
- Sets up the parameters for the external input video signals.

(2) Use conditions

There are no particular conditions with respect to the call of this function.

(3) Parameter details

The members of the vdc_input_t structure is described below.

```
typedef struct
{
    vdc_input_sel_t            inp_sel;
    uint16_t                    inp_fh50;
    uint16_t                    inp_fh25;
    const vdc_sync_delay_t   * dly;
    const vdc_ext_in_sig_t   * ext_sig;
} vdc_input_t;
```

Type Member Name	Description
vdc_input_sel_t inp_sel	Input select • VDC_INPUT_SEL_EXT (1): Signals supplied via the external input pins Specify VDC_INPUT_SEL_EXT.
uint16_t inp_fh50	Vsync signal 1/2fH phase timing 0x0000 to 0x03FF 1/2 clock cycle of the horizontal cycle should be set.
uint16_t inp_fh25	Vsync signal 1/4fH phase timing 0x0000 to 0x03FF 1/4 clock cycle of the horizontal cycle should be set.
const vdc_sync_delay_t * dly	Sync signal delay adjustment parameter The setting is not changed if NULL is specified. If this parameter has never been set up after a hardware reset, the initial value that is defined in the hardware manual remains valid. See the description about the structure for the initial value.
const vdc_ext_in_sig_t * ext_sig	External input signal parameter Do not specify NULL.

The members of the vdc_sync_delay_t structure is described below.

```
typedef struct
{
    uint16_t    inp_vs_dly_l;
    uint16_t    inp_fld_dly;
    uint16_t    inp_vs_dly;
    uint16_t    inp_hs_dly;
} vdc_sync_delay_t;
```

Type Member Name	Initial Value	Description
uint16_t inp_vs_dly_l	0	Number of lines for delaying vsync signal and field differentiation signal 0 to 7 [lines]
uint16_t inp_fld_dly	0	Field differentiation signal delay amount 0 to 254 [clock cycles]
uint16_t inp_vs_dly	0	Vsync signal delay amount 0 to 254 [clock cycles]
uint16_t inp_hs_dly	0	Hsync signal delay amount 0 to 254 [clock cycles]

The members of the vdc_ext_in_sig_t structure is described below.

```
typedef struct
{
    vdc_extin_format_t     inp_format;
    vdc_edge_t             inp_pxd_edge;
    vdc_edge_t             inp_vs_edge;
    vdc_edge_t             inp_hs_edge;
    vdc_onoff_t            inp_endian_on;
    vdc_onoff_t            inp_swap_on;
    vdc_sig_pol_t          inp_vs_inv;
    vdc_sig_pol_t          inp_hs_inv;
    vdc_extin_ref_hsync_t  inp_h_edge_sel;
    vdc_extin_input_line_t inp_f525_625;
    vdc_extin_h_pos_t      inp_h_pos;
} vdc_ext_in_sig_t;
```

Type **Member Name**	**Description**
vdc_extin_format_t inp_format	External input format select • VDC_EXTIN_FORMAT_RGB888 (0): RGB888 • VDC_EXTIN_FORMAT_RGB666 (1): RGB666 • VDC_EXTIN_FORMAT_RGB565 (2): RGB565 • VDC_EXTIN_FORMAT_BT656 (3): BT6556 • VDC_EXTIN_FORMAT_BT601 (4): BT6501 • VDC_EXTIN_FORMAT_YCBCR422 (5): YCbCr422 • VDC_EXTIN_FORMAT_YCBCR444 (6): YCbCr444
vdc_edge_t inp_pxd_edge	Clock edge select for capturing external input video image signals DV_DATA23 to DV_DATA0 • VDC_EDGE_RISING: Rising edge • VDC_EDGE_FALLING: Falling edge
vdc_edge_t inp_vs_edge	Clock edge select for capturing external input Vsync signals DV_VSYNC • VDC_EDGE_RISING: Rising edge • VDC_EDGE_FALLING: Falling edge
vdc_edge_t inp_hs_edge	Clock edge select for capturing external input Hsync signals DV_HSYNC • VDC_EDGE_RISING: Rising edge • VDC_EDGE_FALLING: Falling edge
vdc_onoff_t inp_endian_on	External input bit endian change on/off control • VDC_OFF • VDC_ON
vdc_onoff_t inp_swap_on	External input B/R signal swap on/off control • VDC_OFF • VDC_ON
vdc_sig_pol_t inp_vs_inv	External input Vsync signal DV_VSYNC inversion control • VDC_SIG_POL_NOT_INVERTED: Not inverted (positive polarity) • VDC_SIG_POL_INVERTED: Inverted (negative polarity)
vdc_sig_pol_t inp_hs_inv	External input Hsync signal DV_HSYNC inversion control • VDC_SIG_POL_NOT_INVERTED: Not inverted (positive polarity) • VDC_SIG_POL_INVERTED: Inverted (negative polarity)

vdc_extin_ref_hsync_t inp_h_edge_sel	Reference select for external input BT656 Hsync signal • VDC_EXTIN_REF_H_EAV (0): EAV • VDC_EXTIN_REF_H_SAV (1): SAV
vdc_extin_input_line_t inp_f525_625	Number of lines for BT656 external input • VDC_EXTIN_LINE_525 (0): 525 lines • VDC_EXTIN_LINE_625 (1): 625 lines
vdc_extin_h_pos_t inp_h_pos	Y/Cb/Y/Cr data string start timing to Hsync reference • VDC_EXTIN_H_POS_CBYCRY (0): Cb/Y/Cr/Y (BT656/601), Cb/Cr (YCbCr422) • VDC_EXTIN_H_POS_YCRYCB (1): Y/Cr/Y/Cb (BT656/601), inhibited (YCbCr422) • VDC_EXTIN_H_POS_CRYCBY (2): Cr/Y/Cb/Y (BT656/601), inhibited (YCbCr422) • VDC_EXTIN_H_POS_YCBYCR (3): Y/Cb/Y/Cr (BT656/601), Cr/Cb (YCbCr422)

The function returns an unauthorized condition error (VDC_ERR_PARAM_CONDITION) if the data string start timing to Hsync reference (inp_h_pos) is set to VDC_EXTIN_H_POS_YCRYCB or VDC_EXTIN_H_POS_CRYCBY when YCbCr422 is selected as the external input format (inp_format).

6.4 R_VDC_SyncControl

Synopsis	Synchronization control setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_SyncControl(const vdc_channel_t ch, const vdc_sync_ctrl_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_sync_ctrl_t * param: Synchronization control parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_RESOURCE_CLK: Clock resource error — VDC_ERR_RESOURCE_INPUT: Input signal resource error

Details

(1) Function

This function performs the following synchronization control processing:

- Selects the vertical sync signal.
- Sets up the period of the sync signal.
- Sets up the delay of the vertical sync signal.
- Sets up the full-screen enable signal.
- Sets up the compensation for the vertical sync signal.

The settings established by this function remain valid until a hardware reset occurs or they are overwritten by this function with other settings.

(2) Use conditions

Before this function is used, the panel clock needs to have been set up. The function returns a clock resource error (VDC_ERR_RESOURCE_CLK) if the panel clock is not set up.

When selecting the external input Vsync signal as the Vsync signal output select to be specified in this function, it is necessary to enable the video input by calling the function R_VDC_VideoInput before using this function. The function returns an input signal resource error (VDC_ERR_RESOURCE_INPUT) if the video input is disabled.

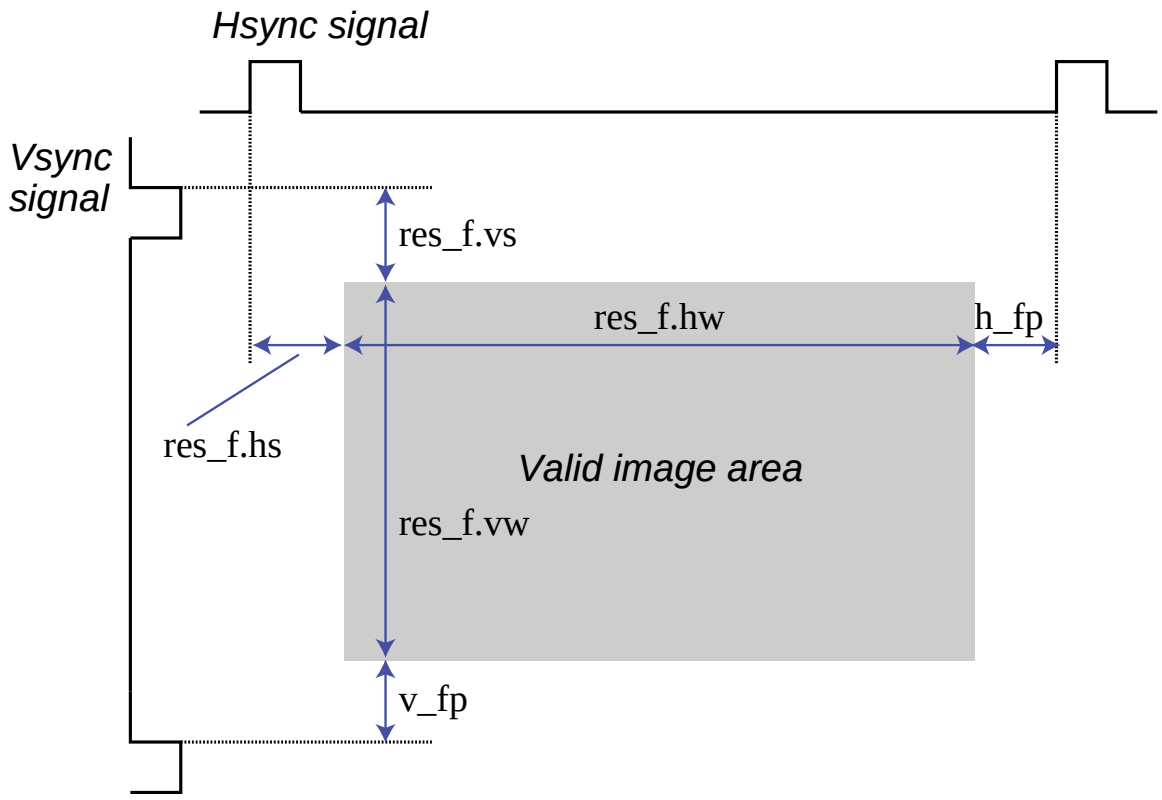
(3) Parameter details

The members of the vdc_sync_ctrl_t structure is described below.

```
typedef struct
{
    vdc_onoff_t                res_vs_sel;
    vdc_res_vs_in_sel_t        res_vs_in_sel;
    uint16_t                    res_fv;
    uint16_t                    res_fh;
    uint16_t                    res_vsdly;
    vdc_period_rect_t          res_f;
    const vdc_vsync_cpmpe_t  * vsync_cpmpe;
} vdc_sync_ctrl_t;
```

Type Member Name	Description
vdc_onoff_t res_vs_sel	Vsync signal output select (free-running Vsync signal) • VDC_OFF: External input Vsync signal • VDC_ON: Internally generated free-running Vsync signal
vdc_res_vs_in_sel_t res_vs_in_sel	Horizontal/vertical sync signal output and full-screen enable signal select • VDC_RES_VS_IN_SEL_SC0 (0): Scaler 0 outputs Specify VDC_RES_VS_IN_SEL_SC0.
uint16_t res_fv	Free-running Vsync period setting Free-running Vsync period = (res_fv + 1) x horizontal period [usec] 0x0000 to 0x07FF
uint16_t res_fh	Hsync period setting Hsync period [usec] = (res_fh + 1) / pixel clock frequency [MHz] 0x0000 to 0x07FF
uint16_t res_vsdly	Vsync signal delay control Adjusts the Vsync signal delay in the output Hsync period units. 0 to 255
vdc_period_rect_t res_f	Full-screen enable signal See 5.3(1) for the structure. res_f.vs should be 4 lines or more and res_f.vs + res_f.vw shold be equal to or less than 2039 lines. res_f.hs should be 16 clock cycles or more and res_f.hs + res_f.hw should be equal to or less than 2015 clock cycles. See also Figure 6-1 and its explanation for these settings.
const vdc_vsync_cpmpe_t * vsync_cpmpe	Vsync signal compensation parameter Specifying NULL turns off the repeated Vsync signal masking control and the compensation of missing Vsync signals. If the compensation of missing Vsync signals is set to OFF, the missing-sync compensating pulse output wait time is set to its maximum value (0xFFFF) by the driver.

Figure 6-1 shows the valid period of the full-screen enable and the output image. The 16 clock cycles before and after the Hsync signal and the 4 lines before and after the Vsync signal are not included in the valid period of the image.



res_f.vs: Vertical enable signal start position for full screen (4 lines or more)
res_f.vw: Vertical enable signal width for full screen
v_fp: From the end of the vertical enable signal to the Vsync signal (4 lines or more)
res_f.hs: Horizontal enable signal start position for full screen (16 clock cycles or more)
res_f.hw: Horizontal enable signal width for full screen
h_fp: From the end of the horizontal enable signal to the Hsync signal (16 clock cycles or more)

Figure 6-1 Full Screen Enable Settings

vdc_vsync_cpmpe_t structure is described below.

```
typedef struct
{
    uint16_t        res_vmask;
    uint16_t        res_vlack;
} vdc_vsync_cpmpe_t;
```

Type Member Name	Description
uint16_t res_vmask	Repeated Vsync signal masking period Masking period [usec] = res_vmask x 128 / pixel clock frequency [MHz]
uint16_t res_vlack	Missing-sync compensating pulse output wait time Wait time [usec] = res_vlack x 128 / pixel clock frequency [MHz]

6.5 R_VDC_DisplayOutput

Synopsis	Display output setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_DisplayOutput(const vdc_channel_t ch, const vdc_output_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_output_t * param: Display output configuration parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_CONDITION: Unauthorized condition error — VDC_ERR_RESOURCE_CLK: Clock resource error — VDC_ERR_RESOURCE_VSYNC: Vertical sync signal resource error

Details

(1) Function

This function performs the following processing on the display output:

- Sets up the timing signals for driving the LCD panel.
- Sets up the phase, data sequence, and format of the LCD panel output data.
- Sets up the background color.

The settings established by this function remain valid until a hardware reset occurs or they are overwritten by this function with other settings.

(2) Use conditions

Before this function is used, the panel clock and sync signals need to have been set up. The function returns a clock resource error (VDC_ERR_RESOURCE_CLK) if the panel clock is not set up and a vertical sync signal resource error (VDC_ERR_RESOURCE_VSYNC) if no sync signal is set up.

(3) Parameter details

The members of the vdc_output_t structure is described below.

```
typedef struct
{
    uint16_t                        tcon_half;
    uint16_t                        tcon_offset;
    const vdc_lcd_tcon_timing_t   * outctrl[VDC_LCD_TCONSIG_NUM];
    vdc_edge_t                      outcnt_lcd_edge;
    vdc_onoff_t                     out_endian_on;
    vdc_onoff_t                     out_swap_on;
    vdc_lcd_outformat_t             out_format;
    vdc_lcd_clkfreqsel_t            out_frq_sel;
    vdc_lcd_scan_t                  out_dir_sel;
    vdc_lcd_clkphase_t              out_phase;
    uint32_t                        bg_color;
} vdc_output_t;
```

Type Member Name	Description
uint16_t tcon_half	1/2fH timing Specifies the clock count from the rising edge of the Hsync signal as the counting timing of horizontal counter. 0x0000 to 0x07FF
uint16_t tcon_offset	Offset Hsync signal timing Sets the clock cycle count from the rising edge of the Hsync signal. 0x0000 to 0x07FF
const vdc_lcd_tcon_timing_t * outctrl[VDC_LCD_TCONSIG_NUM]	LCD TCON timing signal parameter Specify NULL for any signals that are not to be used.
vdc_edge_t outcnt_lcd_edge	Output phase control of LCD_DATA23 to LCD_DATA0 pin • VDC_EDGE_RISING: Output at the rising edge of LCD_CLK pin. • VDC_EDGE_FALLING: Output at the falling edge of LCD_CLK pin.
vdc_onoff_t out_endian_on	Bit endian change on/off control • VDC_OFF • VDC_ON
vdc_onoff_t out_swap_on	B/R signal swap on/off control • VDC_OFF • VDC_ON
vdc_lcd_outformat_t out_format	Output format select • VDC_LCD_OUTFORMAT_RGB888 (0): RGB888 • VDC_LCD_OUTFORMAT_RGB666 (1): RGB666 • VDC_LCD_OUTFORMAT_RGB565 (2): RGB565 • VDC_LCD_OUTFORMAT_SERIAL_RGB (3): Serial RGB
vdc_lcd_clkfreqsel_t out_frq_sel	Clock frequency control • VDC_LCD_PARALLEL_CLKFRQ_1 (0): 100% speed (parallel RGB) • VDC_LCD_SERIAL_CLKFRQ_3 (1): Triple speed (serial RGB) • VDC_LCD_SERIAL_CLKFRQ_4 (2): Quadruple speed (serial RGB) This parameter is referenced only when out_format is set to VDC_LCD_OUTFORMAT_SERIAL_RGB. In this case, setting this parameter to 100% speed is inhibited.

vdc_lcd_scan_t out_dir_sel	Scan direction select • VDC_LCD_SERIAL_SCAN_FORWARD (0): Forward scan • VDC_LCD_SERIAL_SCAN_REVERSE (1): Reverse scan This parameter is referenced only when out_format is set to VDC_LCD_OUTFORMAT_SERIAL_RGB.
vdc_lcd_clkphase_t out_phase	Clock phase adjustment during serial RGB output • VDC_LCD_SERIAL_CLKPHASE_0 (0): 0 [clocks] • VDC_LCD_SERIAL_CLKPHASE_1 (1): 1 [clocks] • VDC_LCD_SERIAL_CLKPHASE_2 (2): 2 [clocks] • VDC_LCD_SERIAL_CLKPHASE_3 (3): 3 [clocks] This parameter is referenced only when out_format is set to VDC_LCD_OUTFORMAT_SERIAL_RGB. It is inhibited to set this parameter to VDC_LCD_SERIAL_CLKPHASE_3 when out_frq_sel is set to VDC_LCD_SERIAL_CLKFRQ_3.
uint32_t bg_color	Background Color Specify in the RGB888 format (LSB justified).

vdc_lcd_tcon_sigsel_t is an enumeration type for representing the timing signals (LCD TCON) for driving the LCD panel.

```
typedef enum
{
    VDC_LCD_TCONSIG_STVA_VS = 0,
    VDC_LCD_TCONSIG_STVB_VE,
    VDC_LCD_TCONSIG_STH_SP_HS,
    VDC_LCD_TCONSIG_STB_LP_HE,
    VDC_LCD_TCONSIG_CPV_GCK,
    VDC_LCD_TCONSIG_POLA,
    VDC_LCD_TCONSIG_POLB,
    VDC_LCD_TCONSIG_DE,
    VDC_LCD_TCONSIG_NUM
} vdc_lcd_tcon_sigsel_t;
```

Enumeration constant	Value	Description
VDC_LCD_TCONSIG_STVA_VS	0	Gate start signal, Vsync signal (STVA/VS)
VDC_LCD_TCONSIG_STVB_VE	1	Gate start signal, vertical enable signal (STVB/VE)
VDC_LCD_TCONSIG_STH_SP_HS	2	Source start signal, Hsync signal (STH/SP/HS)
VDC_LCD_TCONSIG_STB_LP_HE	3	Source strobe signal, horizontal enable signal (STB/LP/HE)
VDC_LCD_TCONSIG_CPV_GCK	4	Gate clock signal (CPV/GCK)
VDC_LCD_TCONSIG_POLA	5	VCOM voltage polarity control signal (POLA)
VDC_LCD_TCONSIG_POLB	6	VCOM voltage polarity control signal (POLB)
VDC_LCD_TCONSIG_DE	7	Data enable signal (DE)
VDC_LCD_TCONSIG_NUM	8	Number of LCD panel drive signal types

The members of the vdc_lcd_tcon_timing_t structure is described below.

```
typedef struct
{
    uint16_t                tcon_hsvs;
    uint16_t                tcon_hwvw;
    vdc_lcd_tcon_polmode_t tcon_md;
    vdc_lcd_tcon_refsel_t  tcon_hs_sel;
    vdc_sig_pol_t          tcon_inv;
    vdc_lcd_tcon_pin_t     tcon_pin;
    vdc_edge_t             outcnt_edge;
} vdc_lcd_tcon_timing_t;
```

Type Member Name	Description
uint16_t tcon_hsvs	Signal pulse start position (first changing timing) Starts pulse output after the time specified by the value of tcon_hsvs from the rising edge of the reference signal. 0x0000 to 0x07FF [clock cycles, 1/2fH cycles] Set a value of 1 or greater if tcon_md is set to a value other than VDC_LCD_TCON_POLMD_NORMAL when using the POLA/POLB signal.
uint16_t tcon_hwvw	Pulse width (second changing timing) Outputs a pulse of the duration of the value of tcon_hwvw. 0x0000 to 0x07FF [clock cycles, 1/2fH cycles]
vdc_lcd_tcon_polmode_t tcon_md	POLA/POLB signal generation mode select • VDC_LCD_TCON_POLMD_NORMAL (0): Normal mode Generates the signal whose polarity is inverted every horizontal period. • VDC_LCD_TCON_POLMD_1X1REV (1): 1x1 reverse mode Generates the signal whose polarity is inverted every horizontal period. • VDC_LCD_TCON_POLMD_1X2REV (2): 1x2 reverse mode Generates the signal whose polarity is inverted in the first horizontal period and is subsequently inverted every two horizontal periods. • VDC_LCD_TCON_POLMD_2X2REV (3): 2x2 reverse mode Generates the signal whose polarity is inverted every two horizontal periods.
vdc_lcd_tcon_refsel_t tcon_hs_sel	Operating reference select • VDC_LCD_TCON_REFSEL_HSYNC (0): Hsync signal reference • VDC_LCD_TCON_REFSEL_OFFSET_H (1): Offset Hsync signal reference
vdc_sig_pol_t tcon_inv	Polarity inversion control of signal • VDC_SIG_POL_NOT_INVERTED Not inverted (positive polarity) • VDC_SIG_POL_INVERTED: Inverted (negative polarity)
vdc_lcd_tcon_pin_t tcon_pin	LCD TCON output pin select • VDC_LCD_TCON_PIN_NON (-1): Nothing output.

	• VDC_LCD_TCON_PIN_0 (0): LCD_TCON0 output. • VDC_LCD_TCON_PIN_1 (1): LCD_TCON1 output. • VDC_LCD_TCON_PIN_2 (2): LCD_TCON2 output. • VDC_LCD_TCON_PIN_3 (3): LCD_TCON3 output. • VDC_LCD_TCON_PIN_4 (4): LCD_TCON4 output. • VDC_LCD_TCON_PIN_5 (5): LCD_TCON5 output. • VDC_LCD_TCON_PIN_6 (6): LCD_TCON6 output.
vdc_edge_t outcnt_edge	Output phase control of the signal • VDC_EDGE_RISING: Output at the rising edge of LCD_CLK pin. • VDC_EDGE_FALLING: Output at the falling edge of LCD_CLK pin.

Different members of the vdc_lcd_tcon_timing_t structure are referenced depending on the type of the LCD panel drive signals that are to be set up. The table below summarizes the members that are valid or invalid when the respective signals are set up. Invalid members are not referenced.

Table 6-3 Valid Parameters for the LCD Panel Drive Signals

LCD Panel Drive Signal	**vdc_lcd_tcon_timing_t Structure Member**						
	tcon_hsvs	**tcon_hwvw**	**tcon_md**	**tcon_hs_sel**	**tcon_inv**	**tcon_pin**	**outcnt_edge**
STVA/VS	valid	valid	-	-	valid	valid	valid
STVB/VE	valid	valid	-	-	valid	valid	valid
STH/SP/HS	valid	valid	-	valid	valid	valid	valid
STB/LP/HE	valid	valid	-	valid	valid	valid	valid
CPV/GCK	valid	valid	-	valid	valid	valid	valid
POLA	valid	valid	valid	valid	valid	valid	valid
POLB	valid	valid	valid	valid	valid	valid	valid
DE	-	-	-	-	valid	valid	valid

Note: 'valid' denotes a valid member whose value is referenced.
'-' denotes an invalid member whose value is not referenced.

6.6 R_VDC_CallbackISR

Synopsis	Interrupt callback setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_CallbackISR(const vdc_channel_t ch, const vdc_int_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_int_t * param: Interrupt callback setup parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_RESOURCE_CLK: Clock resource error — VDC_ERR_RESOURCE_VSYNC: Vertical sync signal resource error

Details

(1) Function

This function performs the following VDC interrupt processing:

- Enables the interrupt when the pointer to the corresponding interrupt callback function is specified.
- Registers the specified interrupt callback function.
- Disables the interrupt when the pointer to the corresponding interrupt callback function is not specified.

With this driver, all of the VDC interrupts are disabled and the registered entries of the callback functions are removed during R_VDC_Initialize and R_VDC_Terminate. The setting of an existing interrupt can also be overwritten by applying this function R_VDC_CallbackISR to that interrupt.

(2) Use conditions

Before this function is used, the panel clock and sync signals need to have been set up. The function returns a clock resource error (VDC_ERR_RESOURCE_CLK) if the panel clock is not set up and a vertical sync signal resource error (VDC_ERR_RESOURCE_VSYNC) if no sync signal is set up.

(3) Parameter details

The members of the vdc_int_t structure is described below.

```
typedef struct
{
    vdc_int_type_t     type;
    void             (* callback)(vdc_int_type_t);
    uint16_t            line_num;
} vdc_int_t;
```

Type Member Name	Description
vdc_int_type_t type	VDC interrupt type See 5.2(8) for details.
void (* callback)(vdc_int_type_t)	Interrupt callback function pointer Specify the pointer to the interrupt callback function associated with the interrupt that is specified in type. The callback function needs to be implemented by the user. The interrupt processing of the interrupt for which a callback function is specified becomes enabled. If a '0' is specified in callback, the interrupt specified in type becomes disabled.
uint16_t line_num	Line interrupt set An interrupt signal is output when the line position of the image matches the value of line_num. This parameter is valid only when one of the following line interrupts is specified in type: — VDC_INT_TYPE_VLINE — VDC_INT_TYPE_S0_WLINE

6.7 R_VDC_WriteDataControl

Synopsis	Data write control processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_WriteDataControl(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_write_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_0_WR: Layer 0 write processing Specify layer 0 write processing in this driver. • vdc_write_t * param: Data write control parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_RESOURCE_INPUT: Input signal resource error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following data write control processing:

- Sets up the input image area to be captured.
- Makes input image scaling-down and rotation control settings.
- Makes frame buffer write control settings.

(2) Use conditions

Before using this function, it is necessary to enable a video input by calling the function R_VDC_VideoInput. If no video input is enabled, the function returns an input signal resource error (VDC_ERR_RESOURCE_INPUT).

This function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the layer that is specified in layer_id is found enabled already when this function is used. The enabled layer can be disabled by calling the function R_VDC_ReleaseDataControl.

Control of the vertical scale-down processing is mutually exclusive with the control of the vertical scale-up processing. Normal operation of the driver operation is not guaranteed if vertical scale-down and vertical

scale-up are specified at the same time. The setup of vertical scale-up processing is accomplished by the functions R_VDC_ReadDataControl and R_VDC_ChangeReadProcess.

(3) Parameter details

The members of the vdc_write_t structure is described below

```
typedef struct
{
    vdc_scalingdown_rot_t  scalingdown_rot;
    vdc_wr_rd_swa_t        res_wrswa;
    vdc_res_md_t           res_md;
    vdc_bst_md_t           res_bst_md;
    vdc_res_inter_t        res_inter;
    vdc_res_fs_rate_t      res_fs_rate;
    vdc_res_fld_sel_t      res_fld_sel;
    vdc_onoff_t            res_dth_on;
    void                  * base;
    uint32_t                ln_off;
    uint32_t                flm_num;
    uint32_t                flm_off;
    void                  * btm_base;
} vdc_write_t;
```

Type Member Name	Description
vdc_scalingdown_rot_t scalingdown_rot	Scaling-down and rotation parameter
vdc_wr_rd_swa_t res_wrswa	8-bit, 16-bit, or 32-bit swap setting • VDC_WR_RD_WRSWA_NON (0): 1-2-3-4-5-6-7-8 Not swapped • VDC_WR_RD_WRSWA_8BIT (1): 2-1-4-3-6-5-8-7 Swapped in 8-bit units • VDC_WR_RD_WRSWA_16BIT (2): 3-4-1-2-7-8-5-6 Swapped in 16-bit units • VDC_WR_RD_WRSWA_16_8BIT (3): 4-3-2-1-8-7-6-5 Swapped in 16-bit units + 8-bit units • VDC_WR_RD_WRSWA_32BIT (4): 5-6-7-8-1-2-3-4 Swapped in 32-bit units • VDC_WR_RD_WRSWA_32_8BIT (5): 6-5-8-7-2-1-4-3 Swapped in 32-bit units + 8-bit units • VDC_WR_RD_WRSWA_32_16BIT (6): 7-8-5-6-3-4-1-2 Swapped in 32-bit units + 16-bit units • VDC_WR_RD_WRSWA_32_16_8BIT (7): 8-7-6-5-4-3-2-1 Swapped in 32-bit units + 16-bit units + 8-bit units
vdc_res_md_t res_md	Frame buffer video-signal writing format • VDC_RES_MD_YCBCR422 (0): YCbCr422 • VDC_RES_MD_RGB565 (1): RGB565 • VDC_RES_MD_RGB888 (2): RGB888 • VDC_RES_MD_YCBCR444 (3): YCbCr444
vdc_bst_md_t res_bst_md	Transfer burst length for frame buffer writing • VDC_BST_MD_32BYTE (0): 32-byte transfer (4 bursts) • VDC_BST_MD_128BYTE (1): 128-byte transfer (16 bursts)

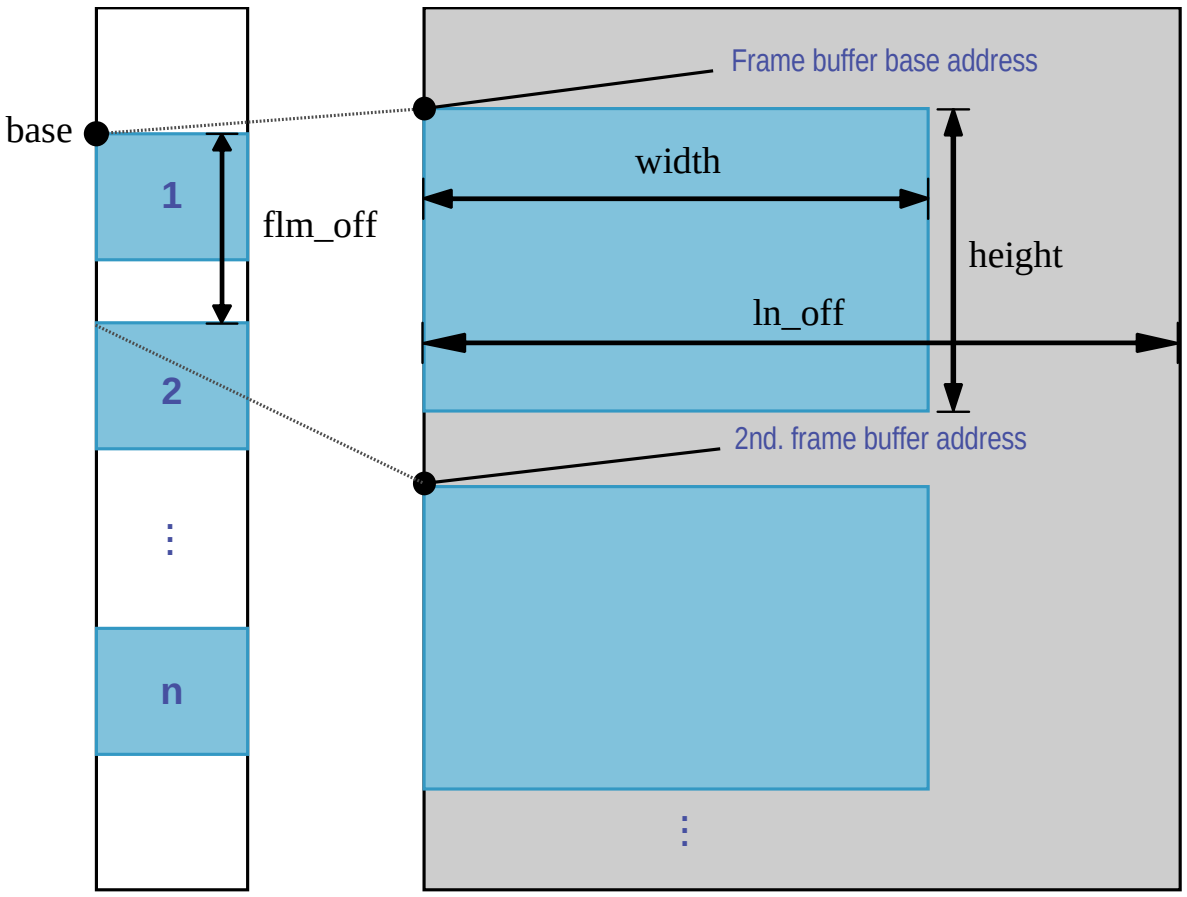
vdc_res_inter_t res_inter	Field operating mode select • VDC_RES_INTER_PROGRESSIVE (0): Progressive • VDC_RES_INTER_INTERLACE (1): Interlace
vdc_res_fs_rate_t res_fs_rate	Writing rate • VDC_RES_FS_RATE_PER1 (0): 1/1 an input signal • VDC_RES_FS_RATE_PER2 (1): 1/2 an input signal • VDC_RES_FS_RATE_PER4 (2): 1/4 an input signal • VDC_RES_FS_RATE_PER8 (3): 1/8 an input signal
vdc_res_fld_sel_t res_fld_sel	Write field select This parameter is valid only when res_fs_rate is set to a value other than VDC_RES_FS_RATE_PER1. • VDC_RES_FLD_SEL_TOP (0): Top field • VDC_RES_FLD_SEL_BOTTOM (1): Bottom field
vdc_onoff_t res_dth_on	Dither correction on/off • VDC_OFF: Rounded off • VDC_ON: 2x2 dither pattern
void * base	Frame buffer base address Do not specify NULL. When the value specified in res_bst_md is: • VDC_BST_MD_32BYTE Specify an address that is aligned to 32 bytes. • VDC_BST_MD_128BYTE Specify an address that is aligned to 128 byte.
uint32_t ln_off	Frame buffer line offset address 0x0000 to 0x7FFF When the value specified in res_bst_md is: • VDC_BST_MD_32BYTE Specify a multiple of 32. • VDC_BST_MD_128BYTE Specify a multiple of 128.
uint32_t flm_num	Number of frames of buffer to be written to 0x0000 to 0x03FF Number of frames defined by flm_num + 1 is used. Specify 2 frames ('1') or more for rotation processing.
uint32_t flm_off	Frame buffer frame offset address 0x00000000 to 0x007FFFFF This parameter is invalid when the number of frames is 1 (flm_num is set to '0'). When the value specified in res_bst_md is: • VDC_BST_MD_32BYTE Specify a multiple of 32. • VDC_BST_MD_128BYTE Specify a multiple of 128.
void * btm_base	Frame buffer base address for bottom Specify NULL if not required. When the value specified in res_bst_md is: • VDC_BST_MD_32BYTE Specify an address that is aligned to 32 bytes. • VDC_BST_MD_128BYTE Specify an address that is aligned to 128 bytes.

The members of the vdc_scalingdown_rot_t structure is described below.

```
typedef struct
{
    vdc_period_rect_t  res;
    vdc_onoff_t        res_pfil_sel;
    uint16_t            res_out_vw;
    uint16_t            res_out_hw;
    vdc_onoff_t        adj_sel;
    vdc_wr_md_t        res_ds_wr_md;
} vdc_scalingdown_rot_t;
```

Type Member Name	Description
vdc_period_rect_t res	Image area to be captured See 5.3(1) for the vdc_period_rect_t structure. res.vs should be 4 lines or more and res.vs + res.vw shold be equal to or less than 2039 lines. res.hs should be 16 clock cycles or more and res.hs + res.hw should be equal to or less than 2015 clock cycles. The actual vertical position setting for video signal capturing is res.vs + 1.
vdc_onoff_t res_pfil_sel	Prefilter mode select for brightness signals • VDC_OFF • VDC_ON
uint16_t res_out_vw	Number of valid lines in vertical direction output by scaling-down control block (lines) 0x0000 to 0x07FF Specify a value that is aligned in 4-line units and equal to or smaller than the res.vw value.
uint16_t res_out_hw	Number of valid horizontal pixels output by scaling-down control block (video-image clock cycles) 0x0000 to 0x07FF Specify a value that is aligned in 4-pixel units and equal to or smaller than the res.hw value.
vdc_onoff_t adj_sel	Handling for lack of last-input line Specifies whether to take countermeasures for decreasing the influence by the lack of last-input line in scale-down processing. • VDC_OFF • VDC_ON
vdc_wr_md_t res_ds_wr_md	Frame buffer writing mode for image processing • VDC_WR_MD_NORMAL (0): Normal • VDC_WR_MD_MIRROR (1): Horizontal mirroring • VDC_WR_MD_ROT_90DEG (2): 90-degree rotation • VDC_WR_MD_ROT_180DEG (3): 180-degree rotation • VDC_WR_MD_ROT_270DEG (4): 270-degree rotation Setting this parameter to 90-degree, 180-degree, or 270-degree rotation is valid only when frame buffer video-signal writing format (res_md) is set to YCbCr422 or RGB565.

Figure 6-2 shows the relationship between the settings of the frame buffer related parameters and the memory allocation.



base: Frame buffer base address
flm_off: Frame buffer frame offset address [bytes]
n: Number of frame buffers (= flm_num + 1)
height: Number of valid lines in vertical direction output by scaling-down control block [lines]
width: Number of valid horizontal pixels output by scaling-down control block [video-image clock cycles]
ln_off: Frame buffer line offset address [bytes]

Figure 6-2 Frame Buffer Related Parameter Settings and Memory Allocation

The width and height of the image in the frame buffer (width and height in Figure 6-2) are the number of valid horizontal pixels (res_out_hw) and the number of valid lines in vertical direction (res_out_vw) output by scaling-down control block, respectively. But when the frame buffer writing mode for image processing (res_ds_wr_md) is set to 90-degree rotation or 270-degree rotation, the width and height of the image are res_out_vw and res_out_hw, respectively.

6.8 R_VDC_ChangeWriteProcess

Synopsis	Data write change processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_ChangeWriteProcess(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_write_chg_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_0_WR: Layer 0 write processing Specify layer 0 write processing in this driver. • vdc_write_chg_t * param: Data write change parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function takes the following data write control actions during data write processing:

- Changes the input image area to be captured.
- Makes changes with respect to scaling-down and rotation control of the input image.

(2) Use conditions

This function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) unless the layer specified in layer_id of this function meets the following conditions:

- The specified layer is enabled.
- The specified layer is running.

A layer is enabled by calling the function R_VDC_WriteDataControl. A layer that is in the stopped state can be started by calling the function R_VDC_StartProcess.

Control of the vertical scale-down processing is mutually exclusive with the control of the vertical scale-up processing. Normal operation of the driver operation is not guaranteed if vertical scale-down and vertical scale-up are specified at the same time. The setup of vertical scale-up processing is accomplished by the functions R_VDC_ReadDataControl and R_VDC_ChangeReadProcess.

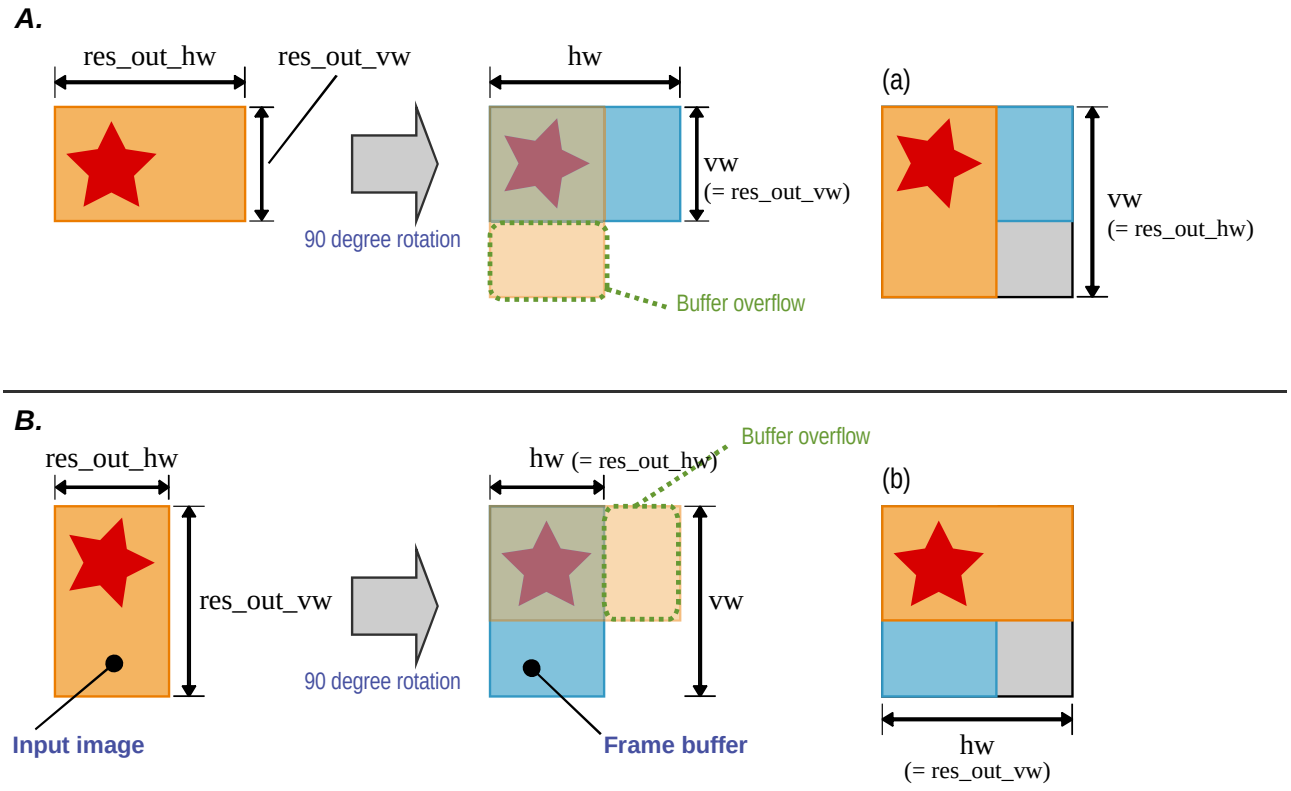
(3) Parameter details

The members of the vdc_write_chg_t structure is described below.

```
typedef struct
{
    vdc_scalingdown_rot_t  scalingdown_rot;
} vdc_write_chg_t;
```

Type Member Name	Description
vdc_scalingdown_rot_t scalingdown_rot	Scaling-down and rotation parameter See 6.7(3) for details.

This function can be used to change the rotation control mode during data write processing. The size of the frame buffer that is required for the data write may be altered of the rotation angle is changed by 90 or 270 degrees from the current angle.



res_out_vw: Vertical image width after scale-down
res_out_hw: Horizontal image width after scale-down
vw: Frame buffer vertical width hw: Frame buffer horizontal width

Figure 6-3 Rotation Processing and Frame Buffer Size

Figure 6-3 shows two cases of processing in which the frame buffer size is altered as the result of rotating the input image by 90 degrees. Before the rotation, the width (hw) and height (vw) of the frame buffer that is necessary for data write processing are set to (res_out_hw) and (res_out_vw), respectively. Since these size values change as the result of the 90-degree rotation, data is likely to be written into an unexpected area. To avoid this situation, it is necessary to reserve a larger frame buffer in advance ((a) and (b) in the figure).

6.9 R_VDC_ReadDataControl

Synopsis	Data read control processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_ReadDataControl(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_read_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_0_RD: Layer 0 read processing — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing • vdc_read_t * param: Data read control parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_PARAM_CONDITION: Unauthorized condition error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following data read control processing:

- Sets up the display area for graphics images.
- Makes image scale-up control settings (layer 0 only).
- Makes frame buffer read control settings.

(2) Use conditions

This function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the layer specified in layer_id is found enabled already when this function is used. The enabled layer can be disabled by calling the function R_VDC_ReleaseDataControl.

Control of the vertical scale-up processing is mutually exclusive with the control of the vertical scale-down processing. Normal operation of the driver operation is not guaranteed if vertical scale-down and vertical scale-up are specified at the same time. The setup of vertical scale-down processing is accomplished by the functions R_VDC_WriteDataControl and R_VDC_ChangeWriteProcess.

(3) Parameter details

vdc_read_t structure is described below.

```
typedef struct
{
    vdc_gr_ln_off_dir_t            gr_ln_off_dir;
    vdc_gr_flm_sel_t               gr_flm_sel;
    vdc_onoff_t                    gr_imr_flm_inv;
    vdc_bst_md_t                   gr_bst_md;
    void                          * gr_base;
    uint32_t                       gr_ln_off;
    const vdc_width_read_fb_t     * width_read_fb;
    vdc_onoff_t                    adj_sel;
    vdc_gr_format_t                gr_format;
    vdc_gr_ycc_swap_t              gr_ycc_swap;
    vdc_wr_rd_swa_t                gr_rdswa;
    vdc_period_rect_t              gr_grc;
} vdc_read_t;
```

Type Member Name	Description
vdc_gr_ln_off_dir_t gr_ln_off_dir	Line offset address direction of the frame buffer • VDC_GR_LN_OFF_DIR_INC (0): Increments the address by the line offset address. • VDC_GR_LN_OFF_DIR_DEC (1): Decrements the address by the line offset address.
vdc_gr_flm_sel_t gr_flm_sel	Frame buffer address setting signal • VDC_GR_FLM_SEL_SCALE_DOWN (0): Links to scaling-down process. • VDC_GR_FLM_SEL_FLM_NUM (1): Selects frame 0. • VDC_GR_FLM_SEL_POINTER_BUFF (3): Links to pointer buffer.
vdc_onoff_t gr_imr_flm_inv	Frame buffer number for distortion correction This parameter is not referred in this driver. Specify VDC_OFF.
vdc_bst_md_t gr_bst_md	Frame buffer burst transfer mode • VDC_BST_MD_32BYTE (0): 32-byte transfer • VDC_BST_MD_128BYTE (1): 128-byte transfer
void * gr_base	Frame buffer base address Do not specify NULL if gr_flm_sel is set to a value other than VDC_GR_FLM_SEL_POINTER_BUFF.
uint32_t gr_ln_off	Frame buffer line offset address 0x0000 to 0x7FFF When the value specified in gr_bst_md is: • VDC_BST_MD_32BYTE Specify a multiple of 32. • VDC_BST_MD_128BYTE Specify a multiple of 128.
const vdc_width_read_fb_t * width_read_fb	Size of the frame buffer to be read If NULL is specified, the size values of the frame buffer are assumed to be equal to the graphics display area width (gr_grc.hw) and height (gr_grc.vw).
vdc_onoff_t adj_sel	Folding handling

	Specifies whether to take countermeasures for decreasing the influence by folding pixels in scale-up processing. • VDC_OFF • VDC_ON
vdc_gr_format_t gr_format	Format of the frame buffer read signal • VDC_GR_FORMAT_RGB565 (0): RGB565 • VDC_GR_FORMAT_RGB888 (1): RGB888 • VDC_GR_FORMAT_ARGB1555 (2): ARGB1555 • VDC_GR_FORMAT_ARGB4444 (3): ARGB4444 • VDC_GR_FORMAT_ARGB8888 (4): ARGB8888 • VDC_GR_FORMAT_CLUT8 (5): CLUT8 • VDC_GR_FORMAT_CLUT4 (6): CLUT4 • VDC_GR_FORMAT_CLUT1 (7): CLUT1 • VDC_GR_FORMAT_YCBCR422 (8): YCbCr422 * • VDC_GR_FORMAT_YCBCR444 (9): YCbCr444 * • VDC_GR_FORMAT_RGBA5551 (10): RGBA5551 • VDC_GR_FORMAT_RGBA8888 (11): RGBA8888
vdc_gr_ycc_swap_t gr_ycc_swap	Swapping of data read from buffer in the YCbCr422 format This parameter is valid only when gr_format is set to VDC_GR_FORMAT_YCBCR422. • VDC_GR_YCCSWAP_CBY0CRY1 (0): CbY0/Cr/Y1 • VDC_GR_YCCSWAP_Y0CBY1CR (1): Y0/Cb/Y1/Cr • VDC_GR_YCCSWAP_CRY0CBY1 (2): Cr/Y0/Cb/Y1 • VDC_GR_YCCSWAP_Y0CRY1CB (3): Y0/Cr/Y1/Cb • VDC_GR_YCCSWAP_Y1CRY0CB (4): Y1/Cr/Y0/Cb • VDC_GR_YCCSWAP_CRY1CBY0 (5): Cr/Y1/Cb/Y0 • VDC_GR_YCCSWAP_Y1CBY0CR (6): Y1/Cb/Y0/Cr • VDC_GR_YCCSWAP_CBY1CRY0 (7): Cb/Y1/Cr/Y0
vdc_wr_rd_swa_t gr_rdswa	8-bit, 16-bit, or 32-bit swap setting • VDC_WR_RD_WRSWA_NON (0): 1-2-3-4-5-6-7-8 No swap • VDC_WR_RD_WRSWA_8BIT (1): 2-1-4-3-6-5-8-7 8-bit swap • VDC_WR_RD_WRSWA_16BIT (2): 3-4-1-2-7-8-5-6 16-bit swap • VDC_WR_RD_WRSWA_16_8BIT (3): 4-3-2-1-8-7-6-5 16-bit + 8-bit swap • VDC_WR_RD_WRSWA_32BIT (4): 5-6-7-8-1-2-3-4 32-bit swap • VDC_WR_RD_WRSWA_32_8BIT (5): 6-5-8-7-2-1-4-3 32-bit + 8-bit swap • VDC_WR_RD_WRSWA_32_16BIT (6): 7-8-5-6-3-4-1-2 32-bit + 16-bit swap • VDC_WR_RD_WRSWA_32_16_8BIT (7): 8-7-6-5-4-3-2-1 32-bit + 16-bit + 8-bit swap
vdc_period_rect_t gr_grc	Graphics display area See 5.3(1) for the vdc_period_rect_t structure. gr_grc.vs should be 4 lines or more and gr_grc.vs + gr_grc.vw shold be equal to or less than 2039 lines. gr_grc.hs should be 16 clock cycles or more and gr_grc.hs + gr_grc.hw should be equal to or less than 2015 clock cycles.

Note: YCbCr422 and YCbCr444 can be specified only for graphics 0.

The legitimate parameter values for the frame buffer address setting signal (gr_flm_sel) differ from layer to layer. See Table 6-4 for legitimate parameter values for the layers.

Table 6-4 Legitimate Frame Buffer Address Setting Signal Parameter Values

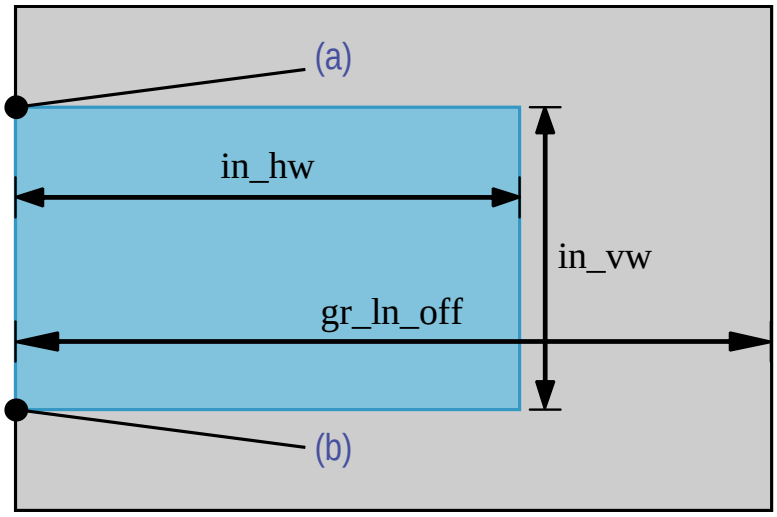
Layer ID	Legitimate Value
VDC_LAYER_ID_0_RD	VDC_GR_FLM_SEL_SCALE_DOWN VDC_GR_FLM_SEL_FLM_NUM VDC_GR_FLM_SEL_POINTER_BUFF
VDC_LAYER_ID_2_RD	VDC_GR_FLM_SEL_FLM_NUM
VDC_LAYER_ID_3_RD	VDC_GR_FLM_SEL_FLM_NUM

The members of the vdc_width_read_fb_t structure is described below.

```
typedef struct
{
    uint16_t                in_vw;
    uint16_t                in_hw;
} vdc_width_read_fb_t;
```

Type Member Name	Description
uint16_t in_vw	Number of lines in a frame (lines) 0x0000 to 0x07FF
uint16_t in_hw	Width of the horizontal valid period (pixels) 0x0000 to 0x07FF

Figure 6-4 shows the relationship between the parameter settings for the frame buffer to be used during data read processing and the memory allocation.



(a), (b): Frame buffer base address (gr_base)
in_vw: Number of lines in a frame [lines]
in_hw: Width of the horizontal valid period [pixels]
gr_ln_off: Frame buffer line offset address [bytes]

Figure 6-4 Read Processing Frame Buffer Related Parameter Settings and Memory Allocation

The value of the frame buffer base address (gr_base) need be changed according to the setting of the line offset address direction of the frame buffer (gr_ln_off_dir).

- When gr_ln_off_dir is set to VDC_GR_LN_OFF_DIR_INC
  Data is read sequentially starting at the beginning of the frame buffer. Specify in gr_base the start address of the frame buffer (Figure 6-4, (a)).
- When gr_ln_off_dir is set to VDC_GR_LN_OFF_DIR_DEC
  Data is read from the last line of the frame buffer. Specify in gr_base the address of the last line in the frame buffer (Figure 6-4, (b)).

6.10 R_VDC_ChangeReadProcess

Synopsis	Data read change processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_ChangeReadProcess(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_read_chg_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_0_RD: Layer 0 read processing — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing • vdc_read_chg_t * param: Data read change parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function takes the following data read control actions during data read processing:

- Changes the frame buffer base address.
- Changes the frame buffer read size (image scale-up control, layer 0 only).
- Changes the display area for graphics images.
- Changes the graphics display mode.

(2) Use conditions

This function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) unless the layer specified in layer_id of this function meets the following conditions:

- The specified layer is enabled.
- The specified layer is running.

A layer is enabled by calling the function R_VDC_ReadDataControl. A layer that is in the stopped state can be started by calling the function R_VDC_StartProcess.

Control of the vertical scale-up processing is mutually exclusive with the control of the vertical scale-down processing. Normal operation of the driver operation is not guaranteed if vertical scale-down and vertical scale-up are specified at the same time. The setup of vertical scale-down processing is accomplished by the functions R_VDC_WriteDataControl and R_VDC_ChangeWriteProcess.

(3) Parameter details

The members of the vdc_read_chg_t structure is described below.

```
typedef struct
{
    void                          * gr_base;
    const vdc_width_read_fb_t     * width_read_fb;
    const vdc_period_rect_t       * gr_grc;
    const vdc_gr_disp_sel_t       * gr_disp_sel;
} vdc_read_chg_t;
```

Type Member Name	Description
void * gr_base	Frame buffer base address Specify NULL when this parameter value is not to be changed.
const vdc_width_read_fb_t * width_read_fb	Size of the frame buffer to be read See 6.9(3) for the vdc_width_read_fb_t structure. Specify NULL when this parameter value is not to be changed.
const vdc_period_rect_t * gr_grc	Graphics display area See 5.3(1) for the vdc_period_rect_t structure. Specify NULL when this parameter value is not to be changed.
const vdc_gr_disp_sel_t * gr_disp_sel	Graphics display mode See 5.2(9) and 6.11(3) for details. Specify NULL when this parameter value is not to be changed.

The values of the parameters in the vdc_read_chg_t structure are not changed if NULL is specified for the parameters. The parameters in gr_base, width_read_fb and gr_grc retain the values that have been set up by the function R_VDC_ReadDataControl. The parameter in gr_disp_sel retains the value that has been set up by the function R_VDC_StartProcess.

6.11 R_VDC_StartProcess

Synopsis	Data write/read start processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_StartProcess(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_start_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_ALL: All enabled layers — VDC_LAYER_ID_0_WR: Layer 0 write processing — VDC_LAYER_ID_0_RD: Layer 0 read processing — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing • vdc_start_t * param: Data write/read start parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs layer start processing. If the layer ID specified in layer_id is VDC_LAYER_ID_ALL, the function starts all the layers that are in the stopped state and also enabled. If the layer ID is not VDC_LAYER_ID_ALL, the function starts only the specified layer.

When performing start processing for write, the function starts a write to the frame buffer. When performing start processing for read, the function starts a read from the frame buffer and sets the graphics display mode to the specified values for each layer.

(2) Use conditions

No particular use conditions are imposed if layer_id is found to be set to VDC_LAYER_ID_ALL when the function is used. In the other cases, the conditions listed below apply. If these conditions are not met, the function returns a layer resource error (VDC_ERR_RESOURCE_LAYER).

- The specified layer is enabled.
- The specified layer is stopped.

Layers involved in write processing are enabled by calling the function R_VDC_WriteDataControl and layers involved in read processing are enabled by calling the function R_VDC_ReadDataControl. Layers that are running can be stopped by calling the function R_VDC_StopProcess.

(3) Parameter details

The members of the vdc_start_t structure is described below.

```
typedef struct
{
    const vdc_gr_disp_sel_t      * gr_disp_sel;
} vdc_start_t;
```

Type Member Name	Description
const vdc_gr_disp_sel_t * gr_disp_sel	Graphics display mode See 5.2(9) and the following description for details. Specify NULL when this parameter value is not to be changed. This setting pertains to the graphics (read processing). This setting is invalid if a layer for write processing is specified in layer_id.

See the sample settings given below for the setup of gr_disp_sel.

1. When specifying VDC_LAYER_ID_ALL in layer_id
   Specify the graphics display settings for all the graphics (layers for read processing).
   Specify VDC_DISPSEL_IGNORED for layers that need no change.

```
1       vdc_error_t            error;
2       vdc_gr_disp_sel_t      gr_disp_sel[VDC_GR_TYPE_NUM];
3       vdc_start_t            start;
4
5       gr_disp_sel[VDC_GR_TYPE_GR0] = VDC_DISPSEL_IGNORED;
6       gr_disp_sel[VDC_GR_TYPE_GR2] = VDC_DISPSEL_CURRENT;
7       gr_disp_sel[VDC_GR_TYPE_GR3] = VDC_DISPSEL_BLEND;
8
9       start.gr_disp_sel             = gr_disp_sel;
10
11      error = R_VDC_StartProcess(VDC_CHANNEL_0, VDC_LAYER_ID_ALL, &start);
```

2. When specifying a single layer in layer_id
   Specify the graphics display mode only for the specified graphics (layer for read processing). Given below are example settings for graphics 2 (read process for layer 2).

```
1       vdc_error_t             error;
2       vdc_gr_disp_sel_t       gr_disp_sel;
3       vdc_start_t             start;
4
5       gr_disp_sel         = VDC_DISPSEL_CURRENT;
6
7       start.gr_disp_sel   = &gr_disp_sel;
8
9       error = R_VDC_StartProcess(VDC_CHANNEL_0, VDC_LAYER_ID_2_RD, &start);
```

The graphics display mode for the layers are initialized by the driver when calling the function R_VDC_DisplayOutput, R_VDC_ReadDataControl, and R_VDC_StopProcess. If they are not changed through the function R_VDC_StartProcess, the initial values that are set up by the driver are retained. Table 6-5 shows the initial values of the graphics display mode for the layers.

Table 6-5 Graphics Display Mode Initial Values

Layer ID	Initial Value	Description
VDC_LAYER_ID_0_RD	VDC_DISPSEL_BACK	Displays background color if not used.
VDC_LAYER_ID_2_RD	VDC_DISPSEL_LOWER	Displays the lower layers if not used.
VDC_LAYER_ID_3_RD	VDC_DISPSEL_LOWER	Displays the lower layers if not used.

For the graphics display mode, VDC_DISPSEL_BACK is specified to display background color, and VDC_DIPSEL_CURRENT is specified to display the graphics. Other values are specified depending on the layer and its purpose of use.

Table 6-6 Graphics Display Mode and Uses

Layer ID	Value	Use
VDC_LAYER_ID_0_RD	VDC_DISPSEL_LOWER	• Displays the input video image. • Displays the enlarged graphics.
	VDC_DISPSEL_BLEND	• Displays the graphics processed by the chroma-key.
VDC_LAYER_ID_2_RD / VDC_LAYER_ID_3_RD	VDC_DISPSEL_LOWER	• Displays the lower-layer graphics.
	VDC_DISPSEL_BLEND	• Displays the blended image of lower-layer graphics and current graphics.

6.12 R_VDC_StopProcess

Synopsis	Data write/read stop processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_StopProcess(const vdc_channel_t ch, const vdc_layer_id_t layer_id);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_ALL: All enabled layers — VDC_LAYER_ID_0_WR: Layer 0 write processing — VDC_LAYER_ID_0_RD: Layer 0 read processing — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs layer stop processing. If the layer ID specified in layer_id is VDC_LAYER_ID_ALL, the function stops all the layers that are enabled and running. If the layer ID is not VDC_LAYER_ID_ALL, the function stops only the specified layer.

When performing stop processing for write, the function stops the write to the frame buffer. When performing stop processing for read, the function stops the read from the frame buffer and resets the graphics display mode to the initial values for each of the layers. See Table 6-5 for the initial values of the graphics display mode.

(2) Use conditions

No particular use conditions are imposed if layer_id is found to be set to VDC_LAYER_ID_ALL when the function is used. In the other cases, the conditions listed below apply. If these conditions are not met, the function returns a layer resource error (VDC_ERR_RESOURCE_LAYER).

- The specified layer is enabled.
- The specified layer is running.

Layers involved in write processing are enabled by calling the function R_VDC_WriteDataControl and layers involved in read processing are enabled by calling the function R_VDC_ReadDataControl. Layers that are in the stopped state can be started by calling the function R_VDC_StartProcess.

6.13 R_VDC_ReleaseDataControl

Synopsis	Data write/read control release processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_ReleaseDataControl(const vdc_channel_t ch, const vdc_layer_id_t layer_id);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_ALL: All enabled layers — VDC_LAYER_ID_0_WR: Layer 0 write processing — VDC_LAYER_ID_0_RD: Layer 0 read processing — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following processing:

- Disables the specified layer.

If the layer ID specified in layer_id is VDC_LAYER_ID_ALL, the function disables all the layers that are not running and also enabled. If the layer ID is not VDC_LAYER_ID_ALL, the function disables only the specified layers.

(2) Use conditions

No particular use conditions are imposed if layer_id is found to be set to VDC_LAYER_ID_ALL when the function is used. In the other cases, the conditions listed below apply. If these conditions are not met, the function returns a layer resource error (VDC_ERR_RESOURCE_LAYER).

- The specified layer is enabled.
- The specified layer is stopped.

Layers involved in write processing are enabled by calling the function R_VDC_WriteDataControl and layers involved in read processing are enabled by calling the function R_VDC_ReadDataControl. Layers that are running can be stopped by calling the function R_VDC_StopProcess.

6.14 R_VDC_VideoNoiseReduction

Synopsis	Noise reduction setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_VideoNoiseReduction(const vdc_channel_t ch, const vdc_onoff_t nr1d_on, const vdc_noise_reduction_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_onoff_t nr1d_on: Noise reduction ON/OFF setting • vdc_noise_reduction_t * param: Noise reduction setup parameter The setting is not changed if NULL is specified. If this parameter has never been set up after a hardware reset, the initial value that is defined in the hardware manual remains valid. See the description about the structure for the initial value.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_RESOURCE_INPUT: Input signal resource error

Details

(1) Function

This function performs the following noise reduction processing:

- Turns on and off noise reduction processing.
- Sets up the noise reduction parameters for the Y/G, Cb/B, and Cr/R signals.

The setup of noise reduction parameters and noise reduction ON/OFF control can be made separately. Once set up noise reduction parameters remain valid until a hardware reset occurs or they are overwritten by this function with other settings.

(2) Use conditions

Before this function is used, it is necessary to enable a video input by calling the function R_VDC_VideoInput. If no video input is enabled, the function returns an input signal resource error (VDC_ERR_RESOURCE_INPUT).

(3) Parameter details

The members of the vdc_noise_reduction_t are described below.

```
typedef struct
{
    vdc_nr_param_t     y;
    vdc_nr_param_t     cb;
    vdc_nr_param_t     cr;
} vdc_noise_reduction_t;
```

Type Member Name	Description
vdc_nr_param_t y	Y/G signal noise reduction parameter
vdc_nr_param_t cb	Cb/B signal noise reduction parameter
vdc_nr_param_t cr	Cr/R signal noise reduction parameter

The members of the vdc_nr_param_t structure is described below.

```
typedef struct
{
    vdc_nr_tap_t   nr1d_tap;
    uint32_t        nr1d_th;
    vdc_nr_gain_t  nr1d_gain;
} vdc_nr_param_t;
```

Type Member Name	Initial Value	Description
vdc_nr_tap_t nr1d_tap	0	TAP select • VDC_NR_TAPSEL_1 (0): Adjacent pixel • VDC_NR_TAPSEL_2 (1): 2 adjacent pixels • VDC_NR_TAPSEL_3 (2): 3 adjacent pixels • VDC_NR_TAPSEL_4 (3): 4 adjacent pixels
uint32_t nr1d_th	8	Maximum value of coring (absolute value) 0x0000 to 0x007F
vdc_nr_gain_t nr1d_gain	3	Noise reduction gain adjustment • VDC_NR_GAIN_1_2 (0): 1/2 • VDC_NR_GAIN_1_4 (1): 1/4 • VDC_NR_GAIN_1_8 (2): 1/8 • VDC_NR_GAIN_1_16 (3): 1/16

6.15 R_VDC_ImageColorMatrix

Synopsis	Color matrix setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_ImageColorMatrix(const vdc_channel_t ch, const vdc_color_matrix_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_color_matrix_t * param: Color matrix setup parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_CONDITION: Unauthorized condition error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function sets up the specified color matrix. The VDC incorporated into the RZ/A2M has two color matrixes (see Figure 6-5).

The color matrixes automatically set up by the VDC driver according to the color format to be used. Consequently, this function need not be used except when there is a need to change color matrix values dynamically. The automatic setup of the color matrixes by the VDC driver proceeds as follows:

- When the function R_VDC_WriteDataControl is called
  — The driver determines the necessary color conversion from the input format of the video image and the frame buffer video-signal writing format that is designated by the function R_VDC_WriteDataControl and sets it up in the color matrix in the input controller.
- When the function R_VDC_ReadDataControl is called for layer 0 read processing
  — The driver determines the necessary color conversion from the format of the frame buffer read signal designated by the function R_VDC_ReadDataControl and sets it up in the color matrix in image quality improver 0.

See 5.5 for the values of the color matrixes that are automatically set up by the VDC driver.

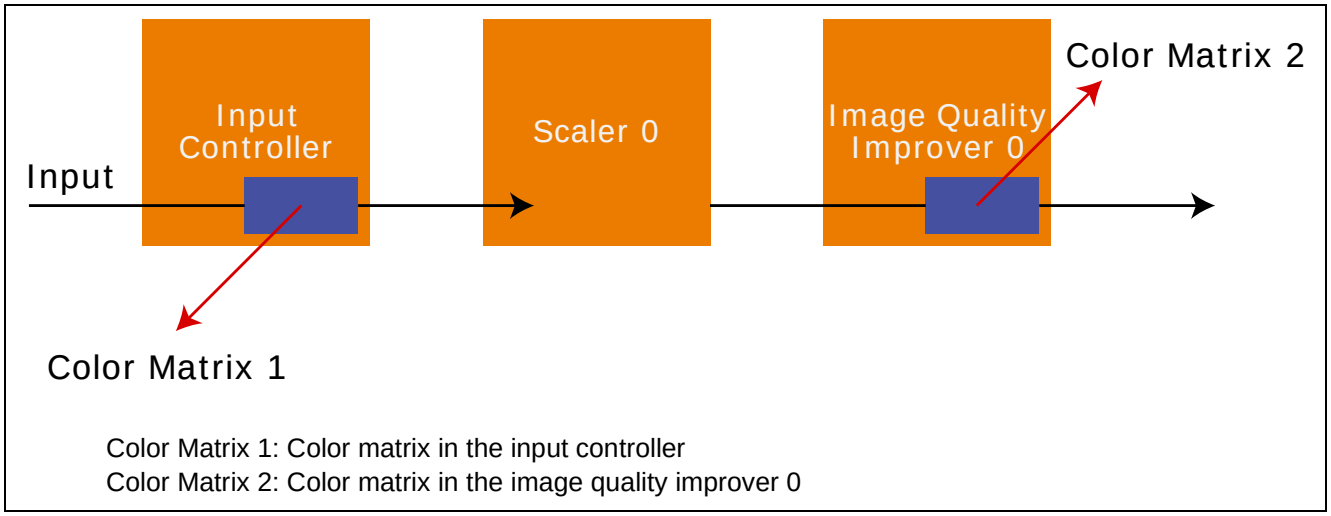


Figure 6-5 Color Matrixes

(2) Use conditions

When this function is to be used to make settings for the color matrix in the input controller, the write process for layer 0 needs to be enabled. The write process for layer 0 is enabled by calling the function R_VDC_WriteDataControl.

When this function is to be used to make settings for the color matrix in the image quality improver 0, the read process for layer 0 needs to be enabled. The read process for layer 0 is enabled by calling the function R_VDC_ReadDataControl.

The function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if it is called when the layers associated with the color matrix are disabled.

(3) Parameter details

The members of the vdc_color_matrix_t structure is described below.

```
typedef struct
{
    vdc_colormtx_module_t  module;
    vdc_colormtx_mode_t    mtx_mode;
    uint16_t                offset[VDC_COLORMTX_OFFST_NUM];
    uint16_t                gain[VDC_COLORMTX_GAIN_NUM];
} vdc_color_matrix_t;
```

Type Member Name	Description
vdc_colormtx_module_t module	Color matrix module • VDC_COLORMTX_IMGCNT (0): Input controller • VDC_COLORMTX_ADJ_0 (1): Image quality improver 0
vdc_colormtx_mode_t mtx_mode	Operating mode • VDC_COLORMTX_GBR_GBR: GBR → GBR • VDC_COLORMTX_GBR_YCBCR: GBR → YCbCr * • VDC_COLORMTX_YCBCR_GBR: YCbCr → GBR • VDC_COLORMTX_YCBCR_YCBCR: YCbCr → YCbCr *
uint16_t offset[VDC_COLORMTX_OFFST_NUM]	Offset (DC) adjustment of Y/G, B, and R signal 0x0000 (-128) to 0x0080 (0) to 0x00FF (+127)
uint16_t gain[VDC_COLORMTX_GAIN_NUM]	GG, GB, GR, BG, BB, BR, RG, RB, and RR signal gain adjustment Signed (2's complement) -1024 to +1023[LSB], 256[LSB] = 1.0[times]

Note: The operating mode in which conversion to YCbCr is performed is made available only when the input controller (VDC_COLORMTX_IMGCNT) is specified in module.

vdc_colormtx_offset_t is an enumeration type for representing the color matrix offset.

```
typedef enum
{
    VDC_COLORMTX_OFFST_YG = 0,
    VDC_COLORMTX_OFFST_B,
    VDC_COLORMTX_OFFST_R,
    VDC_COLORMTX_OFFST_NUM
} vdc_colormtx_offset_t;
```

Enumeration Constant	Value	Description
VDC_COLORMTX_OFFST_YG	0	Offset (DC) adjustment of Y/G signal
VDC_COLORMTX_OFFST_B	1	Offset (DC) adjustment of B signal
VDC_COLORMTX_OFFST_R	2	Offset (DC) adjustment of R signal
VDC_COLORMTX_OFFST_NUM	3	Number of color matrix offset parameters

vdc_colormtx_gain_t is an enumeration type for representing the color matrix gain.

```
typedef enum
{
    VDC_COLORMTX_GAIN_GG = 0,
    VDC_COLORMTX_GAIN_GB,
    VDC_COLORMTX_GAIN_GR,
    VDC_COLORMTX_GAIN_BG,
    VDC_COLORMTX_GAIN_BB,
    VDC_COLORMTX_GAIN_BR,
    VDC_COLORMTX_GAIN_RG,
    VDC_COLORMTX_GAIN_RB,
    VDC_COLORMTX_GAIN_RR,
    VDC_COLORMTX_GAIN_NUM
} vdc_colormtx_gain_t;
```

Enumeration constant	Value	Description
VDC_COLORMTX_GAIN_GG	0	Y/G signal gain adjustment for Y/G signal output
VDC_COLORMTX_GAIN_GB	1	Cb/B signal gain adjustment for Y/G signal output
VDC_COLORMTX_GAIN_GR	2	Cr/R signal gain adjustment for Y/G signal output
VDC_COLORMTX_GAIN_BG	3	Y/G signal gain adjustment for Cb/B signal output
VDC_COLORMTX_GAIN_BB	4	Cb/B signal gain adjustment for Cb/B signal output
VDC_COLORMTX_GAIN_BR	5	Cr/R signal gain adjustment for Cb/B signal output
VDC_COLORMTX_GAIN_RG	6	Y/G signal gain adjustment for Cr/R signal output
VDC_COLORMTX_GAIN_RB	7	Cb/B signal gain adjustment for Cr/R signal output
VDC_COLORMTX_GAIN_RR	8	Cr/R signal gain adjustment for Cr/R signal output
VDC_COLORMTX_GAIN_NUM	9	Number of color matrix gain parameters

6.16 R_VDC_ImageEnhancement

Synopsis: Image enhancement processing

Header: r_vdc.h

Declaration:

```
vdc_error_t R_VDC_ImageEnhancement(
        const vdc_channel_t             ch,
        const vdc_imgimprv_id_t         imgimprv_id,
        const vdc_onoff_t               shp_h_on,
        const vdc_enhance_sharp_t       * const sharp_param,
        const vdc_onoff_t               lti_h_on,
        const vdc_enhance_lti_t         * const lti_param,
        const vdc_period_rect_t         * const enh_area);
```

Arguments:

- vdc_channel_t ch: Channel
  - VDC_CHANNEL_0: Channel 0

  Specify channel 0 in this driver.
- vdc_imgimprv_id_t imgimprv_id: Image quality improver ID
  - VDC_IMG_IMPRV_0: Image quality improver 0

  Specify image quality improver 0 in this driver.
- vdc_onoff_t shp_h_on: Sharpness ON/OFF setting
- vdc_enhance_sharp_t * sharp_param: Sharpness setup parameter
  The setting is not changed if NULL is specified.
- vdc_onoff_t lti_h_on: LTI ON/OFF setting
- vdc_enhance_lti_t * lti_param: LTI setup parameter
  The setting is not changed if NULL is specified.
- vdc_period_rect_t * enh_area: Enhancer-enabled area setup parameter
  The setting is not changed if NULL is specified.

  If parameters described above have never been set up after a hardware reset, the initial values that are defined in the hardware manual remain valid. See the description about the structure for the initial value.

Return value:

- vdc_error_t: Error code
  - VDC_OK: Normal termination
  - VDC_ERR_PARAM_CHANNEL: Channel invalid error
  - VDC_ERR_PARAM_BIT_WIDTH: Bit width error
  - VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error
  - VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error
  - VDC_ERR_IF_CONDITION: Interface condition error
  - VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following image quality improvement processing:

- Turns on and off sharpness processing.
- Sets up the sharpness parameter.
- Turns on and off LTI processing.
- Sets up the LTI parameter.
- Sets up the enhancer-enabled area to be subjected to sharpness and LTI processing.

The setup of parameters and ON/OFF control for sharpness and LTI processing can be made separately. The parameters that are once set up remain valid until a hardware reset occurs or they are overwritten by this function with other settings.

(2) Use conditions

When this function is to be used, the read process for layer 0 needs to be enabled. The read process for layer 0 is enabled by calling the function R_VDC_ReadDataControl for layer 0. The function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the read process for layer 0 is disabled.

This processing is inhibited if the color format (format of the frame buffer read signal) of the layer 0 is set to the format other than YCbCr422 and YCbCr444. In such a case, the function returns an interface condition error (VDC_ERR_IF_CONDITION).

(3) Parameter details

vdc_enhance_sharp_t structure is described below.

```
typedef struct
{
    vdc_onoff_t             shp_h2_lpf_sel;
    vdc_sharpness_ctrl_t    hrz_sharp[VDC_IMGENH_SHARP_NUM];
} vdc_enhance_sharp_t;
```

Type Member Name	Initial Value	Description
vdc_onoff_t shp_h2_lpf_sel	VDC_OFF (0)	LPF selection for folding prevention before H2 edge detection • VDC_OFF: LPF not selected • VDC_ON: LPF selected
vdc_sharpness_ctrl_t hrz_sharp [VDC_IMGENH_SHARP_NUM]	-	Sharpness control parameter Horizontal sharpness (H1, H2, H3)

vdc_img_enh_sh_t is an enumeration type for representing the sharpness band.

```
typedef enum
{
    VDC_IMGENH_SHARP_H1 = 0,
    VDC_IMGENH_SHARP_H2,
    VDC_IMGENH_SHARP_H3,
    VDC_IMGENH_SHARP_NUM
} vdc_img_enh_sh_t;
```

Enumeration constant	Value	Description
VDC_IMGENH_SHARP_H1	0	Horizontal sharpness (H1)
VDC_IMGENH_SHARP_H2	1	Horizontal sharpness (H2)
VDC_IMGENH_SHARP_H3	2	Horizontal sharpness (H3)
VDC_IMGENH_SHARP_NUM	3	Number of horizontal sharpness bands

The members of the vdc_sharpness_ctrl_t structure is described below.

```
typedef struct
{
    uint8_t     shp_clip_o;
    uint8_t     shp_clip_u;
    uint8_t     shp_gain_o;
    uint8_t     shp_gain_u;
    uint8_t     shp_core;
} vdc_sharpness_ctrl_t;
```

Type Member Name	Initial Value	Description
uint8_t shp_clip_o	0	Sharpness correction value clipping (on the overshoot side) 0x0000 to 0x00FF
uint8_t shp_clip_u	0	Sharpness correction value clipping (on the undershoot side) 0x0000 to 0x00FF
uint8_t shp_gain_o	0	Sharpness edge amplitude value gain (on the overshoot side) 0x0000 (0 time) to 0x0040 (1 time) to 0x00FF (approx. 4 times)
uint8_t shp_gain_u	0	Sharpness edge amplitude value gain (on the undershoot side) 0x0000 (0 time) to 0x0040 (1 time) to 0x00FF (approx. 4 times)
uint8_t shp_core	0	Active sharpness range 0x0000 to 0x007F

The members of the vdc_enhance_lti_t structure is described below.

```
typedef struct
{
    vdc_onoff_t             lti_h2_lpf_sel;
    vdc_lti_mdfil_sel_t     lti_h4_median_tap_sel;
    vdc_lti_ctrl_t          lti[VDC_IMGENH_LTI_NUM];
} vdc_enhance_lti_t;
```

Type Member Name	Initial Value	Description
vdc_onoff_t lti_h2_lpf_sel	VDC_OFF (0)	LPF selection for folding prevention before H2 edge detection • VDC_OFF: LPF not selected • VDC_ON: LPF selected
vdc_lti_mdfil_sel_t lti_h4_median_tap_sel	0	Median filter reference pixel select • VDC_LTI_MDFIL_SEL_ADJ2 (0): Second adjacent pixel selected as reference • VDC_LTI_MDFIL_SEL_ADJ1 (1): Adjacent pixel selected as reference
vdc_lti_ctrl_t lti[VDC_IMGENH_LTI_NUM]	-	LTI control parameter Horizontal LTI (H2, H4)

vdc_img_enh_lti_t is an enumeration type for representing the LTI band.

```
typedef enum
{
    VDC_IMGENH_LTI1 = 0,
    VDC_IMGENH_LTI2,
    VDC_IMGENH_LTI_NUM
} vdc_img_enh_lti_t;
```

Enumeration constant	Value	Description
VDC_IMGENH_LTI1	0	Horizontal LTI (H2) Second adjacent pixel used as reference
VDC_IMGENH_LTI2	1	Horizontal LTI (H4) Fourth adjacent pixel used as reference
VDC_IMGENH_LTI_NUM	2	Number of LTI bands.

The members of the vdc_lti_ctrl_t structure is described below.

```
typedef struct
{
    uint8_t     lti_inc_zero;
    uint8_t     lti_gain;
    uint8_t     lti_core;
} vdc_lti_ctrl_t;
```

Type Member Name	Initial Value	Description
uint8_t lti_inc_zero	10	Median filter LTI correction threshold 0x0000 to 0x00FF
uint8_t lti_gain	0	LTI edge amplitude value gain 0x0000 (0 time) to 0x0040 (1 time) to 0x00FF (approx. 4 times)
uint8_t lti_core	0	LTI coring 0x0000 to 0x00FF

See also 5.3(1) for the vdc_period_rect_t structure.

Type Member Name	Initial Value	Description
uint16_t vs	0	Start position of vertical valid image area in enhancer-enabled area (lines) Set to 2 or greater lines.
uint16_t vw	0	Width of vertical valid image area in enhancer-enabled area (lines)
uint16_t hs	0	Start position of horizontal valid image area in enhancer-enabled area (clock cycles) Set to 4 or greater clocks.
uint16_t hw	0	Width of horizontal valid image area in enhancer-enabled area (clock cycles)

6.17 R_VDC_ImageBlackStretch

Synopsis	Black stretch setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_ImageBlackStretch(const vdc_channel_t ch, const vdc_imgimprv_id_t imgimprv_id, const vdc_onoff_t bkstr_on, const vdc_black_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_imgimprv_id_t imgimprv_id: Image quality improver ID — VDC_IMG_IMPRV_0: Image quality improver 0 Specify image quality improver 0 in this driver. • vdc_onoff_t bkstr_on: Black stretch ON/OFF setting • vdc_black_t * param: Black stretch setup parameter The setting is not changed if NULL is specified. If this parameter has never been set up after a hardware reset, the initial value that is defined in the hardware manual remains valid. See the description about the structure for the initial value.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_IF_CONDITION: Interface condition error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following black stretch processing for the specified image quality improver:

- Turns on and off black stretch processing.
- Sets up the black stretch parameters.

The setup of parameters and ON/OFF control of the black stretch processing can be made separately. The settings once established by this function remain valid until a hardware reset occurs or they are overwritten by this function with other settings.

(2) Use conditions

When this function is to be used, the read process for layer 0 needs to be enabled. The read process for layer 0 is enabled by calling the function R_VDC_ReadDataControl for layer 0. The function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the read process for layer 0 is disabled.

This processing is inhibited if the color format (format of the frame buffer read signal) of the layer 0 is set to the format other than YCbCr422 and YCbCr444. In such a case, the function returns an interface condition error (VDC_ERR_IF_CONDITION).

(3) Parameter details

vdc_black_t structure is described below.

```
typedef struct
{
    uint16_t    bkstr_st;
    uint16_t    bkstr_d;
    uint16_t    bkstr_t1;
    uint16_t    bkstr_t2;
} vdc_black_t;
```

Type Member Name	Initial Value	Description
uint16_t bkstr_st	0	Black stretch start point 0 (low) to 15 (high)
uint16_t bkstr_d	0	Black stretch depth 0 (shallow) to 15 (deep)
uint16_t bkstr_t1	0	Black stretch time constant (T1) 0 (small) to 31 (large)
uint16_t bkstr_t2	0	Black stretch time constant (T2) 0 (small) to 30 (large)

6.18 R_VDC_AlphaBlending

Synopsis	Alpha blending setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_AlphaBlending(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_alpha_blending_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing • vdc_alpha_blending_t * param: Alpha blending setup parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following processing for alpha blending except for rectangle alpha blending:

- Sets up the alpha value of the ARGB1555/RGBA5551 formats.
- Make settings for premultiplication processing at alpha blending in one-pixel.

This function can set up the alpha value of ARGB1555/RGBA5551 for layer 2 and 3 read processes. The alpha value ARGB1555/RGBA5551 which is used for layer 0 read process is automatically set to '255 (= 1.0, nontransparent)' by the driver.

(2) Use conditions

When this function is to be used, the layer specified in layer_id needs to be enabled. The layer is enabled by calling and executing the function R_VDC_ReadDataControl on that layer. The function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the layer specified in layer_id is found disabled.

(3) Parameter details

The members of the vdc_alpha_blending_t structure is described below.

```
typedef struct
{
    const vdc_alpha_argb1555_t   * alpha_1bit;
    const vdc_alpha_pixel_t      * alpha_pixel;
} vdc_alpha_blending_t;
```

Type Member Name	Description
const vdc_alpha_argb1555_t * alpha_1bit	Alpha signal of the ARGB1555/RGBA5551 formats The setting is not changed if NULL is specified.
const vdc_alpha_pixel_t * alpha_pixel	Premultiplication processing at alpha blending in one-pixel The setting is not changed if NULL is specified.

If the parameters alpha_1bit and alpha_pixel have never been set up after a hardware reset, their initial values that are defined in the hardware manual remain valid. See the description about the structures for the initial values.

The members of the vdc_alpha_argb1555_t structure is described below.

```
typedef struct
{
    uint8_t     gr_a0;
    uint8_t     gr_a1;
} vdc_alpha_argb1555_t;
```

Type Member Name	Initial Value	Description
uint8_t gr_a0	0	Alpha signal when alpha is set to '0' 0 to 255
uint8_t gr_a1	0	Alpha signal when alpha is set to '1' 0 to 255

The members of the vdc_alpha_pixel_t structure is described below.

```
typedef struct.
{
    vdc_onoff_t    gr_acalc_md;
} vdc_alpha_pixel_t;
```

Type Member Name	Initial Value	Description
vdc_onoff_t gr_acalc_md	VDC_OFF (0)	Premultiplication processing at alpha blending in one-pixel units • VDC_OFF • VDC_ON

6.19 R_VDC_AlphaBlendingRect

Synopsis	Rectangle alpha blending setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_AlphaBlendingRect(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_onoff_t gr_arc_on, const vdc_alpha_blending_rect_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing • vdc_onoff_t gr_arc_on: ON/OFF setting for alpha blending in a rectangular area • vdc_alpha_blending_rect_t * param: Setup parameter for alpha blending in a rectangular area The setting is not changed if NULL is specified. See the description about the structure for the initial value.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following processing for rectangle alpha blending:

- Turns on and off alpha blending in a rectangular area.
- Sets up the rectangular area subjected to alpha blending.
- Sets up the alpha value for alpha blending in a rectangular area.
- Makes fade-in/-out settings to be applied to rectangle alpha blending.

The setup and ON/OFF control of the alpha blending in rectangular area can be made separately. The alpha blending settings once established by this function remain valid until a hardware reset occurs, until overwritten with other settings, or until the specified layer resources are destroyed by the function R_VDC_ReleaseDataControl.

(2) Use conditions

When this function is to be used, the specified layer needs to be enabled. The layer is enabled by calling the function R_VDC_ReadDataControl on that layer. The function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the layer specified in layer_id is found disabled.

(3) Parameter details

The members of the vdc_alpha_blending_rect_t structure is described below.

```
typedef struct
{
    const vdc_pd_disp_rect_t * gr_arc;
    const vdc_alpha_rect_t   * alpha_rect;
    const vdc_scl_und_sel_t  * scl_und_sel;
} vdc_alpha_blending_rect_t;
```

Type Member Name	Description
const vdc_pd_disp_rect_t * gr_arc	Rectangular area subjected to alpha blending The setting is not changed if NULL is specified.
const vdc_alpha_rect_t * alpha_rect	Parameter for alpha blending in a rectangular area The setting is not changed if NULL is specified.
const vdc_scl_und_sel_t * scl_und_sel	Selection of lower-layer plane in scaler This parameter is not referred in this driver. Specify NULL.

The parameter gr_arc is initialized by the driver to the same graphics display area when the API function R_VDC_ReadDataControl is called. The parameter alpha_rect is kept at its initial values that are defined in the hardware manual if it has never been set up after a hardware reset. See the structure descriptions given below for its initial values. The parameters in vdc_alpha_blending_rect_t structure, including the parameters gr_arc and alpha_rect are kept at their initial values if they have never been set up after a hardware reset.

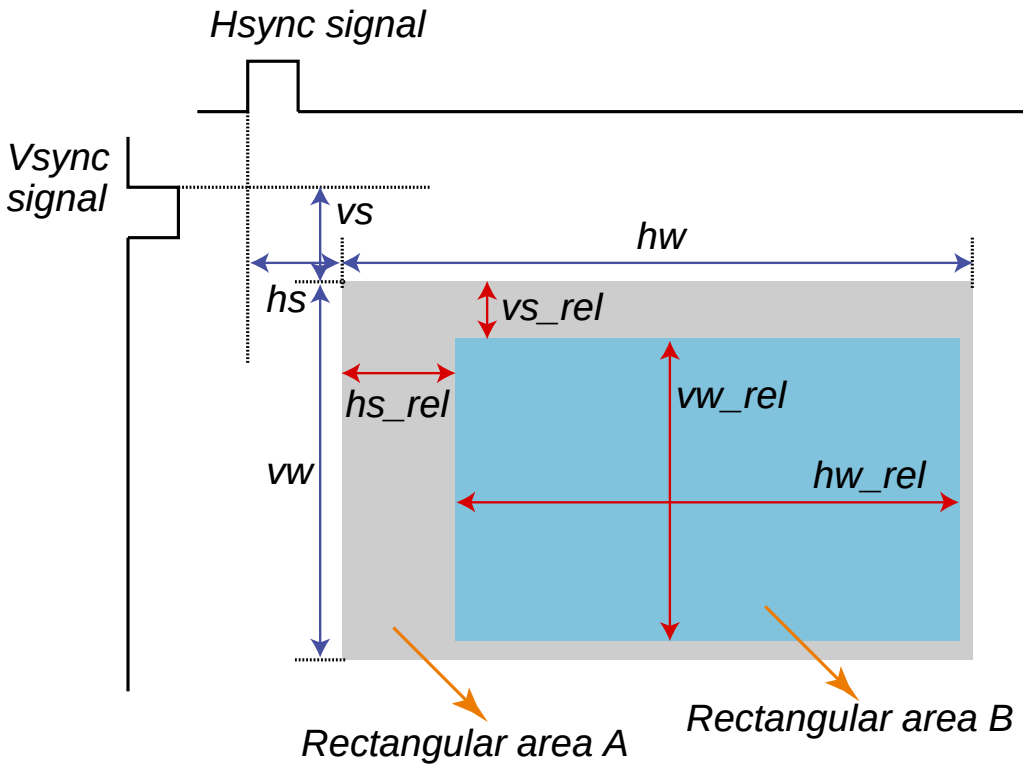
vdc_pd_disp_rect_t structure is described below.

```
typedef struct
{
    uint16_t   vs_rel;
    uint16_t   vw_rel;
    uint16_t   hs_rel;
    uint16_t   hw_rel;
} vdc_pd_disp_rect_t;
```

Type Member Name	Initial Value	Description
uint16_t vs_rel	0	Vertical start position of the valid image area for alpha blending in a rectangular area (lines) This is the relative position from the vertical start position of the graphics display area.
uint16_t vw_rel	- *	Vertical width of the valid image area for alpha blending in a rectangular area (lines)
uint16_t hs_rel	0	Horizontal start position of the valid image area for alpha blending in a rectangular area (clock cycles) This is the relative position from the horizontal start position of the graphics display area.
uint16_t hw_rel	- *	Horizontal width of the valid image area for alpha blending in a rectangular area (clock cycles)

Note: The effective image area vertical width and horizontal width are initialized by the driver to the same values of the graphics display area.

The rectangular area to be subjected to alpha blending is specified as a relative position within the graphics display area for the pertinent layer that is specified through the function R_VDC_ReadDataControl (see Figure 6-6).



Area A: Graphics display area
See 6.9(3) and 5.3(1) for details.
Area B: Rectangular area for alpha blending

Figure 6-6 Rectangular Area Setting for Alpha Blending

Even when the graphics display area is changed by the function R_VDC_ChangeReadProcess, the rectangular area for alpha blending does not follow the change. To change the graphics display area when using alpha blending in a rectangular area, it is also necessary to change the rectangular area for alpha blending.

The members of the vdc_alpha_rect_t structure is described below.

```
typedef struct
{
    int16_t         gr_arc_coef;
    uint8_t         gr_arc_rate;
    uint8_t         gr_arc_def;
    vdc_onoff_t    gr_arc_mul;
} vdc_alpha_rect_t;
```

Type Member Name	Initial Value	Description
int16_t gr_arc_coef	0	Alpha coefficient for alpha blending in a rectangular area Variation (-255 to +255)
uint8_t gr_arc_rate	0	Frame rate for alpha blending in a rectangular area gr_arc_coef is added to the alpha value every time Vsync rises the number of times equal to gr_arc_rate + 1. 0 to 255

uint8_t gr_arc_def	255	Initial alpha value for alpha blending in a rectangular area 0 to 255
vdc_onoff_t gr_arc_mul	VDC_OFF (0)	Multiplication processing with current alpha at alpha blending in a rectangular area • VDC_OFF • VDC_ON

6.20 R_VDC_Chromakey

Synopsis	Chroma-key setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_Chromakey(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_onoff_t gr_ck_on, const vdc_chromakey_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_0_RD: Layer 0 read processing — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing • vdc_onoff_t gr_ck_on: Chroma-key ON/OFF setting • vdc_chromakey_t * param: Chroma-key setup parameter The setting is not changed if NULL is specified. If this parameter has never been set up after a hardware reset, the initial value that is defined in the hardware manual remains valid. See the description about the structure for the initial value.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_IF_CONDITION: Interface condition error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function performs the following chroma-key related processing:

- Turns on and off the chroma-key processing.
- Sets up the color signals to be subject to chroma-key processing and the color signals after replacement.

The setup of chroma-key processing and its ON/OFF control can be made separately. Once set up chroma-key settings remain valid until a hardware reset occurs, until overwritten with other settings, or until the specified layer resources are destroyed by the function R_VDC_ReleaseDataControl.

(2) Use conditions

When this function is to be used, the layer specified in layer_id needs to be enabled. The layer is enabled by calling the function R_VDC_ReadDataControl on that layer. The function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the layer specified in layer_id is found disabled.

The execution of chroma-key processing is inhibited if the color format (format of the frame buffer read signals) of the specified layer is set to YCbCr422 or YCbCr444. In such a case, the function returns an interface condition error (VDC_ERR_IF_CONDITION).

(3) Parameter details

vdc_chromakey_t structure is described below.

```
typedef struct
{
    uint32_t    ck_color;
    uint32_t    rep_color;
    uint8_t     rep_alpha;
} vdc_chromakey_t;
```

Type Member Name	Initial Value	Description
uint32_t ck_color	0	RGB/CLUT signal for RGB-index/CLUT-index chroma-key processing Specify in the color format that is used in the target layer (LSB justified).
uint32_t rep_color	0	Replaced RGB signal after RGB/CLUT-index chroma-key processing Specify in the color format that is used in the target layer (LSB justified). Specify, however, in the RGB888 format if the color format is set to CLUT8, CLUT4, or CLUT1. The alpha value in this parameter is ignored. Specify the replaced alpha signal in rep_alpha.
uint8_t rep_alpha	0 *	Replaced alpha signal after RGB/CLUT-index chroma-key processing Specify an alpha value in 8 bits. 0 ~ 255

Note: The alpha value for layer 0 is automatically set by the driver to '255'.

6.21 R_VDC_CLUT

Synopsis	CLUT setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_CLUT(const vdc_channel_t ch, const vdc_layer_id_t layer_id, const vdc_clut_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_layer_id_t layer_id: Layer ID — VDC_LAYER_ID_0_RD: Layer 0 read processing — VDC_LAYER_ID_2_RD: Layer 2 read processing — VDC_LAYER_ID_3_RD: Layer 3 read processing • vdc_clut_t * param: CLUT setup parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_LAYER_ID: Invalid layer ID error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_EXCEED_RANGE: Out-of-value-range error — VDC_ERR_RESOURCE_LAYER: Layer resource error

Details

(1) Function

This function sets up CLUT for the specified layer.

(2) Use conditions

When this function is to be used, the layer specified in layer_id needs to be enabled. The layer is enabled by calling and executing the function R_VDC_ReadDataControl. The function returns a layer resource error (VDC_ERR_RESOURCE_LAYER) if the layer specified in layer_id is found disabled.

(3) Parameter details

The members of the vdc_clut_t structure is described below.

```
typedef struct
{
    uint32_t            color_num;
    const uint32_t    * clut;
} vdc_clut_t;
```

Type Member Name	Description
uint32_t color_num	Number of colors in CLUT When CLUT1 format is used: 1 to 2 When CLUT4 format is used: 1 to 16 When CLUT8 format is used: 1 to 256
const uint32_t * clut	Address of the area storing the CLUT data (in ARGB8888 format) Do not specify NULL.

6.22 R_VDC_DisplayCalibration

Synopsis	Display calibration processing
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_DisplayCalibration(const vdc_channel_t ch, const vdc_disp_calibration_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_disp_calibration_t * param: Display calibration parameter Do not specify NULL.
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_NULL: NULL specification error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_PARAM_UNDEFINED: Undefined parameter specification error — VDC_ERR_RESOURCE_OUTPUT: Output resource error

Details

(1) Function

This function performs the following processing for display calibration:

- Sets up panel brightness adjustment.
- Sets up contrast adjustment.
- Sets up panel dithering.
- Makes control settings for the correction circuit sequence.

The settings established by this function remain valid until a hardware reset occurs or they are overwritten by this function with other settings.

(2) Use conditions

Before using this function, it is necessary to set up display output by calling the function R_VDC_DisplayOutput. The function returns an output resource error (VDC_ERR_RESOURCE_OUTPUT) if the display output is not set up.

(3) Parameter details

The members of the vdc_disp_calibration_t structure is described below.

```
typedef struct
{
    vdc_calibr_route_t            route;
    const vdc_calibr_bright_t   * bright;
    const vdc_calibr_contrast_t * contrast;
    const vdc_calibr_dither_t   * panel_dither;
} vdc_disp_calibration_t;
```

Type Member Name	Description
vdc_calibr_route_t route	Correction circuit sequence control • VDC_CALIBR_ROUTE_BCG: Brightness → contrast → gamma correction • VDC_CALIBR_ROUTE_GBC: Gamma correction → brightness → contrast
const vdc_calibr_bright_t * bright	Brightness (DC) adjustment parameter Specify NULL when this parameter value need not be changed.
const vdc_calibr_contrast_t * contrast	Contrast (gain) adjustment parameter Specify NULL when this parameter value need not be changed.
const vdc_calibr_dither_t * panel_dither	Panel dithering parameter Specify NULL when this parameter value need not be changed.

If the brightness, contrast, and panel_dither parameters have never been set up after a hardware reset, their initial values that are defined in the hardware manual remain valid. See the structure descriptions given below for their initial values.

The members of the vdc_calibr_bright_t structure is described below.

```
typedef struct
{
    uint16_t    pbrt_g;
    uint16_t    pbrt_b;
    uint16_t    pbrt_r;
} vdc_calibr_bright_t;
```

Type Member Name	Initial Value	Description
uint16_t pbrt_g	512	Brightness (DC) adjustment of G signal 0x0000 (-512) to 0x03FF (+511)
uint16_t pbrt_b	512	Brightness (DC) adjustment of B signal 0x0000 (-512) to 0x03FF (+511)
uint16_t pbrt_r	512	Brightness (DC) adjustment of R signal 0x0000 (-512) to 0x03FF (+511)

The members of the vdc_calibr_contrast_t structure is described below.

```
typedef struct
{
    uint8_t     cont_g;
    uint8_t     cont_b;
    uint8_t     cont_r;
} vdc_calibr_contrast_t;
```

Type Member Name	Initial Value	Description
uint8_t cont_g	128	Contrast (gain) adjustment of G signal 0x0000 (0/128[times]) to 0x00FF (255/128[times])
uint8_t cont_b	128	Contrast (gain) adjustment of B signal 0x0000 (0/128[times]) to 0x00FF (255/128[times])
uint8_t cont_r	128	Contrast (gain) adjustment of R signal 0x0000 (0/128[times]) to 0x00FF (255/128[times])

The members of the vdc_calibr_dither_t structure is described below.

```
typedef struct
{
    vdc_panel_dither_md_t  pdth_sel;
    uint8_t                 pdth_pa;
    uint8_t                 pdth_pb;
    uint8_t                 pdth_pc;
    uint8_t                 pdth_pd;
} vdc_calibr_dither_t;
```

Type Member Name	Initial Value	Description
vdc_panel_dither_md_t pdth_sel	0	Panel dither operation mode • VDC_PDTH_MD_TRU (0): Truncate • VDC_PDTH_MD_RDOF (1): Round-off • VDC_PDTH_MD_2X2 (2): 2x2 pattern dither • VDC_PDTH_MD_RAND (3): Random pattern dither
uint8_t pdth_pa	3	Pattern value (A) of 2x2 pattern dither 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.
uint8_t pdth_pb	0	Pattern value (B) of 2x2 pattern dither 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.
uint8_t pdth_pc	2	Pattern value (C) of 2x2 pattern dither 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.
uint8_t pdth_pd	1	Pattern value (D) of 2x2 pattern dither 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.

6.23 R_VDC_GammaCorrection

Synopsis	Gamma correction setup
Header	r_vdc.h
Declaration	vdc_error_t R_VDC_GammaCorrection(const vdc_channel_t ch, const vdc_onoff_t gam_on, const vdc_gamma_correction_t * const param);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_onoff_t gam_on: Gamma correction ON/OFF setting • vdc_gamma_correction_t * param: Gamma correction setup parameter
Return value	• vdc_error_t: Error code — VDC_OK: Normal termination — VDC_ERR_PARAM_CHANNEL: Channel invalid error — VDC_ERR_PARAM_BIT_WIDTH: Bit width error — VDC_ERR_RESOURCE_OUTPUT: Output resource error

Details

(1) Function

This function performs the following processing for gamma correction:

- Turns on and off gamma correction processing.
- Sets up the gamma correction gain adjustment values for the G/B/R signals.
- Sets up the gamma correction start threshold values for the G/B/R signals.

The setup of the gamma correction parameters and the ON/OFF control of gamma correction processing can be made separately. The gamma correction parameter settings once established by this function remain valid until a hardware reset occurs or they are overwritten with other settings.

(2) Use conditions

Before using this function, it is necessary to set up display output by calling the function R_VDC_DisplayOutput. The function returns an output resource error (VDC_ERR_RESOURCE_OUTPUT) if the display output is not set up.

(3) Parameter details

The members of vdc_gamma_correction_t structure is described below.

```
typedef struct
{
    const uint16_t    * gam_g_gain;
    const uint8_t     * gam_g_th;
    const uint16_t    * gam_b_gain;
    const uint8_t     * gam_b_th;
    const uint16_t    * gam_r_gain;
    const uint8_t     * gam_r_th;
} vdc_gamma_correction_t;
```

Type Member Name	Description
const uint16_t * gam_g_gain	Gain adjustment of area 0 to 31 of G signal Unsigned (0 to 2047[LSB], 1024[LSB] = 1.0[time]) Specify NULL when this parameter value need not be changed.
const uint8_t * gam_g_th	Start threshold of area 1 to 31 of G signal Unsigned (0 to 255[LSB]) Specify NULL when this parameter value need not be changed.
const uint16_t * gam_b_gain	Gain adjustment of area 0 to 31 of B signal Unsigned (0 to 2047[LSB], 1024[LSB] = 1.0[time]) Specify NULL when this parameter value need not be changed.
const uint8_t * gam_b_th	Start threshold of area 1 to 31 of B signal Unsigned (0 to 255[LSB]) Specify NULL when this parameter value need not be changed.
const uint16_t * gam_r_gain	Gain adjustment of area 0 to 31 of R signal Unsigned (0 to 2047[LSB], 1024[LSB] = 1.0[time]) Specify NULL when this parameter value need not be changed.
const uint8_t * gam_r_th	Start threshold of area 1 to 31 of R signal Unsigned (0 to 255[LSB]) Specify NULL when this parameter value need not be changed.

If the parameters have never been set up after a hardware reset, their initial values that are defined in the hardware manual remain valid. The initial values are given below.

- Gain adjustment of area 0 to 31 of G/B/R signal: All 1024 (= 1.0[time])
- Start threshold of area 1 to 31 of G/B/R signal: The start threshold value of area n is n x 8.

6.24 R_VDC_GetISR

Synopsis	Interrupt service routine acquisition processing
Header	r_vdc.h
Declaration	void (*R_VDC_GetISR(const vdc_channel_t ch, const vdc_int_type_t type)) (const uint32_t int_sense);
Arguments	• vdc_channel_t ch: Channel — VDC_CHANNEL_0: Channel 0 Specify channel 0 in this driver. • vdc_int_type_t type: Interrupt type See 5.2(8) for details.
Return value	• void (*)(const uint32_t int_sense): Function pointer to the interrupt service routine — Other than 0: Normal termination — 0: Error

Details

(1) Function

This function returns the function pointer to the specified interrupt service routine.

It returns a '0' if the channel specified in ch or the interrupt type specified in type is found invalid.

(2) Use conditions

There are no particular conditions with respect to the call of this function.

6.25 R_SPEA_WindowOffset

Synopsis	Setting the coordinate offset of the window
Header	r_spea.h
Declaration	spea_error_t R_SPEA_WindowOffset(const vdc_layer_id_t layer_id, const uint16_t offset_x, const uint16_t offset_y);
Arguments	• vdc_layer_id_t layer_id: layer ID — VDC_LAYER_ID_2_RD: layer 2 — VDC_LAYER_ID_3_RD: layer 3 Do not specify VDC_LAYER_ID_0_RD. • uint16_t offset_x: offset_x should be 0 or more and equal to or less than 2047 in 2 [pixel] units. • uint16_t offset_y: offset_y should be 0 or more and equal to or less than 8191.
Return value	• spea_error_t: Error code — SPEA_OK: Normal termination — SPEA_ERR_PARAM_LAYER_ID: Invalid layer ID error — SPEA_ERR_PARAM: Unauthorized condition error

Details

(1) Function

This function performs the following processing related to data read control.

- Set the arrangement of VDC (layers 2 and 3) display area on virtual frame of SPEA.

(2) Use conditions

There are no particular conditions with respect to the call of this function.

(3) Parameter details

In this function, as shown in Figure 6-7, set the arrangement of the display area of VDC on the SPEA virtual frame. This offset is the origin (0, 0) when setting the SPEA Window position.

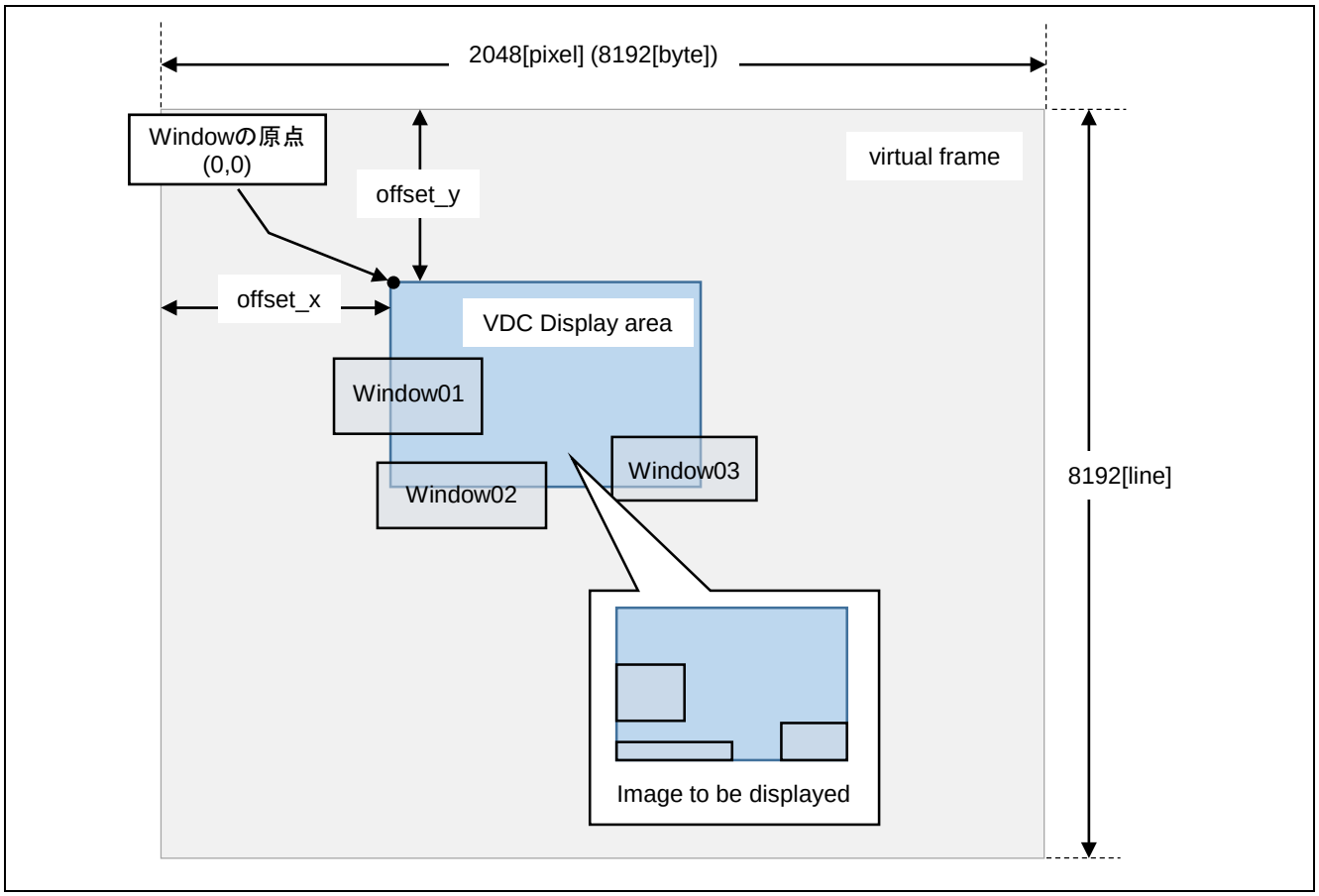


Figure 6-7 Relationship between virtual frame and VDC display area

6.26 R_SPEA_SetWindow

Synopsis	Setting Window Parameters
Header	r_spea.h
Declaration	spea_error_t R_SPEA_SetWindow(const vdc_layer_id_t layer_id, const spea_window_id_t window_id, const spea_onoff_t sken, const spea_sklym_t * sklym, const spea_skpsm_t * skpsm, const void * buffer);
Arguments	• vdc_layer_id_t layer_id: layer ID — VDC_LAYER_ID_2_RD: layer 2 — VDC_LAYER_ID_3_RD: layer 3 Do not specify VDC_LAYER_ID_0_RD. • spea_window_id_t window_id: Window ID — WINDOW_00 – WINDOW_15: Window ID • spea_onoff_t sken: Window ON/OFF — SPEA_ON : — SPEA_OFF : • spea_sklym_t * sklym: Window size — sklym.width be 0 or more and equal to or less than 2047 in 2 [pixel] units. — sklym.height be 0 or more and equal to or less than 8191. • spea_skpsm_t * skpsm: Window start coordinates. — Set skpsm.x in units of 2 [pixels]. An error occurs if the result of adding offset_x set in “R_SPEA_WindowOffset()” to skpsm.x is not 0 or more but 2047 or less. — An error occurs if the result of adding offset_y set in “R_SPEA_WindowOffset()” to skpsm.y is not 0 or more but 8191 or less. • void * buffer : Window read buffer address — Specify a multiple of 8 address.
Return value	• spea_error_t: Error code — SPEA_OK: Normal termination — SPEA_ERR_PARAM_LAYER_ID: Invalid layer ID error — SPEA_ERR_PARAM: Unauthorized condition error

Details

(1) Function

This function performs the following processing related to data read control.

- SPEA Window ON / OFF
- Set window starting coordinates and size and read buffer address to SPEA.
- VDC frame buffer burst transfer mode setting (SPEA_ON: 128 bytes SPEA_OFF: 32 byte transfer)

(2) Use conditions

There are no particular conditions with respect to the call of this function. SPEA operates using layers 2 and 3 of the VDC. Therefore, please also set the VDC.

(3) Sprite layer

When using SPEA, it is necessary to set the layer 2 and layer 3 of the VDC. Please set the following values to the base address of frame buffer used for layer setting of VDC and line offset address of frame buffer.

```
#define VIRTUAL_FRAME_BASE_ADD     (0x30000000u)
#define VIRTUAL_FRAME_STRAID       (8192u)
```

6.27 R_SPEA_WindowUpdate

Synopsis	Window parameter update request
Header	r_spea.h
Declaration	spea_error_t R_SPEA_WindowUpdate(const vdc_layer_id_t layer_id);
Arguments	• vdc_layer_id_t layer_id: layer ID — VDC_LAYER_ID_2_RD: layer 2 — VDC_LAYER_ID_3_RD: layer 3 Do not specify VDC_LAYER_ID_0_RD.
Return value	• spea_error_t: Error code — SPEA_OK: Normal termination — SPEA_ERR_PARAM_LAYER_ID: Invalid layer ID error

Details

(1) Function

This function performs the following processing related to data read control.

- Request to update SPEA Window parameters.

(2) Use conditions

There are no particular conditions with respect to the call of this function.

6.28 R_RLE_SetWindow

Synopsis	Setting the RLE parameter
Header	r_spea.h
Declaration	spea_error_t R_RLE_SetWindow(const vdc_error_t layer_id, const rle_onoff_t sken, const rle_cfg_t * rle_cfg, const void * buffer)
Arguments	• vdc_layer_id_t layer_id: layer ID — VDC_LAYER_ID_0_RD: layer 0 Do not specify VDC_LAYER_ID_2_RD and VDC_LAYER_ID_3_RD. • rle_onoff_t sken: RLE ON / OFF — RLE_ON : — RLE_OFF : • rle_cfg_t * rle_cfg: Set NULL(TBD). • void * buffer : Window read buffer address — Specify a multiple of 8 address. and use Targa format image file with header removed.
Return value	• spea_error_t: Error code — SPEA_OK: Normal termination — SPEA_ERR_PARAM_LAYER_ID: Invalid layer ID error — SPEA_ERR_PARAM: Unauthorized condition error

Details

(1) Function

This function performs the following processing related to data read control.

- RLE ON /OFF .
- Setting SPEA RLE parameters.

(2) Use conditions

There are no particular conditions with respect to the call of this function. RLE operates using layers 0 of the VDC. Therefore, please also set the VDC.

(3) RLE (run-length encoded) layer

RLE unit operates using layer 0 of VDC. Data compressed in Targa format can be expanded and displayed. It is possible to realize high resolution display with less memory. However, the compression ratio of data depends on the image.

(4) About creating image in Targa format.

Please use graphics editor tool such as GIMP to create Targa format image file.(Check operation with GIMP 2.8)

(5) Restrictions on image creation.

Targa format images used for RLE units, the following restrictions must be observed.

Note 1. If an image data size in the horizontal direction does not fit to 128 bytes alignment, some dummy data in each line needs to be added.

2. After completion of the reading of each line, the RLE unit reads additional 128 bytes. For this reason, 128 bytes of additional dummy data must be inserted in each line.

3. Color data format: 24 bpp (Alpha channel not included).

Example : Dummy data insertion required for each line in the horizontal 240 pixel image of 32bpp (Figure 6-8):
64- and 128-byte of dummy data are required to satisfy the restrictions of the Note 1 and Note 2, respectively.Dummy data insertion of 192 bytes in total is required. The number of horizontal pixels after dummy data insertion is 288 pixels.

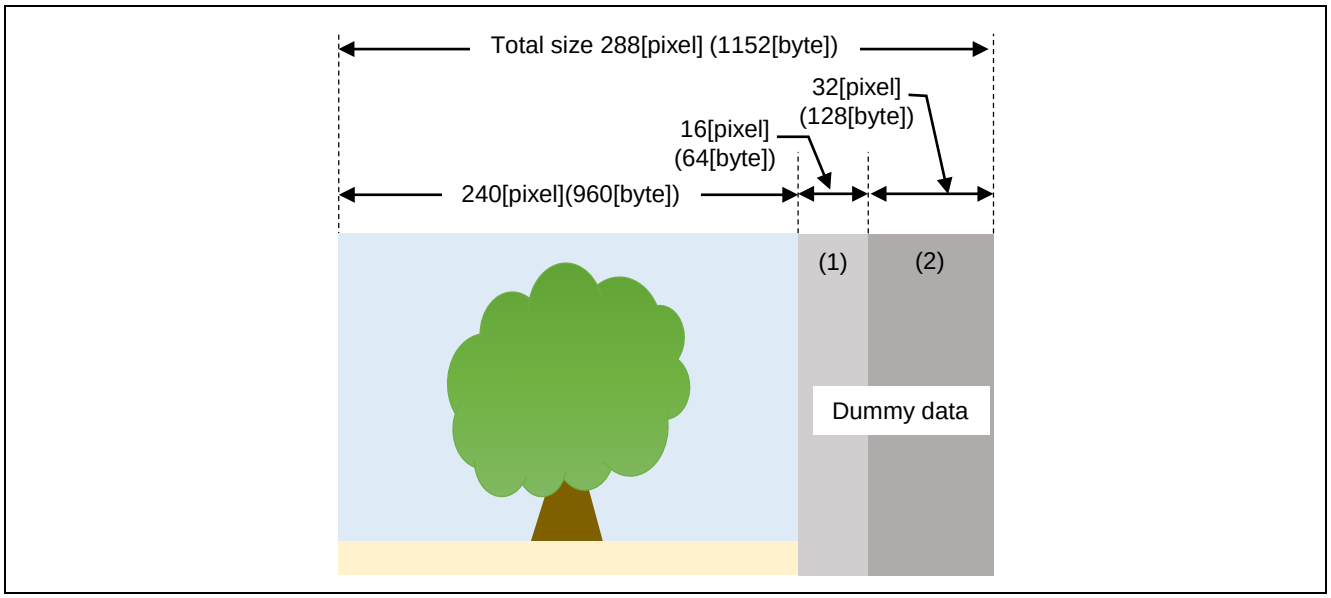


Figure 6-8 About dummy data

(6) Targa image file header.

Specify Targa format image file with header removed. The header of Targa format is fixed at 18 [byte]. Please use the data after that.

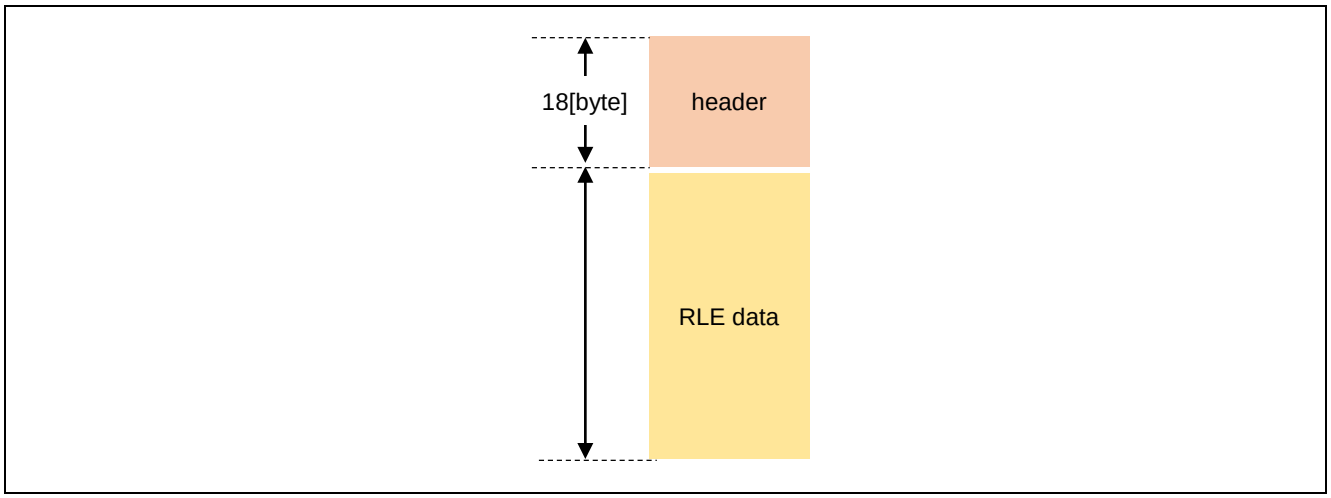


Figure 6-9 Targa image file header.

6.29 R_RLE_WindowUpdate

Synopsis	RLE parameter update request
Header	r_spea.h
Declaration	spea_error_t R_RLE_WindowUpdate(const vdc_layer_id_t layer_id);
Arguments	• vdc_layer_id_t layer_id: layer ID — VDC_LAYER_ID_0_RD: layer 0 Do not specify VDC_LAYER_ID_2_RD and VDC_LAYER_ID_3_RD.
Return value	• spea_error_t: Error code — SPEA_OK: Normal termination — SPEA_ERR_PARAM_LAYER_ID: Invalid layer ID error

Details

(1) Function

This function performs the following processing related to data read control.

- Request to update RLE parameters.

(2) Use conditions

There are no particular conditions with respect to the call of this function.

7. How to Import the Driver

7.1 e² studio

Please refer to the RZ/A2M Smart Configurator User's Guide: e² studio R20AN0583EJ for details on how to import drivers into projects in e2 studio using the Smart Configurator tool.

7.2 For Projects created outside e² studio

This section describes how to import the driver into your project.

Generally, there are two steps in any IDE:

1) Copy the driver to the location in the source tree that you require for your project.

2) Add the link to where you copied your driver to the compiler.

Other required drivers, e.g. r_cbuffer, must be imported similarly.

8. 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 (Revision: r4p1) Technical Reference Manual

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 (Revision: r3p3) Technical Reference Manual

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.

Revision History

Rev.	Date	Description	
		Page	Summary
1.11	Sep.30.20	p8	Table 4-1 title correction ""Stream It! RZ V2.0 Board connection""->"RZ/A2M Evalution Board connection"
		p8	Modified the description of “TXOUT2M/P, TXOUT1M/P, TXOUT0M/P” in Table 4-1.
1.00	Sep.14.18	-	First edition issued
1.01	Dec.28.18	97	Modified the parameter “spea_onoff_t”->”rle_onoff_t”.
		all	Modified the spelling mistake “VDC 5” -> “VDC”.
1.02	Apr.15.19	64	Modified the MCU mistake “RZA1L” -> “RZ/A2M”.
1.10	May.17.19	6	Table 8.1 Peripheral device used(1/2) Remove compiler option "-mthumb-interwork"
		101	Added “7.How to Import the Driver”

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/.