RZ/A2M Group

Video Utility

Introduction

This document describes the functional specification of Renesas Video Application Interface (RVAPI) for a RZ/A series RZ/A2M group MCU.

Target Device

RZ/A2M

Contents

1. Specifications 4

2. Operating environment 5

3. Reference Application Notes 7

4. Hardware Description 8
4.1 List of Pins That are Used 8

5. Software Description....9
5.1 Functions 9

6. Function Reference (VDC)....11
6.1 R_RVAPI_InitializeVDC 11
6.2 R_RVAPI_TerminateVDC 15
6.3 R_RVAPI_DispControlVDC.... 16
6.4 R_RVAPI_GraphCreateSurfaceVDC 19
6.5 R_RVAPI_GraphChangeSurfaceVDC 23
6.6 R_RVAPI_GraphChangeSurfaceConfigVDC 24
6.7 R_RVAPI_GraphDestroySurfaceVDC.... 27
6.8 R_RVAPI_DispPortSettingVDC 28
6.9 R_RVAPI_VideoControlVDC.... 29
6.10 R_RVAPI_VideoCreateSurfaceVDC 32
6.11 R_RVAPI_VideoCreateSurfaceIMRLS2.... 32
6.12 R_RVAPI_VideoDestroySurfaceVDC 39
6.13 R_RVAPI_VideoPortSettingVDC 40
6.14 R_RVAPI_InterruptEnableVDC.... 41
6.15 R_RVAPI_InterruptDisableVDC 43
6.16 R_RVAPI_AlphablendingRectVDC 44
6.17 R_RVAPI_ChromakeyVDC 45
6.18 R_RVAPI_DispCalibrationVDC 46
6.19 R_RVAPI_DispGammaVDC 49
6.20 R_RVAPI_VideoCalibrationVDC 50
6.21 R_RVAPI_VideoSharpnessLtiVDC 52
6.22 R_RVAPI_AlphablendingVDC.... 56

7. Function Reference(CEU)....57
7.1 R_RVAPI_InitializeCEU 57
7.2 R_RVAPI_TerminateCEU 57
7.3 R_RVAPI_PortSettingCEU.... 58
7.4 R_RVAPI_OpenCEU.... 59
7.5 R_RVAPI_CaptureStartCEU 66

7.6 R_RVAPI_CaptureStopCEU 67
7.7 R_RVAPI_InterruptEnableCEU 68

8. Function Reference(MIPI) 69
8.1 R_RVAPI_InitializeMIPI 69
8.2 R_RVAPI_TerminateMIPI 69
8.3 R_RVAPI_OpenMIPI 70
8.4 R_RVAPI_InterruptEnableMIPI 72
8.5 R_RVAPI_SetupMIPI 75
8.6 R_RVAPI_SetBufferMIPI 80
8.7 R_RVAPI_CaptureStartMIPI 80
8.8 R_RVAPI_CaptureStopMIPI 81

9. Function Reference（SPEA） 82
9.1 R_RVAPI_GraphCreateSurfaceSPEA 82
9.2 R_RVAPI_WindowOffsetSPEA 83
9.3 R_RVAPI_SetWindowSPEA 84
9.4 R_RVAPI_WindowUpdateSPEA 85
9.5 R_RVAPI_GraphCreateSurfaceRLE 86
9.6 R_RVAPI_SetWindowRLE 87

10. How to Import the Driver 88
10.1 e^2 studio 88
10.2 For Projects created outside e^2 studio 88

1. Specifications

RVAPI realizes control of display and video input using the drivers for video display controller (VDC), capture engine unit (CEU), MIPI and video input module (VIN) installed in RZ/A2M. RVAPI can also be used as a reference example for each driver control.

Table 1-1 shows the peripheral functions to be used and their uses

Table 1-1 Peripheral Functions Used by RVAPI and Their Uses

Peripheral Function	Use
RZ/A2M-embedded VDC control	Display and video input control Display and image quality adjustment
RZ/A2M embedded CEU control	CMOS camera video input control
RZ/A2M embedded MIPI control	MIPI camera video input control

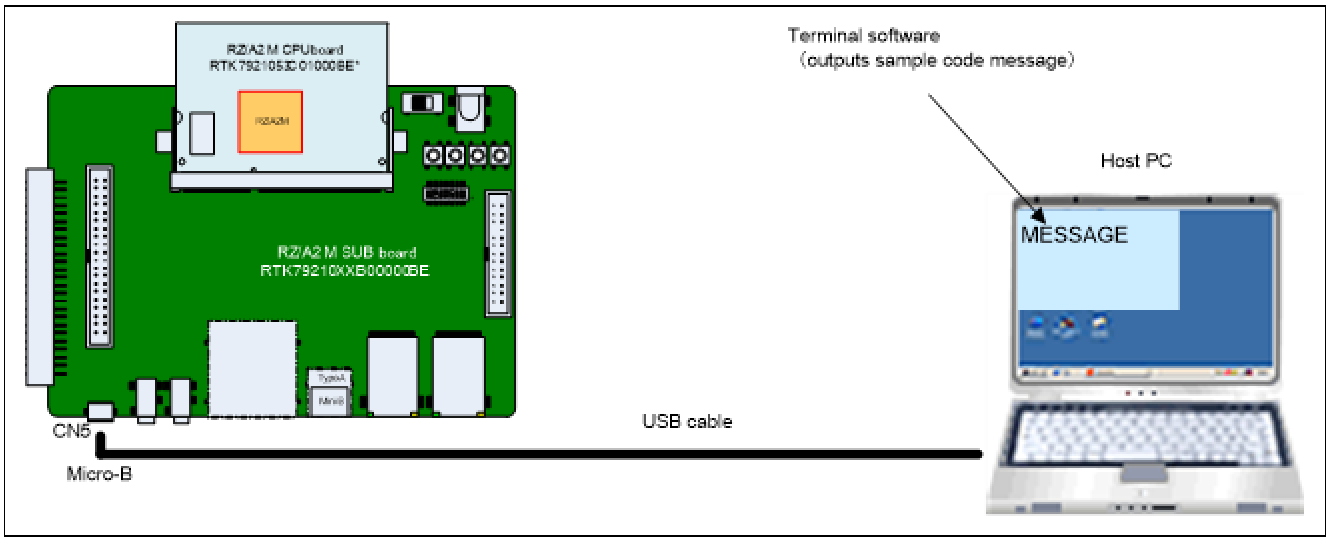

Figure 1.1 Operation check conditions

2. Operating environment

The sample code of this application note supports following environment.

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.) LED1

3. Reference Application Notes

Summaries of the related documents follow.

- RZ/A2M Group Capture Engine Unit Sample Driver(R01AN4474)
  This document describes the functional specifications of CEU driver.
- RZ/A2M Group Video Display Controller and Sprite Engine Sample Driver(R01AN4475)
  This document describes the functional specifications of VDC and SPE driver.
- RZ/A2M Group MIPI Driver(R01AN448)
  This document describes the functional specifications of MIPI driver.

4. Hardware Description

Please refer to the manual of RZ / A2M evaluation board for hardware configuration.

4.1 List of Pins That are Used

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

Table 4-1 Pins to Be Used and Their Functions (Note)

Pin Name	Input/ Output	Description	RZ / A2M evaluation board connection
DV0_CLK	Input	External input clock	NC
DV0_VSYNC	Input	External input Vsync	NC
DV0_HSYNC	Input	External input Hsync	NC
DV0_DATA23 to 0	Input	External input video image data	NC
LCD0_CLK	Output	Panel clock	PJ_6
LCD0_DATA23 to 0	Output	Video image data for panel	PB_5-0, PA_7-0, P8_0, PF_7-0, PH_2
LCD0_TCON6 to 0	Output	Control signal for panel	PC_3(TCON4), PC_4(TCON3), P7_7(TCON0)
LCD0_EXTCLK	Output	Panel clock source	PJ_6
TXCLKOUTM/P	Output	LVDS clock output pins	P4_7, P4_6
TXOUT2M/P	Output	LVDS data output pins	P6_1
TXOUT1M/P	Output	LVDS data output pins	P6_2
TXOUT0M/P	Output	LVDS data output pins	P6_3
VIO_D7 to 0	Input	CEU data bus	PJ_6
VIO_CLK	Input	CEU clock	PB_5-0, PA_7-0, P8_0, PF_7-0, PH_2
VIO_VD	Input	CEU vertical sync	PC_3(TCON4), PC_4(TCON3), P7_7(TCON0)
VIO_HD	Input	CEU horizontal sync	PJ_6
VIO_FLD	Input	Field signal	NC
CSI_DATA0P	Input	Differential positive receiving data input on CSI2 lane 0	Designated pin
CSI_DATA0N	Input	Differential negative receiving data input on CSI2 lane 0	Designated pin
CSI_DATA1P	Input	Differential positive receiving data input on CSI2 lane 1	Designated pin
CSI_DATA1N	Input	Differential negative receiving data input on CSI2 lane 1	Designated pin
CSI_CLKP	Input	Differential positive reception input on CSI2 clock lane	Designated pin
CSI_CLKN	Input	Differential negative reception input on CSI2 clock lane	Designated pin

Note: Refer to the specifications for the individual evaluation board for details.

5. Software Description

5.1 Functions

Table 5-1 gives a list of RVAPI functions. The list also contains the functions that need configuration when providing "display only," "video input only," or "video input and display" functions.

Table 5-1 List of Functions

Display only	Video Input	Video Display	Function Name	Section No.	Outline
VDC video input display function					
Required	Required	Required	R_RVAPI_InitializeVDC	6.1	VDC initialization clock setup
-	-	-	R_RVAPI_TerminateVDC	6.2	VDC termination setup
Required	-	Required	R_RVAPI_DispControlVDC	6.3	Display output setup
Required	-	-	R_RVAPI_GraphCreateSurfaceVDC	6.4	Display area generation
-	-	-	R_RVAPI_GraphChangeSurfaceVDC	6.5	Display buffer address change
-	-	-	R_RVAPI_GraphChangeSurfaceConfigVDC	6.6	Changing the config of data read processing.
-	-	-	R_RVAPI_GraphDestroySurfaceVDC	6.7	Display area disposal
Required	-	Required	R_RVAPI_DispPortSettingVDC	6.8	Display output pin setup
-	Required	Required	R_RVAPI_VideoControlVDC	6.9	Video input setup
-	Required	Required	R_RVAPI_VideoCreateSurfaceVDC	6.10	Video and display area generation
			R_RVAPI_VideoCreateSurface R_RVAPI_VideoCreateSurfaceIMRLS2	6.11	Generate image display area for IMR-LS2
-		-	R_RVAPI_VideoDestroySurfaceVDC	6.12	Video and display area cancellation
-	Required	Required	R_RVAPI_VideoPortSettingVDC	6.13	Video input pin setup
-	-	-	R_RVAPI_InterruptEnableVDC	6.14	VDC interrupt enable setup
-	-	-	R_RVAPI_InterruptDisableVDC	6.15	VDC interrupt disable setup
-	-	-	R_RVAPI_AlphablendingRectVDC	6.16	Rectangle alpha blend
			R_RVAPI_ChromakeyVDC	6.17	Transparency using chroma key
VDC image quality adjustment function					
-	-	-	R_RVAPI_DispCalibrationVDC	6.18	Screen output calibration processing
-	-	-	R_RVAPI_DispGammaVDC	6.19	Gamma calibration setup
-	-	-	R_RVAPI_VideoCalibrationVDC	6.20	Color matrix setup
-	-	-	R_RVAPI_VideoSharpnessLtiVDC	6.21	Image enhancement processing
-	-	-	R_RVAPI_AlphablendingVDC	6.22	1bit alpha blending setup
CEU video input functions (Note 1)					
-	Required	Required	R_RVAPI_InitializeCEU	7.1	CEU initialization setup
-	-	-	R_RVAPI_TerminateCEU	7.2	CEU termination setup
-	Required	Required	R_RVAPI_PortSettingCEU	7.3	Video input pin setup
-	Required	Required	R_RVAPI_OpenCEU	7.4	Image capturing setup
-	Required	Required	R_RVAPI_CaptureStartCEU	7.5	Frame capture start
-	-	-	R_RVAPI_CaptureStopCEU	7.6	Capture stop
-	-	-	R_RVAPI_InterruptEnableCEU	7.7	Interrupt enable setting

Note 1: Setup is required when using CEU for the video inputs.

Display only	Video Input	Video Display	Function Name	Section No.	Outline
<u>MIPI video input functions (Note 2)</u>					
-	Required	Required	R_RVAPI_InitializeMIPI	8.1	MIPI initialization setup
-	-	-	R_RVAPI_TerminateMIPI	8.2	MIPI termination setup
-	Required	Required	R_RVAPI_OpenMIPI	8.3	MIPI capture setup
-	-	-	R_RVAPI_InterruptEnableMIPI	8.4	Interrupt enable setting
-	Required	Required	R_RVAPI_SetupMIPI	8.5	VIN capture setup
	Required	Required	R_RVAPI_SetBufferMIPI	8.6	Capture buffer setting
-	Required	Required	R_RVAPI_CaptureStartMIPI	8.7	Capture start
-	-	-	R_RVAPI_CaptureStopMIPI	8.8	Capture stop
<u>SPEA display setting functions</u>					
Required	-	-	R_RVAPI_GraphCreateSurfaceSPEA	9.1	Display area generation(SPEA)
-	-	-	R_RVAPI_WindowOffsetSPEA	9.2	Setting position of offset for SPEA Window
Required	-	-	R_RVAPI_SetWindowSPEA	9.3	Setting parameter for SPEA Window
Required	-	-	R_RVAPI_WindowUpdateSPEA	9.4	SPEA Window parameter update request
Required	-	-	R_RVAPI_GraphCreateSurfaceRLE	9.5	Display area generation(RLE)
Required	-	-	R_RVAPI_SetWindowRLE	9.6	Setting and updating RLE parameters

Note 2: Setup is required when using MIPI for the video inputs.

6. Function Reference (VDC)

6.1 R_RVAPI_InitializeVDC

R_RVAPI_Initialize

Synopsis	VDC initialization clock setup		
Header	r_rvapi_vdc.h		
Declaration	`vdc_error_t R_RVAPI_InitializeVDC(const vdc_channel_t ch, const clock_config_t * const c_cnf);`		
Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	clock_config_t * c_cnf	: Clock configuration
Return value	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
	VDC_ERR_RESOURCE_LVDS_CLK		: LVDS clock resource error
Remarks			

Declaration as printed:

```
vdc_error_t R_RVAPI_InitializeVDC(
                    const vdc_channel_t ch,
                    const clock_config_t * const c_cnf);
```

(1) Description

VDC can generate the panel clock from various input clocks as the source clocks. This function is used to set up that clock. Since the panel clock is used to control the display device, it is necessary to set up the clock according to the specifications of the display device to be used.

The following driver is used within this function:

- R_VDC_Initialize ()

(2) Parameter details

(a) clock_config_t

The members of the clock_config_t structure are described below.

```
typedef struct
{
    vdc_panel_clksel_t     panel_clk;
    vdc_panel_clk_dcdr_t   panel_clk_div;
    const vdc_lvds_t     * lvds;
} clock_config_t;
```

Type/Member Name	Description
vdc_panel_clksel_t panel_clk	Selects the panel clock. • VDC_PANEL_ICKSEL_IMG_DV Frequency-divided clock for video clock (DV_CLK) • VDC_PANEL_ICKSEL_EXT_0 Frequency-divided clock for peripheral clock 0 (LCD0_EXTCLK) • VDC_PANEL_ICKSEL_PERI Frequency-divided clock for peripheral clock 1 (P1φ) • VDC_PANEL_ICKSEL_LVDS: LVDS PLL clock • VDC_PANEL_ICKSEL_LVDS_DIV7 Clock generated by dividing frequency of LVDS PLL by 7
vdc_panel_clk_dcdr_t panel_clk_div	Specifies the 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 This parameter is invalid when the panel clock select (panel_icksel) is set to LVDS PLL (VDC_PANEL_ICKSEL_LVDS or VDC_PANEL_ICKSEL_LVDS_DIV7).
const vdc_lvds_t * lvds	LVDS-related parameter Specify NULL if this parameter is not required.

(b) **The members of the vdc_lvds_t structure are 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	Selects the frequency divider 1 input • 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	Specifies the frequency divider 1 division ratio NIDIV(Not use). • VDC_LVDS_NDIV_1: NIDIV = 1 • VDC_LVDS_NDIV_2: NIDIV = 2 • VDC_LVDS_NDIV_4: NIDIV = 4
uint16_t lvdspll_tst	Specifies the LVDS PLL internal parameter(Not use).
vdc_lvds_ndiv_t lvds_odiv_set	Specifies the frequency divider 2 division ratio NODIV. • VDC_LVDS_NDIV_1: NODIV = 1 • VDC_LVDS_NDIV_2: NODIV = 2 • VDC_LVDS_NDIV_4: NODIV = 4
vdc_channel_t lvds_vdc_sel	Selects the LVDS VDC channel. • VDC_CHANNEL_0

uint16_t
lvdspll_fd
	Specifies the LVDS PLL feedback ratio NFD.
	NRD = lvdspll_fd + 1
	NFD = lvdspll_fd (22 to 62)

uint16_t lvdspll_rd	Specifies the LVDS PLL input frequency division ratio NRD. NRD = lvdspll_rd + 1 lvdspll_rd (0 to 7)
vdc_lvds_pll_nod_t lvdspll_od	Specifies the LVDS PLL output frequency division ratio NOD(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

(3) Setting up the panel clock

An example of VDC panel clock configuration is shown in Table 6-1. Since the clock generated by the LVDS's PLL can be used for purposes other than LVDS crystal output, the user can generate an arbitrary clock. Examples of VDC panel clock configuration using the LVDS's PLL are shown in Table 6-2.

Table 6-1 Example of Panel Clock Configuration

Member Name	33.0 [MHz]	22.0 [MHz]
panel_icksel	VDC_LVDS_INCLK_SEL_PERI Peripheral clock 1 (P1φ) 66.0 [MHz]	
panel_dcdr	VDC_PANEL_CLKDIV_1_2	VDC_PANEL_CLKDIV_1_3

Note: Peripheral clock 1 (P1φ) is assumed to be 66.0 [MHz].

Table 6-2 Example of Panel Clock Configuration Using LVDS PLL

Member Name	74.25 [MHz]	85.25 [MHz]
panel_icksel	VDC_PANEL_ICKSEL_LVDS	VDC_PANEL_ICKSEL_LVDS
lvds_in_clk_sel	VDC_LVDS_INCLK_SEL_PERI	VDC_LVDS_INCLK_SEL_PERI
lvds_idiv_set	-	-
lvds_odiv_set	VDC_LVDS_NDIV_4	VDC_LVDS_NDIV_4
lvdspll_fd	(27u-1u)	(31u-1u)
lvdspll_rd	(6u-1u)	(6u-1u)
lvdspll_od	-	-

Note: Peripheral clock 1 (P1φ) is assumed to be 66.0 [MHz].

6.2 R_RVAPI_TerminateVDC

R_RVAPI_TerminateVDC			
Synopsis	VDC termination setup		
Header	r_rvapi_vdc.h		

Declaration	`vdc_error_t R_RVAPI_TerminateVDC(`
	`const vdc_channel_t ch);`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
Return value	VDC_OK		: Normal termination
	VDC_ERR_PARAM_CHANNEL		: Channel invalid error
Remarks			

(1) Description

This function performs the VDC driver termination processing. It carries out VDC interrupt and panel clock disable processing.

The following driver is used within this function:

- R_VDC_Terminate ()

6.3 R_RVAPI_DispControlVDC

R_RVAPI_DispControlVDC

Synopsis	Display output setup		
Header	r_rvapi_vdc.h		
Declaration	`vdc_error_t R_RVAPI_DispControlVDC( const vdc_channel_t ch, const vdc_onoff_t res_vs_sel, const qe_config_t * const q_cnf);`		
Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_onoff_t res_vs_sel	: Selects the vertical sync signal to be output (self-running sync signal). • VDC_OFF (Note 1) The vertical sync video input signal is used as the vertical sync signal for the liquid crystal. • VDC_ON Internally generated self-running vertical sync signal
	[IN]	qe_config_t * q_cnf	: Display output configuration
Return value	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
	VDC_ERR_PARAM_UNDEFINED		: Undefined parameter specification error
	VDC_ERR_PARAM_CONDITION		: Unauthorized condition error
	VDC_ERR_RESOURCE_VSYNC		: Vertical sync signal resource error
Remarks			

Note 1: Must not be configured if no video input is present.

(1) Description

This function makes settings with respect to the display output. The user may use, as are, the settings that are generated by the "RZ/A Display Compatible Development Support Tool QE for Display" of the solution tool kit which runs in the integrated development environment e² studio. Visit the Renesas web site for the"RZ/A Display Compatible Development Support Tool QE for Display." A header file generated by the tool contains macro named VDC_xxxx. They are treated as VDC_xxxx in RVAPI header file.

The following drivers are used within this function:

- R_VDC_SyncControl ()
- R_VDC_DisplayOutput ()

(2) Parameter details

(a) qe_config_t

The members of the qe_config_t structure are shown below.

```
typedef struct
{
    uint16_t             vps;
    uint16_t             vpw;
    uint16_t             vs;
    uint16_t             vdp;
    uint16_t             hps;
    uint16_t             hpw;
    uint16_t             hs;
    uint16_t             hdp;
    uint16_t             vtp;
    uint16_t             htp;
    vdc_lcd_tcon_pin_t  tcon_vsync;
    vdc_lcd_tcon_pin_t  tcon_hsync;
    vdc_lcd_tcon_pin_t  tcon_de;
    vdc_sig_pol_t       tcon_vsync_inv;
    vdc_sig_pol_t       tcon_hsync_inv;
    vdc_sig_pol_t       tcon_de_inv;
    uint16_t             tcon_half;
    uint16_t             tcon_ofset;
    vdc_edge_t          lcd_data_out_edge;
    vdc_lcd_outformat_t lcd_outformat;
} qe_config_t;
```

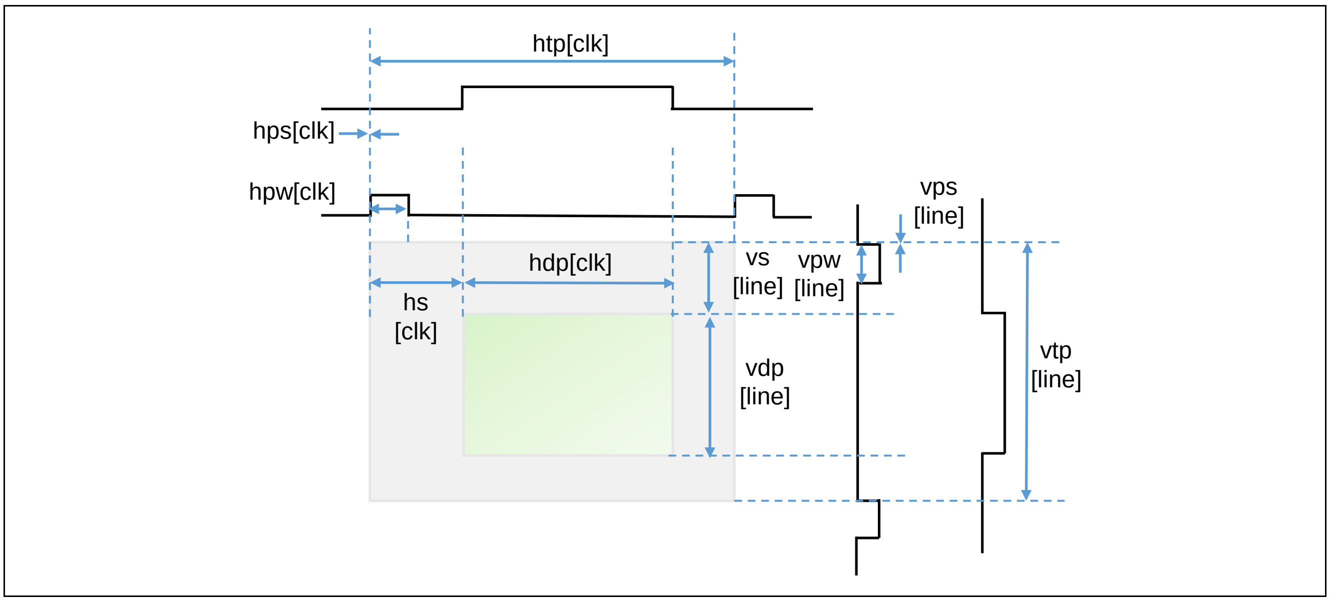

Figure 6-1 Signal Configuration Parameter Diagram

Type/Member Name	Description
uint16_t vps	Vsync pulse start position [in lines]
uint16_t vpw	Vsync pulse width [in lines]
uint16_t vs	Display area vertical start position [in lines]
uint16_t vdp	Vertical display period [in lines]
uint16_t hps	Hsync pulse start position [in clks]
uint16_t hpw	Hsync pulse width [in clks]
uint16_t hs	Display area horizontal start position [in clks]
uint16_t hdp	Horizontal display period [in clks]
uint16_t vtp	Vertical total period [in lines]
uint16_t htp	Horizontal total period [in clks]
vdc_lcd_tcon_pin_t tcon_vsync vdc_lcd_tcon_pin_t tcon_hsync vdc_lcd_tcon_pin_t tcon_de	LCD TCON output pin select • VDC_LCD_TCON_PIN_NON (-1): No output • VDC_LCD_TCON_PIN_0 (0): LCD_TCON0 is output. • VDC_LCD_TCON_PIN_1 (1): LCD_TCON1 is output. • VDC_LCD_TCON_PIN_2 (2): LCD_TCON2 is output. • VDC_LCD_TCON_PIN_3 (3): LCD_TCON3 is output. • VDC_LCD_TCON_PIN_4 (4): LCD_TCON4 is output. • VDC_LCD_TCON_PIN_5 (5): LCD_TCON5 is output. • VDC_LCD_TCON_PIN_6 (6): LCD_TCON6 is output.
vdc_sig_pol_t tcon_vsync_inv vdc_sig_pol_t tcon_hsync_inv vdc_sig_pol_t tcon_de_inv	Horizontal signal operating reference select • VDC_LCD_TCON_REFSEL_HSYNC (0): Horizontal sync signal reference • VDC_LCD_TCON_REFSEL_OFFSET_H (1): Horizontal sync signal reference after offset
uint16_t tcon_half	Specify htp.
uint16_t tcon_ofset	Specify 0.
vdc_edge_t lcd_data_out_edge	LCD_DATA23 to LCD_DATA0 pin output phase control • VDC_EDGE_RISING: Output on rising edge of LCD_CLK pin signal. • VDC_EDGE_FALLING: Output on falling edge of LCD_CLK pin signal.
vdc_lcd_outformat_t lcd_outformat	Output format select • VDC_LCD_OUTFORMAT_RGB888 (0): RGB888 • VDC_LCD_OUTFORMAT_RGB666 (1): RGB666 • VDC_LCD_OUTFORMAT_RGB565 (2): RGB565

6.4 R_RVAPI_GraphCreateSurfaceVDC

R_RVAPI_GraphCreateSurfaceVDC			
Synopsis	Display area generation		
Header	r_rvapi_vdc.h		

Declaration	`vdc_error_t R_RVAPI_GraphCreateSurfaceVDC(`
	`const vdc_channel_t ch,`
	`const gr_surface_disp_config_t * const gr_disp_cnf);`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	gr_surface_disp_config_t * gr_disp_cnf	: Graphics display area settings
Return value	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
Remarks			

(1) Description

This function makes settings for displaying the memory contents allocated in the buffer.

The following drivers are used within this function:

- R_VDC_ReadDataControl ()
- R_VDC_CLUT ()
- R_VDC_StartProcess ()

(2) Parameter details

(a) gr_surface_disp_config_t

The members of the gr_surface_disp_config_t structure are shown below.

```
typedef struct
{
    vdc_layer_id_t          layer_id;
    vdc_pd_disp_rect_t      disp_area;
    void                  * fb_buff;
    uint32_t                 fb_stride;
    vdc_gr_format_t         read_format;
    uint32_t              * clut_table;
    vdc_gr_ycc_swap_t       read_ycc_swap;
    vdc_wr_rd_swa_t         read_swap;
    vdc_gr_disp_sel_t       disp_mode;
} gr_surface_disp_config_t;
```

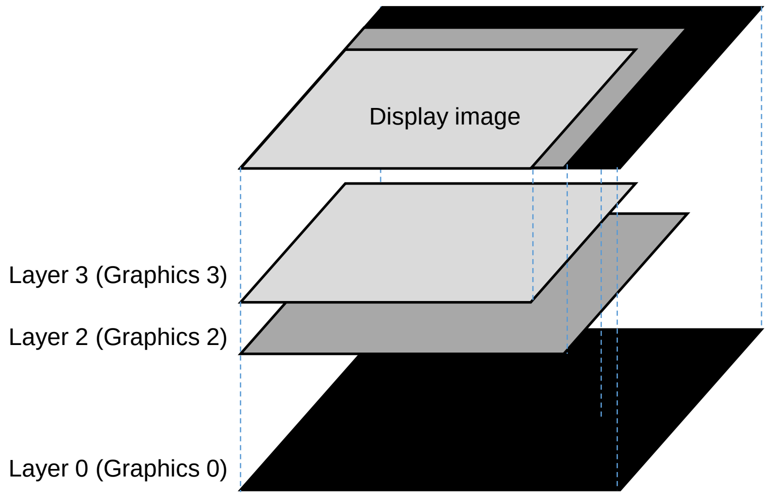

Figure 6-2 Layer Configuration

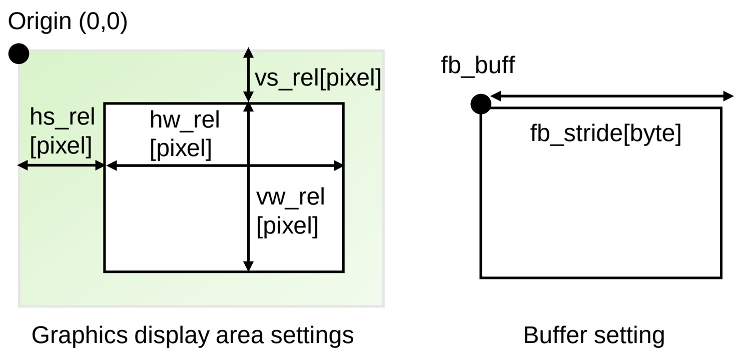

Figure 6-3 Graphics Parameter Diagram

Type/Member Name	Description
vdc_layer_id_t layer_id	Display layer (see Figure 6-2.) • VDC_LAYER_ID_0_RD • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD
vdc_pd_disp_rect_t disp_area	Graphics display area [in pixels] (see Figure 6-3 .) • disp_area.vs_rel / vw_rel: Vertical display start position/vertical display size • disp_area.hs_rel / hw_rel: Horizontal display start position/horizontal display size vs_rel = hs_rel = 0 causes the display to start at the origin.
void * fb_buff	Frame buffer base address (see Figure 6-3.) Do not specify NULL.
uint32_t fb_stride	Frame buffer line offset address [in bytes] (see Figure 6-3.) Specify a multiple of 32 [bytes].
vdc_gr_format_t read_format	Frame buffer read signal format • VDC_GR_FORMAT_RGB565 (0): RGB565 • 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 (Note 1) • VDC_GR_FORMAT_RGBA8888 (11): RGBA8888
uint32_t * clut_table	Color lookup table This parameter is valid only when the value that is set in read_format is VDC_GR_FORMAT_CLUT8/4/1. Specify the address of the area of a size enough to store as many CLUT data blocks (ARGB8888) as the number of colors. If NULL is selected, the default CLUT data is set up. (Default) CLUT8 (256 colors): CLUT Nos. 0-255 Monochrome (black → white) CLUT4 (16 colors): CLUT Nos. 0-15 Black, red, green, cyan, blue, pink, brown, dark green, lightgoldenrod2, dark blue, violet, gray, orange, white, transparent color CLUT1 (2 colors): CLUT Nos. 0-1 black, white
vdc_gr_ycc_swap_t read_ycc_swap	YCbCr422 format mode buffer read data swap control This parameter is valid only when the value specified in read_format is 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 read_swap	Makes 8-bit/16-bit/32-bit swap setting. • VDC_WR_RD_WRSWA_NON (0): No swap 1-2-3-4-5-6-7-8 • VDC_WR_RD_WRSWA_8BIT (1): 8-bit swap 2-1-4-3-6-5-8-7 • VDC_WR_RD_WRSWA_16BIT (2): 16-bit swap 3-4-1-2-7-8-5-6

	• VDC_WR_RD_WRSWA_16_8BIT (3): 16-bit + 8-bit swap 4-3-2-1-8-7-6-5 • VDC_WR_RD_WRSWA_32BIT (4): 32-bit swap 5-6-7-8-1-2-3-4 • VDC_WR_RD_WRSWA_32_8BIT (5): 32-bit + 8-bit swap 6-5-8-7-2-1-4-3 • VDC_WR_RD_WRSWA_32_16BIT (6): 32-bit + 16-bit swap 7-8-5-6-3-4-1-2 • VDC_WR_RD_WRSWA_32_16_8BIT (7): 16-bit + 8-bit swap 8-7-6-5-4-3-2-1
vdc_gr_disp_sel_t disp_mode	Graphics display settings • VDC_DISPSEL_BACK: Background color display • VDC_DISPSEL_LOWER: Lower layer graphics display • VDC_DISPSEL_CURRENT: Current graphics display • VDC_DISPSEL_BLEND : — Blended display of lower layer and current graphics.

Note 1: Layer 0 is configurable.

6.5 R_RVAPI_GraphChangeSurfaceVDC

R_RVAPI_GraphChangeSurfaceVDC			
Synopsis	Display buffer address change		
Header	r_rvapi_vdc.h		

Declaration	`vdc_error_t R_RVAPI_GraphChangeSurfaceVDC(`
	`const vdc_channel_t ch,`
	`const vdc_layer_id_t layer_id,`
	`void* const fb_buff);`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_0_RD • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD
	[IN]	void * framebuff	: Frame buffer base address
Return value	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
Remarks			

(1) Description

This function changes the address of the data read buffer.

The following driver is used within this function:

- R_VDC_ChangeReadProcess ()

6.6 R_RVAPI_GraphChangeSurfaceConfigVDC

R_RVAPI_GraphChangeSurfaceConfigVDC			
Synopsis	Changing the config of data read processing.		
Header	r_rvapi_vdc.h		
Declaration	`vdc_error_t R_RVAPI_GraphChangeSurfaceConfigVDC ( const vdc_channel_t ch, const vdc_layer_id_t layer_id, void* const fb_buff, vdc_period_rect_t * const gr_grc, vdc_width_read_fb_t * const width_read_fb, vdc_gr_disp_sel_t * const gr_disp_sel);`		
Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_0_RD • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD
	[IN]	void * framebuff	: Frame buffer base address
	[IN]	vdc_period_rect_t * gr_grc	: Graphics display area
	[IN]	vdc_width_read_fb_t * width_read_fb	: Size of the frame buffer to be read
	[IN]	vdc_gr_disp_sel_t * r_disp_sel	: Graphics display mode
Return value	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
Remarks			

(1) Description

This function changes the config of data read processing.

The following driver is used within this function:

- R_VDC_ChangeReadProcess ()

(2) Parameter details

(a) 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) vs = 0 causes the display to start at the origin.
uint16_t vw	Vertical signal width (lines)
uint16_t hs	Horizontal signal start position from the reference signal (clock cycles) hs = 0 causes the display to start at the origin.
uint16_t hw	Horizontal signal width (clock cycles)

(b) vdc_width_read_fb

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

(c) **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	**Description**
VDC_DISPSEL_IGNORED	Ignored, no change made
VDC_DISPSEL_BACK	Background color display
VDC_DISPSEL_LOWER	Lower-layer graphics display
VDC_DISPSEL_CURRENT	Current graphics display
VDC_DISPSEL_BLEND	Blended display of lower-layer graphics and current graphics
VDC_DISPSEL_NUM	Number of graphics display modes

6.7 R_RVAPI_GraphDestroySurfaceVDC

R_RVAPI_GraphDestroySurfaceVDC		
Synopsis	Display area disposal	
Header	r_rvapi_vdc.h	
Declaration	`vdc_error_t R_RVAPI_GraphDestroySurfaceVDC( const vdc_channel_t ch, const vdc_layer_id_t layer_id);`	
Arguments	[IN] vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN] vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_0_RD • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD
Return value	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
Remarks		

(1) Description

This function performs stop processing on the specified layer. It stops reading data from the frame buffer and returns the layer's graphics display settings to their initial values.

The following drivers are used within this function:

- R_VDC_StopProcess ()
- R_VDC_ReleaseDataControl ()

6.8 R_RVAPI_DispPortSettingVDC

R_RVAPI_DispPortSettingVDC			
Synopsis	Display output pin setup		
Header	r_rvapi_vdc.h		

Declaration	`void R_RVAPI_DispPortSettingVDC(`
	`        const vdc_channel_t ch,`
	`        void (* const port_func)(uint32_t));`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	void (*port_func) (uint32_t)	: Pointer of the function to set the display control pins.
Return value	None.		
Remarks			

(1) Description

The callback function to be set up with this function must configure the pins that are necessary for display output. This function must be called after making all VDC display settings as shown in Figure 6-4. A control signal of an unexpected period may be output if pin configuration is made before making display settings.

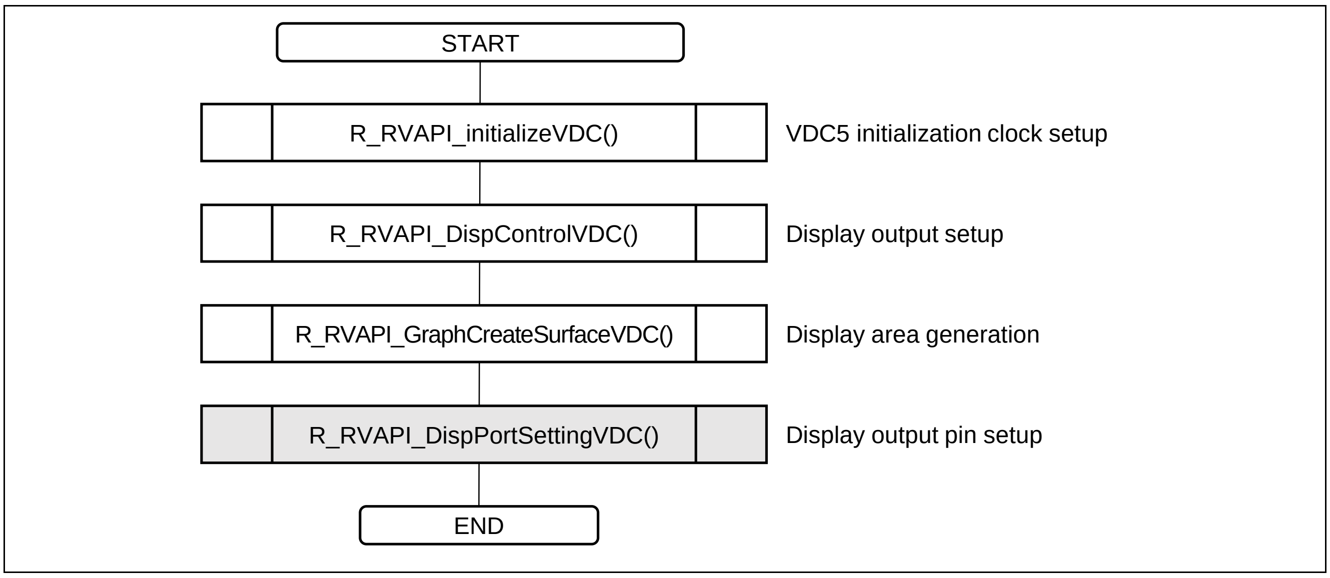

Figure 6-4 Display Output Pin Configuration Timing

6.9 R_RVAPI_VideoControlVDC

R_RVAPI_VideoControlVDC			
Synopsis	Video input setup		
Header	r_rvapi_vdc.h		

Declaration	`vdc_error_t R_RVAPI_VideoControlVDC(`
	`const vdc_channel_t ch,`
	`const digital_in_t * const digital);`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	digital_in_t * digital	: Digital video settings Do not specify NULL.
Return value	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
Remarks			

(1) Description

This function makes video input settings. For the VDC, make settings for the digital video input such as that from the CMOS camera.

The following driver is used within this function:

- R_VDC_VideoInput ()

(2) Parameter details

(a) digital_in_t

The members of the digital_in_t structure are shown below.

```
typedef struct
{
    vdc_extin_format_t     inp_format;
    vdc_edge_t             inp_pxd_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;
} digital_in_t;
```

Type/Member Name	Description
vdc_extin_format_t inp_format	Selects the format of the external input. • VDC_EXTIN_FORMAT_RGB888 (0): RGB888 • VDC_EXTIN_FORMAT_RGB666 (1): RGB666 • VDC_EXTIN_FORMAT_RGB565 (2): RGB565 • VDC_EXTIN_FORMAT_BT656 (3): BT656 • VDC_EXTIN_FORMAT_BT601 (4): BT601 • VDC_EXTIN_FORMAT_YCBCR422 (5): YCbCr422 • VDC_EXTIN_FORMAT_YCBCR444 (6): YCbCr444
vdc_edge_t inp_pxd_edge	Selects the edge on which the external input video signal DV_DATA is to be sampled into the input stage. • VDC_EDGE_RISING : Rising edge • VDC_EDGE_FALLING : Falling edge
vdc_onoff_t inp_endian_on	Sets the bit endian mode of the external inputs. • VDC_OFF • VDC_ON
vdc_onoff_t inp_swap_on	Switches the external input B/R signal. • VDC_OFF • VDC_ON
vdc_sig_pol_t inp_vs_inv	Exercises inversion control of the sync external input signals DV_VSYNC / DV_HSYNC. • VDC_SIG_POL_NOT_INVERTED: Not inverted (positive polarity) • VDC_SIG_POL_INVERTED: Inverted (negative polarity)
vdc_sig_pol_t inp_hs_inv	
vdc_extin_ref_hsync_t inp_h_edge_sel	Selects the reference for the BT656 horizontal sync signal for the external input system. Valid only when inp_format is set to VDC_EXTIN_FORMAT_BT656. • VDC_EXTIN_REF_H_EAV (0): EAV reference • VDC_EXTIN_REF_H_SAV (1): SAV reference
vdc_extin_input_line_t inp_f525_625	Specifies the number of lines for the BT656 input mode for the external input system.

	Valid only when inp_format is set to VDC_EXTIN_FORMAT_BT656. • VDC_EXTIN_LINE_525 (0): 525 lines • VDC_EXTIN_LINE_625 (1): 625 lines
vdc_extin_h_pos_t inp_h_pos	Specifies the data stream start timing with respect to the horizontal sync. The following settings are possible when inp_format is set to VDC_EXTIN_FORMAT_BT656 or VDC_EXTIN_FORMAT_BT601: • VDC_EXTIN_H_POS_CBYCRY (0): Cb/Y/Cr/Y • VDC_EXTIN_H_POS_YCRYCB (1): Y/Cr/Y/Cb • VDC_EXTIN_H_POS_CRYCBY (2): Cr/Y/Cb/Y • VDC_EXTIN_H_POS_YCBYCR (3): Y/Cb/Y/Cr The following settings are possible when inp_format is set to VDC_EXTIN_FORMAT_YCBCR422: • VDC_EXTIN_H_POS_CBYCRY (0): Cb/Y/Cr/Y • VDC_EXTIN_H_POS_YCBYCR (3): Y/Cb/Y/Cr

6.10 R_RVAPI_VideoCreateSurfaceVDC

6.11 R_RVAPI_VideoCreateSurfaceIMRLS2

R_RVAPI_VideoCreateSurfaceVDC
R_RVAPI_VideoCreateSurfaceR_RVAPI_VideoCreateSurfaceIMRLS2

Synopsis Video and display area generation
Generate image display area for IMR-LS2

Header r_rvapi_vdc.h

Declaration

```
vdc_error_t R_RVAPI_VideoCreateSurfaceVDC(
                const vdc_channel_t ch,
                const v_surface_config_t * const v_cnf,
                const v_surface_disp_config_t * const v_disp_cnf);
vdc_error_t R_RVAPI_VideoCreateSurfaceIMRL2(
               const vdc_channel_t ch,
               const v_surface_config_t * const v_cnf,
               const v_surface_disp_config_t * const v_disp_cnf);
```

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	v_surface_config_t * v_cnf l	: Video input area settings Specify NULL when making no video input.
	[IN]	v_surface_disp_config_t * v_g_cnf	: Video input area display settings Specify NULL when making no display.

Return value	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
	VDC_ERR_RESOURCE_LVDS_CLK	: LVDS clock resource error

Remarks

(1) Description

This function sets, as the video input area settings, the video capture timing and buffer write size. It also make settings for the display of the video input. When performing only video capturing, there is no need to make display settings for the video input area. When using IMR-LS 2, please use "R_RVAPI_VideoCreateSurfaceIMRLS2 ()" function. The parameters are the same as the "R_RVAPI_VideoCreateSurfaceVDC ()" function.

The following drivers are used within this function:

- R_VDC_WriteDataControl ()
- R_VDC_ReadDataControl ()
- R_VDC_StartProcess ()

(2) Parameter details

(a) v_surface_config_t

The members of the v_surface_config_t structure are shown below.

```
typedef struct
{
    vdc_layer_id_t      layer_id;
    vdc_period_rect_t   cap_area;
    void               * fb_buff;
    uint32_t             fb_stride;
    uint32_t             fb_offset;
    uint32_t             fb_num;
    vdc_res_md_t        write_format;
    uint16_t             write_fb_vw;
    uint16_t             write_fb_hw;
    vdc_wr_rd_swa_t     write_swap;
    vdc_wr_md_t         write_rot;
    vdc_res_inter_t     res_inter;
} v_surface_config_t;
```

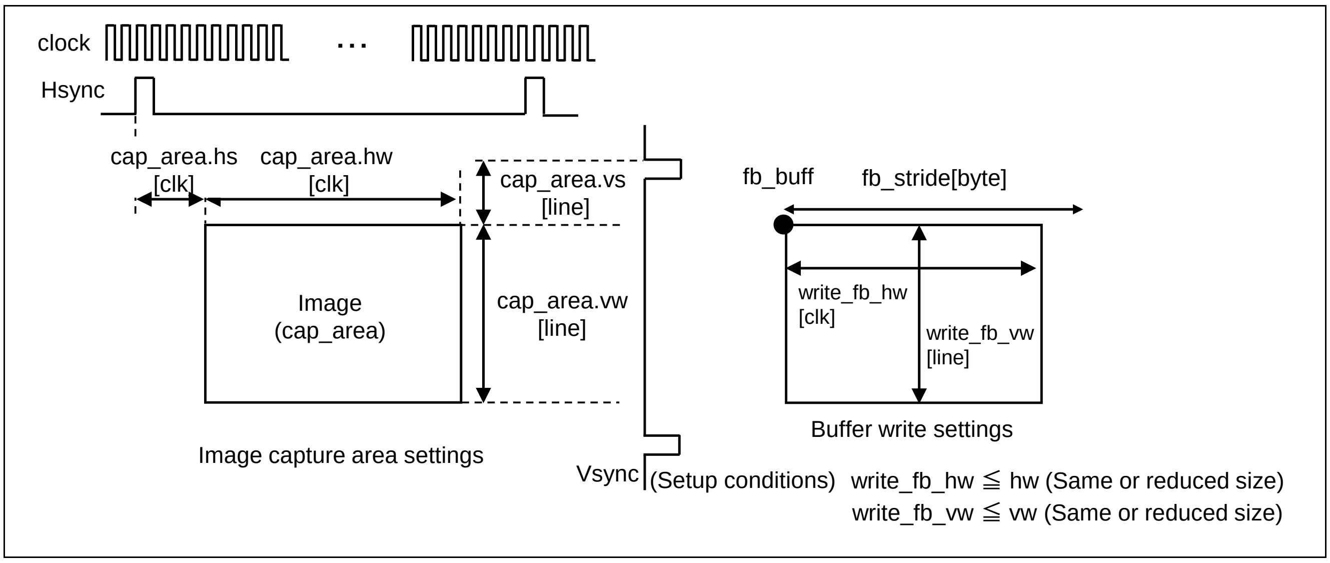

Figure 6-5 Video Input Area Parameter Diagram

Type/Member Name	Description
vdc_layer_id_t layer_id	Layer ID • VDC_LAYER_ID_0_WR
vdc_period_rect_t cap_area	Image capturing range: Horizontal [in clocks] Vertical [in lines] (see Figure 6-5.) cap_area.vs / vw: Vertical capture start position/vertical capture size cap_area.hs / hw: Horizontal capture start position/horizontal capture size
void * fb_buff	Frame buffer base address (see Figure 6-5.) Specify an address that is aligned on a 32 [byte] boundary.
uint32_t fb_stride	Frame buffer line offset address (see Figure 6-5.) Specify a multiple of 32 [lines].
uint32_t fb_offset	Frame buffer frame offset address This parameter is invalid when the number of frames is 1 (fb_num is set to '1'). Specify a multiple of 32.
uint32_t fb_num	Number of write frame buffer frames Specify 1 or 2.
vdc_res_md_t write_format	Frame buffer write video 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
uint16_t write_fb_vw	Buffer write vertical size [in pixels] 0x0000 to 0x07FF Specify a size that is aligned on a 4 [line] boundary and that is not greater than the value of cap_area.res.vw. Data whose size is equal to or smaller than the specified size is written into the buffer.
uint16_t write_fb_hw	Buffer write horizontal size [in clocks] 0x0000 to 0x07FF Specify a size that is aligned on a 4[pixel] boundary and that is not greater than the value of cap_area.hw. Data whose size is equal to or smaller than the specified size is written into the buffer.
vdc_wr_rd_swa_t write_swap	8-bit/16-bit/32-bit swap setting (Note 1) • VDC_WR_RD_WRSWA_NON (0): No swap 1-2-3-4-5-6-7-8 • VDC_WR_RD_WRSWA_8BIT (1): 8-bit swap 2-1-4-3-6-5-8-7 • VDC_WR_RD_WRSWA_16BIT (2): 16-bit swap 3-4-1-2-7-8-5-6 • VDC_WR_RD_WRSWA_16_8BIT (3): 16-bit + 8-bit swap 4-3-2-1-8-7-6-5 • VDC_WR_RD_WRSWA_32BIT (4): 32-bit swap 5-6-7-8-1-2-3-4 • VDC_WR_RD_WRSWA_32_8BIT (5): 32-bit + 8-bit swap 6-5-8-7-2-1-4-3 • VDC_WR_RD_WRSWA_32_16BIT (6): 32-bit + 16-bit swap 7-8-5-6-3-4-1-2 • VDC_WR_RD_WRSWA_32_16_8BIT (7): 32-bit + 16-bit + 8-bit swap 8-7-6-5-4-34-2-1
vdc_wr_rd_swa_t write_swap	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 (write_format) is set to YCbCr422 or RGB565.
vdc_res_inter_t res_inter	Specifies the field operation mode. • VDC_RES_INTER_PROGRESSIVE (0): Progressive • VDC_RES_INTER_INTERLACE (1): Interlace

Note 1: When write_format is set to YCbCr422 or RGB565, be sure to specify a 0 (no swap).

(b) **v_surface_disp_config_t**

The members of the v_surface_disp_config_t structure are shown below.

```
typedef struct
{
    vdc_period_rect_t   disp_area;
    vdc_gr_ycc_swap_t   read_ycc_swap;
    vdc_wr_rd_swa_t     read_swap;
} v_surface_disp_config_t;
```

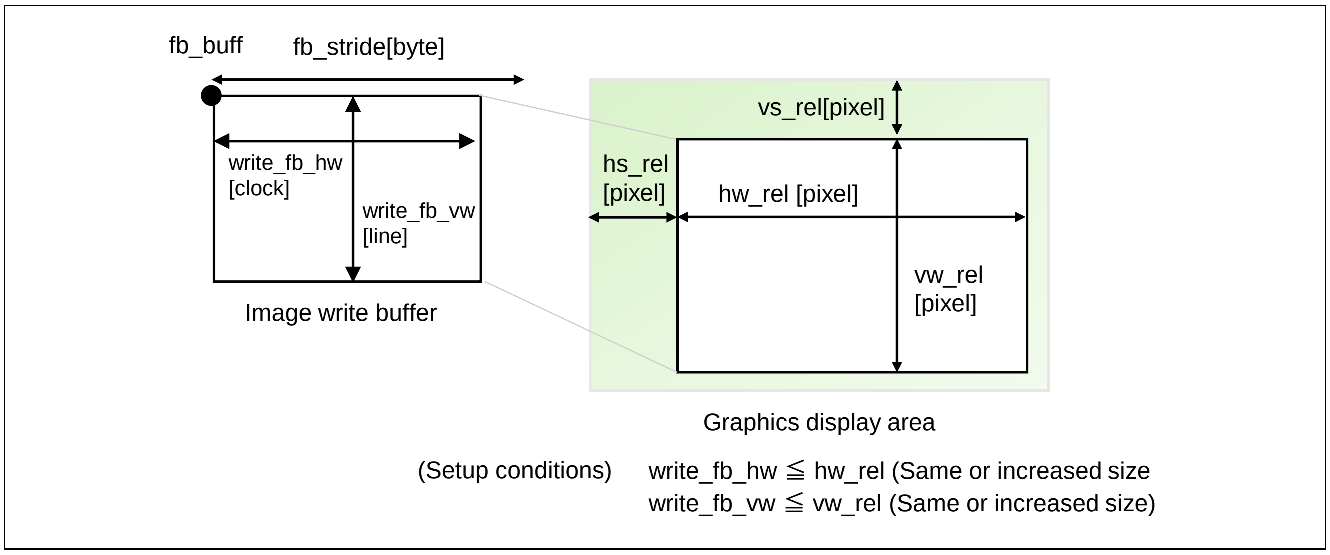

Figure 6-6 Video Input Area Display Parameter Diagram

Type/Member Name	Description
vdc_pd_disp_rect_t disp_area	Graphics display area [in pixels] (see Figure 6-6.) • disp_area.vs_rel / vw_rel: Vertical display start position/vertical display size • disp_area.hs_rel / hw_rel: Horizontal display start position/horizontal display size
vdc_gr_ycc_swap_t read_ycc_swap	YCbCr422 format mode buffer read data swap control This parameter is valid only when the value specified in read_format is 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 read_swap	Makes 8-bit/16-bit/32-bit swap setting. • VDC_WR_RD_WRSWA_NON (0): No swap 1-2-3-4-5-6-7-8 • VDC_WR_RD_WRSWA_8BIT (1): 8-bit swap 2-1-4-3-6-5-8-7 • VDC_WR_RD_WRSWA_16BIT (2): 16-bit swap 3-4-1-2-7-8-5-6 • VDC_WR_RD_WRSWA_16_8BIT (3): 16-bit + 8-bit swap 4-3-2-1-8-7-6-5 • VDC_WR_RD_WRSWA_32BIT (4): 32-bit swap 5-6-7-8-1-2-3-4 • VDC_WR_RD_WRSWA_32_8BIT (5): 32-bit + 8-bit swap 6-5-8-7-2-1-4-3 • VDC_WR_RD_WRSWA_32_16BIT (6): 32-bit + 16-bit swap 7-8-5-6-3-4-1-2 • VDC_WR_RD_WRSWA_32_16_8BIT (7): 32-bit +16-bit + 8-bit swap 8-7-6-5-4-3-2-1

(3) About the configuration of the video capture range

Examples of video capture range configuration are summarized in Table 6-3.

(Example of digital input)

VGA (640 x 480) size progressive input

Writing VGA (640 x 480) size input to buffer in YCbCr422 format with no reduction

The display size is increased from VGA (640 x 480) to SVGA (800 x 600).

Table 6-3 Examples of Video Capture Range Configuration

Structure Name	Member Name	Digital input 24/18/16 bit I/F	Digital input 8-bit I/F
digital_in_t	inp_format	RGB888/666/565 YCbCr422/444	BT6556 BT601
v_surface _config_t	layer_id	VDC_LAYER_ID_0_WR	
	cap_area.vs	Arbitrary	
	cap_area vw	480u	
	cap_area.hs	Arbitrary	
	cap_area.hw	640u x 1u 1[pixel] / 1[clock]	640u x 2u (Note 1) 1[pixel] / 2[clock]
	fb_buff	Internal RAM area	
	fb_stride	640u x 2u (as per YCbCr422)	
	fb_num	2 planes	
	write_format	YCbCr422	
	write_fb_vw	480u	
	write_fb_hw	640u	640u (Note 2)
	res_inter	Progressive	
	fb_offset	Buffer offset	
v_surface _disp_config_t	disp_area.vs_rel	0u	
	disp_area.vw_rel	800u (640u if equal size)	
	disp_area.hs_rel	0u	
	disp_area.hw_rel	600u (480u if equal size)	

Note 1: The capture width clock differs according to the I/F for the external input (1[pixel] / 1[clock] and 1[pixel] / 2[clocks]).

Note 2: Horizontal reduction is required for BT.656/601 because the same image data is captured twice as per VDC specifications. 640u, which is the half of the buffer write setting (write_fb_hw), is set for the capture width clock (cap_area.hw=640u x 2u).

(4) About the configuration of the video capture range to be adopted when using IMR

Make VDC configuration using "R_RVAPI_VideoCreateSurfaceIMRLS2()" when using IMR-LS2. The parameter items are identical to those for "R_RVAPI_VideoCreateSurfaceVDC()." The items that are not referenced are summarized in Table 6-4.

Table 6-4 Parameters Used with IMR

Structure Name	Member Name	IMR-LS2
v_surface _config_t	layer_id	not used
	cap_area.vs	Video input capture position and capture size
	cap_area.hs	
	cap_area vw	
	cap_area.hw	
	fb_buff	Set according to the IMR-LS2 setting.
	fb_stride	
	fb_offset	
	fb_num	
	write_format	not used
	write_fb_vw	Height and width of the video input to IMR-LS2
	write_fb_hw	
	res_inter	Selected according to the video input. Progressive/interlace
v_surface_ disp_config_t	disp_area.vs_rel	Set according to the display size.
	disp_area.vw_rel	
	disp_area.hs_rel	
	disp_area.hw_rel	

6.12 R_RVAPI_VideoDestroySurfaceVDC

R_RVAPI_VideoDestroySurfaceVDC			
Synopsis	Video and display area cancellation		
Header	r_rvapi_vdc.h		
Declaration	`vdc_error_t R_RVAPI_VideoDestroySurfaceVDC ( const vdc_channel_t ch, const vdc_layer_id_t layer_id);`		
Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_0_WR
Return value	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
	VDC_ERR_RESOURCE_LVDS_CLK		: LVDS clock resource error
Remarks			

(1) Description

This function performs stop processing on the specified layer. It stops reading data from the frame buffer and returns the layer's graphics display settings to their initial values.

The following drivers are used within this function:

- R_VDC_StopProcess ()
- R_VDC_ReleaseDataControl ()

6.13 R_RVAPI_VideoPortSettingVDC

R_RVAPI_VideoPortSettingVDC			
Synopsis	Video input pin setup		
Header	r_rvapi_vdc.h		

Declaration	`void R_RVAPI_VideoPortSettingVDC(`
	`        const vdc_channel_t ch,`
	`        void (* const port_func)(uint32_t));`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	void (* const port_func) (uint32_t)	: Pointer of function to set the video input pins.
Return value	None.		
Remarks			

(1) Description

The callback function to be set up with this function must configure the video input pins. This function must have been called by the time the video area is generated as shown in Figure 6-7.

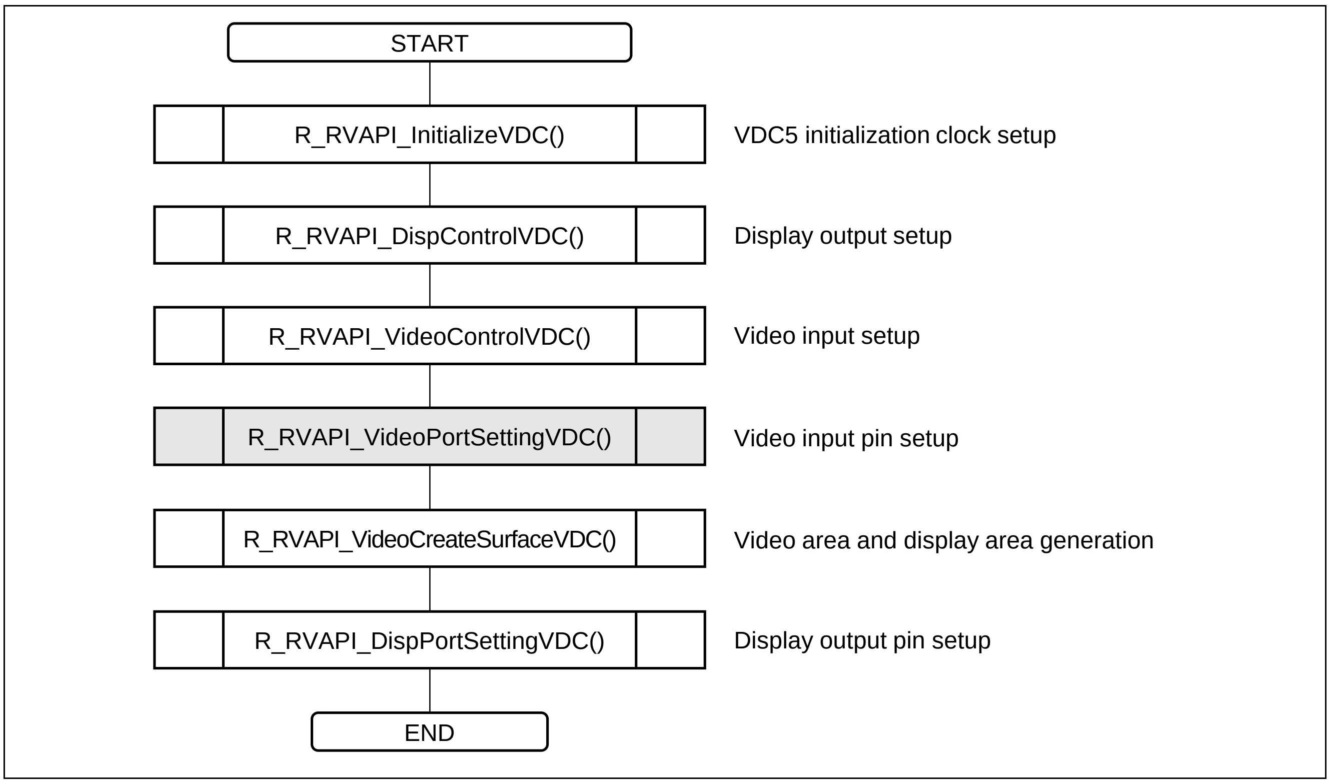
START
R_RVAPI_InitializeVDC()
VDC5 initialization clock setup
R_RVAPI_DispControlVDC()
Display output setup
R_RVAPI_VideoControlVDC()
Video input setup
R_RVAPI_VideoPortSettingVDC()
Video input pin setup
R_RVAPI_VideoCreateSurfaceVDC()
Video area and display area generation
R_RVAPI_DispPortSettingVDC()
Display output pin setup
END

Figure 6-7 Timing of Configuring the Video Input Pins

6.14 R_RVAPI_InterruptEnableVDC

R_RVAPI_InterruptEnableVDC			
Synopsis	VDC interrupt enable setup		
Header	r_rvapi_vdc.h		
Declaration	`vdc_error_t R_RVAPI_InterruptEnableVDC( const vdc_channel_t ch, const vdc_int_type_t flag, const uint16_t line_num, void (* const callback)(vdc_int_type_t int_type, uint32_t buff));`		
Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_int_type_t flag	: VDC interrupt type
	[IN]	uint16_t line_num	: Sets up the line interrupt. Valid only for VDC_INT_TYPE_VLINE
	[IN]	void (*callback) (vdc_int_type_t, void * buff)	: Interrupt callback function pointer
Return value	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
Remarks			

(1) Description

This function enables the interrupts of the VDC interrupt types described in Table 6-5 and registers the specified callback function.

The following driver is used within this function:

- R_VDC_CallbackISR ()

(2) Parameter details

The VDC interrupt types are listed in Table 6-5.

Table 6-5 VDC interrupt type

Enumeration Constant	Value	Description
VDC_INT_TYPE_S0_VI_VSYNC	0	Vertical sync signal input to scaling 0
VDC_INT_TYPE_S0_LO_VSYNC	1	Vertical sync signal output from scaling 0
VDC_INT_TYPE_S0_VSYNCERR	2	Missing vertical sync signal of scaling 0
VDC_INT_TYPE_VLINE	3	Graphics (3) panel output designation line signal
VDC_INT_TYPE_S0_VFIELD	4	End of field signal of the scaling 0 record function
VDC_INT_TYPE_IV1_VBUFERR	5	Scaling 0 frame buffer write overflow signal
VDC_INT_TYPE_IV3_VBUFERR	6	Graphics (0) frame buffer read underflow signal
VDC_INT_TYPE_IV5_VBUFERR	7	Graphics (2) frame buffer read underflow signal
VDC_INT_TYPE_IV6_VBUFERR	8	Graphics (3) frame buffer read underflow signal

6.15 R_RVAPI_InterruptDisableVDC

R_RVAPI_InterruptDisableVDC		
Synopsis	VDC interrupt disable setup	
Header	r_rvapi_vdc.h	
Declaration	`vdc_error_t R_RVAPI_InterruptDisableVDC( const vdc_channel_t ch, const vdc_int_type_t flag);`	
Arguments	[IN] vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN] vdc_int_type_t flag	: VDC interrupt type
Return value	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
Remarks		

(1) Description

This function disables the interrupts of the VDC interrupt types described in Table 6-5.

The following driver is used within this function:

- R_VDC_CallbackISR ()

6.16 R_RVAPI_AlphablendingRectVDC

R_RVAPI_AlphablendingRectVDC

Synopsis	Rectangle alpha blend
Header	r_rvapi_vdc.h

Declaration

```
vdc_error_t R_RVAPI_AlphablendingRectVDC(
                const vdc_channel_t ch,
                const vdc_layer_id_t layer_id,
                const vdc_onoff_t alpha_onoff,
                const vdc_pd_disp_rect_t * const alpha_area,
                const uint8_t alpha_value);
```

Arguments	[IN]	vdc_channel_t ch		: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_layer_id_t	layer_id,	: Layer ID • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD
	[IN]	vdc_onoff_t	alpha_onoff	: Rectangle alpha blend ON/OFF setting • VDC_ON • VDC_OFF
	[IN]	vdc_pd_disp_rect_t *	alpha_area	: Rectangle alpha blend area [in pixels]
	[IN]	uint8_t	alpha_value	: Alpha value (0 to 255) 0: Perfect transparency

Return value	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_IF_CONDITION	: Interface condition error
	VDC_ERR_RESOURCE_LAYER	: Layer resource error
Remarks		

(1) Description

This function turns on and off rectangular area alpha blending, sets up a rectangular area, and sets an alpha value. The following driver is used within this function:

- R_VDC_AlphaBlendingRect ()

6.17 R_RVAPI_ChromakeyVDC

R_RVAPI_ChromakeyVDC

Synopsis	Transparency using chroma key		
Header	r_vapi_vdc.h		

Declaration

```
vdc_error_t R_RVAPI_ChromakeyVDC(
            const vdc_channel_t ch,
            const vdc_layer_id_t layer_id,
            const vdc_onoff_t gr_ck_on,
            const uint32_t ck_color,
            const uint8_t rep_alpha);
```

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_layer_id_t layer_id,	: Layer ID • VDC_LAYER_ID_0_RD • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD
	[IN]	vdc_onoff_t gr_ck_on	: Chroma key ON/OFF setting • VDC_ON • VDC_OFF
	[IN]	uint32_t ck_color	: Color signal subject to chroma keying Specify with the color format that is used for the target layer (LSB justified).
	[IN]	uint8_t rep_alpha	: Alpha value after chroma key replacement (0 to 255)

Return value	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
Remarks		

(1) Description

This function turns on and off chroma keying and sets the color signal to be subjected to chroma keying and a post-replacement alpha value. The following driver is used within this function:

- R_VDC_Chromakey ()

6.18 R_RVAPI_DispCalibrationVDC

R_RVAPI_DispCalibrationVDC

Synopsis	Screen output calibration processing
Header	r_rvapi_vdc.h
Declaration	

```
vdc_error_t R_RVAPI_DispCalibrationVDC(
                const vdc_channel_t ch,
                const vdc_calibr_route_t route,
                const vdc_calibr_bright_t * const bright,
                const vdc_calibr_contrast_t * const contrast,
                const vdc_calibr_dither_t * const panel_dither);
```

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_calibr_route_t route	: Calibration circuit sequence control • VDC_CALIBR_ROUTE_BCG • Brightness ⇒ Contrast ⇒ Gamma calibration • VDC_CALIBR_ROUTE_GBC • Gamma calibration ⇒ Brightness ⇒ Contrast
	[IN]	vdc_calibr_bright_t * bright	: Brightness (DC) adjustment parameter Specify NULL if there is no need to change.
	[IN]	vdc_calibr_contrast_t * contrast	: Contrast (gain) adjustment parameter Specify NULL if there is no need to change.
	[IN]	vdc_calibr_dither_t * panel_dither	: Panel dithering parameter Specify NULL if there is no need to change.

Return value	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
Remarks		

(1) Description

This function makes settings for panel brightness, contrast adjustment, panel dithering, and panel output calibration circuit control. The settings made by this function remain valid until a hardware reset is effected or they are overwritten by other settings made through this function.

The following driver is used within this function:

- R_VDC_DisplayCalibration ()

(2) Parameter details

(a) vdc_calibr_bright_t

The members of the vdc_calibr_bright_t structure are shown 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	G signal brightness (DC) adjustment 0x0000 (-512) to 0x03FF (+511)
uint16_t pbrt_b	512	B signal brightness (DC) adjustment 0x0000 (-512) to 0x03FF (+511)
uint16_t pbrt_r	512	R signal brightness (DC) adjustment 0x0000 (-512) to 0x03FF (+511)

(b) vdc_calibr_contrast_t

The members of the vdc_calibr_contrast_t structure are shown 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	G signal contrast (gain) adjustment 0x0000 (0/128[times]) to 0x00FF (255/128[times])
uint8_t cont_b	128	B signal contrast (gain) adjustment 0x0000 (0/128[times]) to 0x00FF (255/128[times])
uint8_t cont_r	128	R signal contrast (gain) adjustment 0x0000 (0/128[times]) to 0x00FF (255/128[times])

(c) vdc_calibr_dither_t

The members of the vdc_calibr_dither_t structure are shown 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 dithering mode • VDC_PDTH_MD_TRU (0): Truncation • VDC_PDTH_MD_RDOF (1): Rounding • VDC_PDTH_MD_2X2 (2): 2x2 pattern dithering • VDC_PDTH_MD_RAND (3): Random pattern dithering
uint8_t pdth_pa	3	2x2 pattern dithering pattern value 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.
uint8_t pdth_pb	0	2x2 pattern dithering pattern value (B) 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.
uint8_t pdth_pc	2	2x2 pattern dithering pattern value (C) 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.
uint8_t pdth_pd	1	2x2 pattern dithering pattern value (D) 0 to 3 Referenced only when pdth_sel is set to VDC_PDTH_MD_2X2.

6.19 R_RVAPI_DispGammaVDC

R_RVAPI_DispGammaVDC

Synopsis	Gamma calibration setup		
Header	r_rvapi_vdc.h		

Declaration

```
  vdc_error_t R_RVAPI_DispGammaVDC(
                   const vdc_channel_t ch,
                   const vdc_onoff_t gam_on,
                   const uint16_t * const gam_r_gain,
                   const uint8_t * const gam_r_th,
                   const uint16_t * const gam_g_gain,
                   const uint8_t * const gam_g_th,
                   const uint16_t * const gam_b_gain,
                   const uint8_t * const gam_b_th);
```

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_onoff_t gam_on	: Gamma correction ON/OFF setting • VDC_ON • VDC_OFF
	[IN]	uint16_t * gam_r_gain,	: Gain adjustment for the R signal areas 0 to 31 Unsigned (0 to 2047[LSB], 1024[LSB] = 1.0[time])
	[IN]	uint8_t * gam_r_th	: Starting threshold value for the R signal areas 1 to 31 Unsigned (0 to 255[LSB])
	[IN]	uint16_t * gam_g_gain	: Gain adjustment for the G signal areas 0 to 31 Unsigned (0 to 2047[LSB], 1024[LSB] = 1.0[time])
	[IN]	uint8_t * gam_g_th	: Starting threshold value for the G signal areas 1 to 31 Unsigned (0 to 255[LSB])
	[IN]	uint16_t * gam_b_gain	: Gain adjustment for the B signal areas 0 to 31 Unsigned (0 to 2047[LSB], 1024[LSB] = 1.0[time])
	[IN]	uint8_t * gam_b_th	: Starting threshold value for the B signal areas 1 to 31 Unsigned (0 to 255[LSB])
Return value	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
Remarks			

(1) Description

This function turns on and off gamma calibration and sets the gamma calibration values and gamma calibration starting threshold values of the G/B/R signals. For gamma calibration processing, the user can configure gamma calibration ON/OFF control and gamma calibration parameter setup separately. The gamma calibration parameter values, once set, is valid until a hardware reset is effected or they are overwritten by other settings.

The following driver is used within this function:

- R_VDC_GammaCorrection ()

6.20 R_RVAPI_VideoCalibrationVDC

R_RVAPI_VideoCalibrationVDC			
Synopsis	Color matrix setup		
Header	r_rvapi_vdc.h		

Declaration	`vdc_error_t R_RVAPI_VideoCalibrationVDC(`
	`const vdc_channel_t ch,`
	`const vdc_color_matrix_t * const color_matrix);`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_color_matrix_t * color_matrix	: Color matrix setup parameter
Return value	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
Remarks			

(1) Description

This function sets up the specified color matrix. This color matrix is used to adjust the contrast and brightness of the video input.

The following driver is used within this function:

- R_VDC_ImageColorMatrix ()

(2) Parameter details

(a) vdc_color_matrix_t

The members of the vdc_color_matrix_t structure are shown 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	Selects the module to be subjected to color matrix setup. • VDC_COLORMTX_IMGCNT (0): Input controller • VDC_COLORMTX_ADJ_0 (1): Image quality enhancer 0
vdc_colormtx_mode_t mtx_mode	Specifies the color matrix operating mode. • VDC_COLORMTX_GBR_GBR:GBR ⇒ GBR • VDC_COLORMTX_GBR_YCBCR:GBR ⇒ YCbCr (Note 1) • VDC_COLORMTX_YCBCR_GBR:YCbCr ⇒ GBR • VDC_COLORMTX_YCBCR_YCBCR: YCbCr ⇒ YCbCr (Note 1)
uint16_t offset[VDC_COLORMTX_OFFST_NUM]	Y/G, B, and R signal offset (DC) adjustment 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 gain adjustment Signed (2's complement) -1024 to +1023[LSB], 256[LSB] = 1.0 [times]

Note 1: 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.

6.21 R_RVAPI_VideoSharpnessLtiVDC

R_RVAPI_VideoSharpnessLtiVDC			
Synopsis	Image enhancement processing		
Header	r_rvapi_vdc.h		
Declaration	`vdc_error_t R_RVAPI_VideoSharpnessLtiVDC( 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	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_imgimprv_id_t imgimprv_id	: image quality enhancer ID • VDC_IMG_IMPRV_0: Image quality enhancer 0
	[IN]	vdc_onoff_t shp_h_on	: Sharpness ON/OFF setting
	[IN]	vdc_enhance_sharp_t * sharp_param	: Sharpness parameter
	[IN]	vdc_onoff_t lti_h_on	: LTI ON/OFF setting
	[IN]	vdc_enhance_lti_t * lti_param	: LTI parameter
	[IN]	vdc_period_rect_t * enh_area	: Image quality enhancement area parameter
Return value	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
Remarks			

Declaration:

```
vdc_error_t R_RVAPI_VideoSharpnessLtiVDC(
                 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);
```

(1) Description

This function sets up the sharpness ON/OFF setting and sharpness parameters, LTI ON/OFF setting and LTI parameters, and the rectangular area where sharpness and LTI are to be applied.

The following driver is used within this function:

- R_VDC_ImageEnhancement ()

(2) Parameter details

(a) vdc_enhance_sharp_t

The members of the vdc_enhance_sharp_t structure are shown 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)	Selects the LPF to be used for fold removal before H2 edge detection. • VDC_OFF: Without LPF • VDC_ON: With LPF
vdc_sharpness_ctrl_t hrz_sharp [VDC_IMGENH_SHARP_NUM]	-	Sharpness control parameter Horizontal sharpness (H1, H2, H3)

(b) vdc_sharpness_ctrl_t

The members of the vdc_sharpness_ctrl_t structure are shown 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 clip (overshoot side) 0x0000 to 0x00FF
uint8_t shp_clip_u	0	Sharpness correction value clip (undershoot side) 0x0000 to 0x00FF
uint8_t shp_gain_o	0	Specifies the gain for sharpness edge amplitude value (overshoot side) 0x0000 (0 time) to 0x0040 (1 time) to 0x00FF (approx. 4 times)
uint8_t shp_gain_u	0	Specifies the gain for sharpness edge amplitude value (undershoot side) 0x0000 (0 time) to 0x0040 (1 time) to 0x00FF (approx. 4 times)
uint8_t shp_core	0	Specifies the active sharpness area. 0x0000 to 0x007F

(c) vdc_enhance_lti_t

The members of the vdc_enhance_lti_t structure are shown 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)	Selects the LPF to be used for fold removal before H2 edge detection. • VDC_OFF: Without LPF • VDC_ON: With LPF
vdc_lti_mdfil_sel_t lti_h4_median_tap_sel	0	Selects the median filter pixel to be referenced • VDC_LTI_MDFIL_SEL_ADJ2 (0): Reference to 2 adjacent pixels • VDC_LTI_MDFIL_SEL_ADJ1 (1): Reference to 1 adjacent pixel
vdc_lti_ctrl_t lti[VDC_IMGENH_LTI_NUM]	-	LTI control parameter Horizontal LTI (H2, H4)

(d) vdc_lti_ctrl_t

The members of the vdc_lti_ctrl_t structure are shown 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	Specifies the LTI correction threshold for the median filter. 0x0000 to 0x00FF
uint8_t lti_gain	0	Specifies the gain for the LTI edge amplitude value. 0x0000 (0 time) to 0x0040 (1 time) to 0x00FF (approx. 4 times)
uint8_t lti_core	0	LTI coring 0x0000 to 0x00FF

(e) vdc_period_rect_t

The members of the vdc_period_rect_t structure are shown below.

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

Type/Member Name	Initial Value	Description
uint16_t vs	0	Specifies the start position of the effective vertical image area in the enhancer effective area (in lines). Specify 2 lines or more.
uint16_t vw	0	Specifies the width of the effective vertical image area in the enhancer effective area (in lines).
uint16_t hs	0	Specifies the start position of the effective horizontal image area in the enhancer effective area (in clocks). Specify 4 clocks or more.
uint16_t hw	0	Specifies the width of the effective horizontal image area in the enhancer effective area (in clocks).

6.22 R_RVAPI_AlphablendingVDC

R_RVAPI_AlphablendingVDC

Synopsis	1bit alpha blending setup		
Header	r_rvapi_vdc.h		

Declaration	`vdc_error_t R_RVAPI_AlphablendingVDC(`
	`const vdc_channel_t ch,`
	`const vdc_layer_id_t layer_id,`
	`uint8_t alpha_value0,`
	`uint8_t alpha_value1);`		

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_0_RD • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD
	[IN]	uint8_t alpha_value0	: Alpha signal of the ARGB1555/RGBA5551 format Alpha signal when alpha is set to '0' 0 to 255
	[IN]	uint8_t alpha_value1	: Alpha signal of the ARGB1555/RGBA5551 format Alpha signal when alpha is set to '1' 0 to 255
Return value	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
Remarks			

(1) Description

The following driver is used within this function:

- R_VDC_AlphaBlending ()

7. Function Reference(CEU)

7.1 R_RVAPI_InitializeCEU

R_RVAPI_InitializeCEU	
Synopsis	CEU initialization setup
Header	r_rvapi_ceu.h
Declaration	`void R_RVAPI_InitializeCEU(void);`
Arguments	[IN] None :
Return value	None
Remarks	

(1) Description

This function releases the CEU standby mode, enables interrupts, and sets up the interrupt handler.

The following driver is used within this function:

- R_CEU_Initialize ()

7.2 R_RVAPI_TerminateCEU

R_RVAPI_TerminateCEU	
Synopsis	CEU termination setup
Header	r_rvapi_ceu.h
Declaration	`void R_RVAPI_TerminateCEU(void);`
Arguments	[IN] None :
Return value	None
Remarks	

(1) Description

This function enables the CEU standby mode, disables interrupts, and releases the interrupt handler.

The following drivers are used within this function:

- R_CEU_InterruptDisable ()
- R_CEU_Terminate ()

7.3 R_RVAPI_PortSettingCEU

R_RVAPI_PortSettingCEU	
Synopsis	Video input pin setup
Header	r_rvapi_ceu.h

Declaration	`void R_RVAPI_PortSettingCEU(`
	`        void (* const port_func)(uint32_t));`

Arguments	[IN] void (* const port_func) (uint32_t) : Pointer of function to set the video input pins.
Return value	None
Remarks	

(1) Description

The callback function to be set up with this function must configure the pins that are necessary for the CEU to capture video image. This function must have been called by the time the CEU starts image capturing as shown in Figure 7-1.

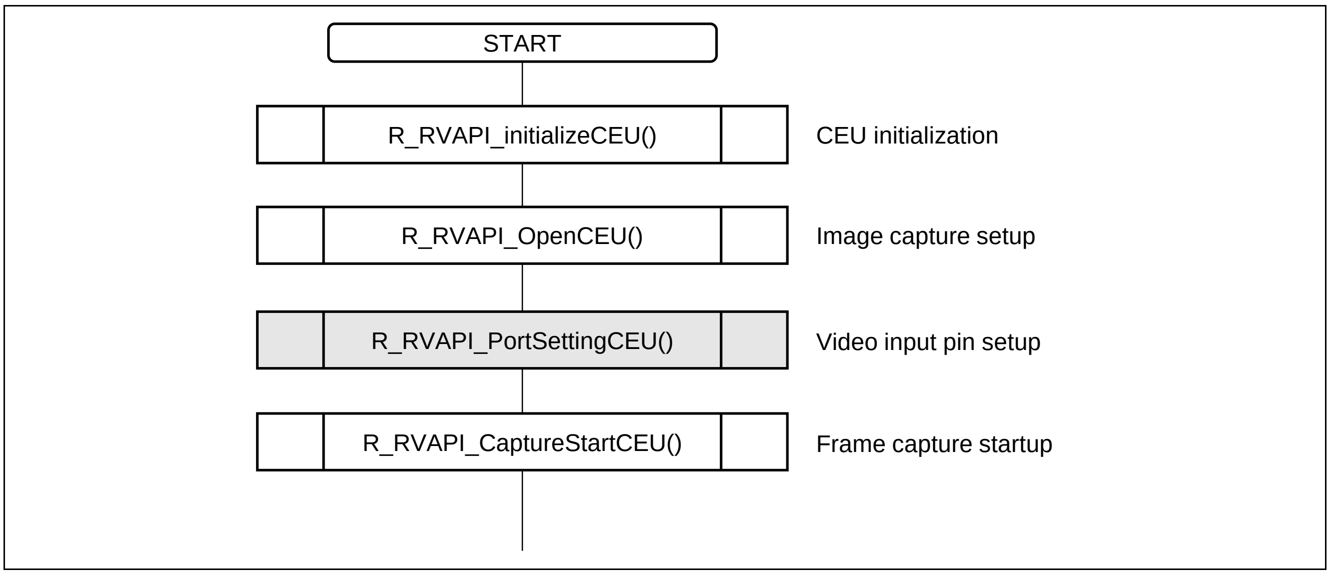

Figure 7-1 Timing When Configuring the CEU's Video Input Pins

7.4 R_RVAPI_OpenCEU

R_RVAPI_OpenCEU			
Synopsis	Image capturing setup		
Header	r_rvapi_ceu.h		

Declaration	`ceu_error_t R_RVAPI_OpenCEU(`
	`const ceu_config_t * const config);`		

Arguments	[IN]	ceu_config_t * config	: Configuration Do not specify NULL.
Return value	CEU_OK		: Normal termination
	CEU_ERR_PARAM		: config or cap is set to NULL, cap and clp values are out of valid range.
Remarks			

(1) Description

This function is used to select the CEU capture mode, set up the capture size, and set up the interface with the external module. There are some parameters that need no configuration depending on the capture mode selected. Table 7-1 lists the parameters that may not be set up.

Table 7-1 Parameters that need not be Set up Depending on the Selected Capture Mode

Capture Mode Selection ceu_jpg_t jpg	Image Capture Mode	Data Synchronous Fetch Mode	Data Enable Fetch Mode
ceu_dtif_t dtif	✓	✓	✓
ceu_sig_pol_t vdpol	✓	✓	Need not be set.
ceu_sig_pol_t hdpol	✓	✓	Need not be set.
ceu_dtary_t dtary	✓	✓ (Note1)	✓ (Note1)
ceu_edge_t dsel	✓	✓	✓
ceu_edge_t fldsel	✓	✓	✓
ceu_edge_t hdsel	✓	✓	✓
ceu_edge_t vdsel	✓	✓	✓
ceu_cap_rect_t * cap	✓	✓	Need not be set.
ceu_clp_t * clp	✓	Need not be set.(Note 2)	Need not be set.
ceu_onoff_t cols/ cows/ cobs	✓	✓	✓

Note 1: CEU_CB0_Y0_CR0_Y1 must be set up by the driver.
Note 2: The driver must set vfclp to vwdth and hfclp to hwdth/2 for the 8-bit interface.
For the 16-bit interface, the driver must set vfclp to vwdth and hfclp to hwdth.

The following drivers are used within this function:

- R_CEU_Open ()

(2) Parameter details

(a) ceu_config_t

The members of the ceu_config_t structure are shown below.

```
typedef struct
{
    ceu_jpg_t           jpg;
    ceu_dtif_t          dtif;
    ceu_sig_pol_t       vdpol;
    ceu_sig_pol_t       hdpol;
    ceu_dtary_t         dtary;
    ceu_edge_t          dsel;
    ceu_edge_t          fldsel;
    ceu_edge_t          hdsel;
    ceu_edge_t          vdsel;
    ceu_cap_rect_t    * cap;
    ceu_clp_t         * clp;
    ceu_onoff_t         cols;
    ceu_onoff_t         cows;
    ceu_onoff_t         cobs;
} ceu_config_t;
```

Type/Member Name	Description
ceu_jpg_t jpg	Capture mode selection • CEU_IMAGE_CAPTURE_MODE Image capture mode • CEU_DATA_SYNC_MODE Data synchronous fetch mode • CEU_DATA_ENABLE_MODE Data enable fetch mode
ceu_dtif_t dtif	Specifies the pins to be used to input the digital image to be captured. • CEU_8BIT_DATA_PINS 8-bit interface • CEU_16BIT_DATA_PINS 16-bit interface
ceu_sig_pol_t vdpol	Specifies the sensing polarity of the vertical sync signal from the external module. • CEU_HIGH_ACTIVE Senses the vertical sync signal from the external module (VD) as a high active signal. • CEU_LOW_ACTIVE Senses the vertical sync signal from the external module (VD) as a low active signal.
ceu_sig_pol_t hdpol	Specifies the sensing polarity of the horizontal sync signal from the external module. • CEU_HIGH_ACTIVE Senses the horizontal sync signal from the external module (HD) as a high active signal. • CEU_LOW_ACTIVE Senses the vertical sync signal from the external module (HD) as a low active signal.

ceu_dtary_t dtary	Specifies the order in which the luminance and color difference components are to be input. Specify CEU_CB0_Y0_CR0_Y1 for the data synchronous and data enable fetch modes. (With the 8-bit interface) • CEU_CB0_Y0_CR0_Y1 The image input data is fetched in the order of Cb0, Y0, Cr0, and Y1. • CEU_CR0_Y0_CB0_Y1 The image input data is fetched in the order of Cr0, Y0, Cb0, and Y1. • CEU_Y0_CB0_Y1_CR0 The image input data is fetched in the order of Y0, Cb0, Y1, and Cr0. • CEU_Y0_CR0_Y1_CB0 The image input data is fetched in the order of Y0, Cr0, Y1, and Cb0. (With the 16-bit interface) • CEU_CB0_Y0_CR0_Y1 The image input data is fetched in the order of {Cb0, Y0} and {Cr0, Y1}. • CEU_CR0_Y0_CB0_Y1 The image input data is fetched in the order of {Cr0, Y0} and {Cb0, Y1}. • CEU_Y0_CB0_Y1_CR0 The image input data is fetched in the order of {Y0, Cb0} and {Y1, Cr0}. • CEU_Y0_CR0_Y1_CB0 The image input data is fetched in the order of {Y0, Cr0} and {Y1, Cb0}.
ceu_edge_t dsel	Sets the edge for fetching the image data from the external module. • CEU_EDGE_RISING Image data is fetched at the rising edge of the camera clock. • CEU_EDGE_FALLING Image data is fetched at the falling edge of the camera clock.
ceu_edge_t fldsel	Sets the edge for capturing the field identification signal from an external module. • CEU_EDGE_RISING The field identification signal is captured at the rising edge of the camera clock. • CEU_EDGE_FALLING The field identification signal is captured at the falling edge of the camera clock.
ceu_edge_t hdsel	Sets the edge for capturing the horizontal sync signal from an external module. • CEU_EDGE_RISING The horizontal sync signal is captured at the rising edge of the camera clock. • CEU_EDGE_FALLING The horizontal sync signal is captured at the falling edge of the camera clock.
ceu_edge_t vdsel	Sets the edge for capturing the vertical sync signal from an external module. • CEU_EDGE_RISING The vertical sync signal is captured at the rising edge of the camera clock. • CEU_EDGE_FALLING The vertical sync signal is captured at the falling edge of the camera clock.
ceu_cap_rect_t * cap	Specifies the capture size. This member needs to be set up when the image capture mode or data synchronous fetch mode is selected. Specify NULL if the member need not be set up.
ceu_clp_t * clp	Filter size clip setting. This member needs to be set up when the image capture mode is selected. Specify NULL if the member need not be set up.

ceu_onoff_t	cols	32-bit swap
ceu_onoff_t	cows	16-bit swap
ceu_onoff_t	cobs	8-bit swap

(b) **ceu_cap_rect_t**

The members of the ceu_cap_rect_t structure are shown below. These members need to be set up when the image capture mode or data synchronous fetch mode is selected.

```
typedef struct
{
    uint32_t    vofst;
    uint32_t    vwdth;
    uint32_t    hofst;
    uint32_t    hwdth;
} ceu_cap_rect_t;
```

Type/Member Name		Description	
uint32_t	vofst	Specifies the capture position with the number of HDs from the vertical sync signal [in 1HD units]. Specify a number 4095 or smaller.	
uint32_t	vwdth	Specifies the capture period in the vertical direction [in 4HD units]. Specify a number not greater than 1920.	
uint32_t	hofst	Specifies the capture position with the number of cycles from the horizontal sync signal [in 1cycle units]. Specify a number 8191 or smaller.	
uint32_t	hwdth	Specifies the capture period in the horizontal direction.	
		(With the 8-bit interface)	
		In image capture mode:	[8 cycle units]: 5,120 cycles or smaller
		In data synchronous fetch mode:	[4 cycle units]: 2,560 or smaller
		(With the 16-bit interface)	
		In image capture mode:	[4 cycle units]: 2,560 cycles or smaller
		In data synchronous fetch mode:	[2 cycle units]: 1,280 or smaller

(c) **ceu_clp_t**

The members of the ceu_clp_t structure are shown below.

These members need to be set up when the image capture mode is selected.

```
typedef struct
{
    uint32_t    vfclp;
    uint32_t    hfclp;
} ceu_clp_t;
```

Type/Member Name		Description
uint32_t	vfclp	Clip value of the vertical direction filter output size [in 4 pixel units]
uint32_t	hfclp	Clip value of the horizontal direction filter output size [in 4 pixel units]

(3) **About the configuration of the capture size**

Given below is an explanation of the capture size configuration (cap) to be made when connecting a CMOS camera which generates YCbCr422 format video output.

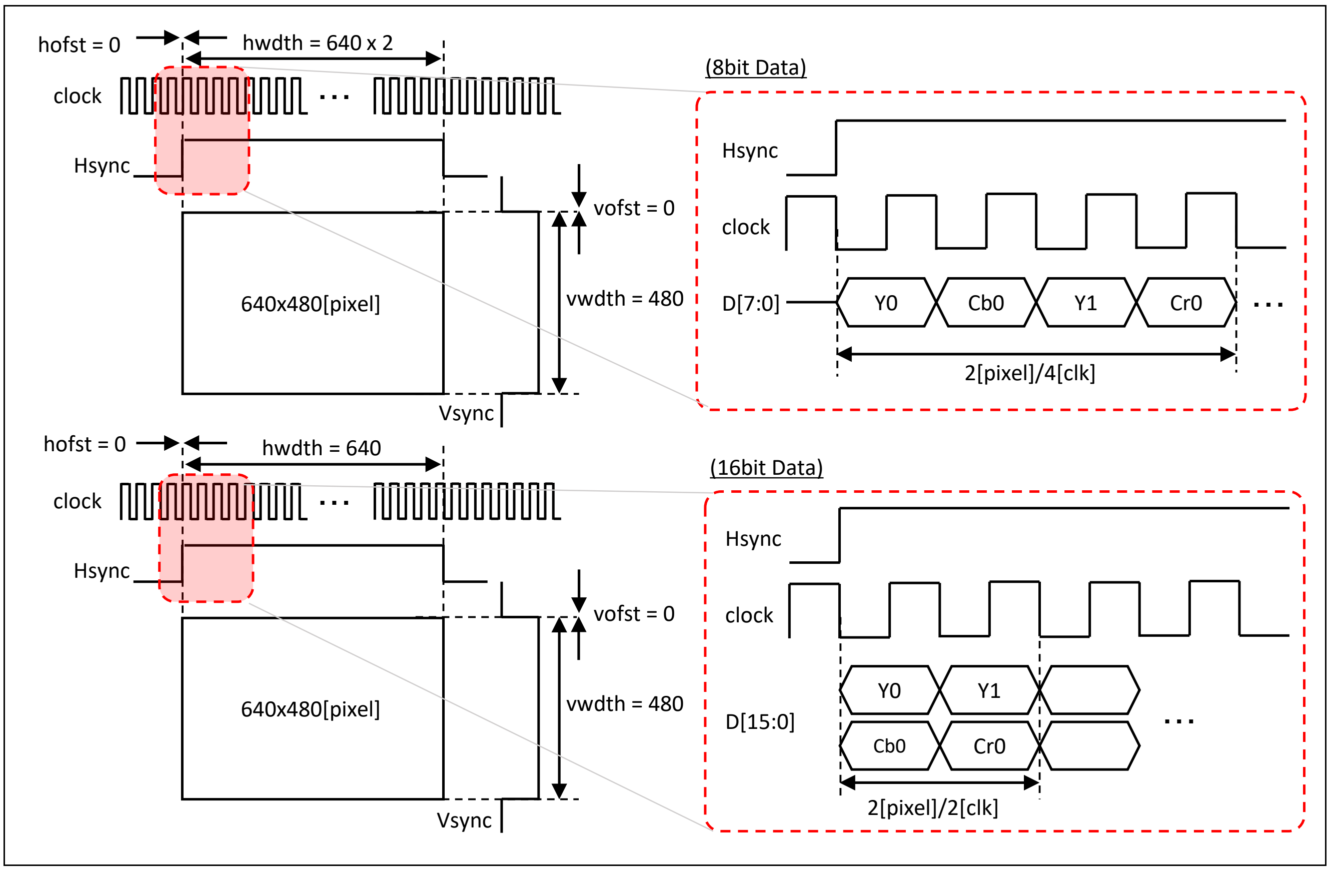

Figure 7-2 Timing of the Signals Output from the Camera

The timing of the camera-output signals is shown in Figure 3-1. This figure shows that since the image data is output from the camera at the same timing when the horizontal sync signals (Hsync)/vertical sync signal (Vsync) rise, hofst/vofst which indicates the image capture position are set to 0.

While the value of vwdth indicating the vertical image capture period is 480 which is the same as the height of the image, the value of hwdth, which indicates the horizontal image capture period, varies depending on the number of clocks that are required to capture 1 pixel.

When an 8-bit interface is attached, since the number of clocks required to capture 2 [pixels] is 4 [clks] (twice), the value of hwdth turns to 640 x 2 [clks].

When a 16-bit interface is attached, since the number of clocks required to capture 2 [pixels] is 2 [clk] (the same value), the value of hwdth turns to 640 [clks].

Figure 7-3 shows a configuration example for a 8-bit interface.

Image capture **mode**

```
ceu_config_t     config;
ceu_cap_rect_t cap;
ceu_clp_t          clp;

config.jpg    =
CEU_IMAGE_CAPTURE_MODE;

cap.hofst     = 0u;
cap.vofst     = 0u;
cap.hwdth   = 640u* 2u;
cap.vwdth   = 480u;
config.cap   = &cap;

clp.hfclp      = 640u;
clp.vfclp      = 480u;
config.clp    = &clp;
```

Data synchronous fetch mode

```
ceu_config_t     config;
ceu_cap_rect_t cap;

config.jpg    =
CEU_DATA_SYNC_MODE;

cap.hofst     = 0u;
cap.vofst     = 0u;
cap.hwdth   = 640u* 2u;
cap.vwdth   = 480u;
config.cap   = &cap;

config.clp    = NULL;
```

Data enable fetch mode

```
ceu_config_t   config;

config.jpg    =
CEU_DATA_ENABLE_MODE;

config.cap   = NULL;

config.clp    = NULL;
```

Figure 7-3 Sample Parameter Settings (8-bit Interface)

Figure 7-4 shows a configuration example for a 16-bit interface.

Image capture mode

```
ceu_config_t      config;
ceu_cap_rect_t cap;
ceu_clp_t          clp;

config.jpg    =
CEU_IMAGE_CAPTURE_MODE;

cap.hofst     = 0u;
cap.vofst     = 0u;
cap.hwdth   = 640u;
cap.vwdth   = 480u;
config.cap   = &cap;

clp.hfclp      = 640u;
clp.vfclp      = 480u;
config.clp    = &clp;
```

Data synchronous fetch mode

```
ceu_config_t     config;
ceu_cap_rect_t cap;

config.jpg    =
CEU_DATA_SYNC_MODE;

cap.hofst     = 0u;
cap.vofst     = 0u;
cap.hwdth   = 640u;
cap.vwdth   = 480u;
config.cap   = &cap;

config.clp    = NULL;
```

Data enable fetch mode

```
ceu_config_t   config;

config.jpg    =
CEU_DATA_ENABLE_MODE;

config.cap   = NULL;

config.clp    = NULL;
```

Figure 7-4 Sample Parameter Settings (16-bit Interface)

7.5 R_RVAPI_CaptureStartCEU

R_RVAPI_CaptureStartCEU

Synopsis Frame capture start

Header r_rvapi_ceu.h

Declaration
```
ceu_error_t R_RVAPI_CaptureStartCEU(
            const void * cayr,
            const void * cacr,
            uint32_t chdw);
```

Arguments

[IN] void * cayr : Data storage area address specification 1
Do not specify NULL.

- In image capture mode
  Address of the area for storing the capture data luminance component data [in 4 byte units]
- Data synchronous fetch mode
  Address of data storage area [in 4 byte units]
- In data enable fetch mode
  Address of data storage area [in 32 byte units]

[IN] void * cacr : Data storage area address specification 2

- This member needs to be set up when the image capture mode is selected.
  Address of the area for storing the capture data color difference component data [in 4 byte units]

[IN] uint32_t chdw : Data buffer stride [bytes]

- In image capture mode
  Capture data buffer stride [in 4 byte units]
- Data synchronous fetch mode
  — (For the 8-bit interface)
  Specify horizontal capture period (hwdth).
  (For the 16-bit interface)
  Specify horizontal capture period (hwdth) x 2.

Return value

CEU_OK	: Normal termination
CEU_ERR_PARAM	: cayr/ cacr set to NULL. (Note 1) : cayr/ cacr values are out of valid range. : chdw value is out of valid range. : The function is called again during capture processing.

Remarks

(1) Description

This function starts capturing one frame. Since this function is of asynchronous type, it is necessary to use function described in "7.6 R_RVAPI_CaptureStopCEU ()" to identify the completion of the 1-frame capturing.

The following driver is used within this function:

- R_CEU_Execute ()

7.6 R_RVAPI_CaptureStopCEU

R_RVAPI_CaptureStopCEU		
Synopsis	Capture stop	
Header	r_rvapi_ceu.h	
Declaration	`ceu_error_t R_RVAPI_CaptureStopCEU(void);`	
Arguments	[in] None	
Return value	CEU_OK	: Normal termination
Remarks		

(1) Description

This function stops the capture.

The following driver is used within this function:

- R_CEU_Stop ()

7.7 R_RVAPI_InterruptEnableCEU

R_RVAPI_CaptureStopCEU			
Synopsis	Capture termination		
Header	r_rvapi_ceu.h		

Declaration	`ceu_error_t R_RVAPI_InterruptEnableCEU(`
	`const ceu_int_type_t int_type,`
	`void (* const callback)(ceu_int_type_t));`		

Arguments	[in]	ceu_int_type_t int_type	:
	[in]	callback void (*callback)(ceu_int_type_t)	:
Return value	CEU_OK		: Normal termination
	CEU_ERR_PARAM		: Callback function is NULL
Remarks			

(1) Description

This function takes the following actions: When using two or more types of interrupts, specify the correct ceu_int_type_t type definitions separated by ORs. The types of interrupts specified in the argument of the callback function will become identifiable.

The following driver is used within this function:

- R_CEU_InterruptEnable ()

8. Function Reference(MIPI)

8.1 R_RVAPI_InitializeMIPI

R_RVAPI_InitializeMIPI		
Synopsis	MIPI initialization setup	
Header	r_rvapi_mipi.h	
Declaration	`void R_RVAPI_InitializeMIPI(void);`	
Arguments	[IN] None	:
Return value	None	
Remarks		

(1) Description

This function release MIPI and VIN standby mode, enables interrupts, and sets up the interrupt handler.

The following driver is used within this function.

- R_MIPI_Initialize ()

8.2 R_RVAPI_TerminateMIPI

R_RVAPI_TerminateMIPI		
Synopsis	MIPI termination setup	
Header	r_rvapi_mipi.h	
Declaration	`void R_RVAPI_TerminateMIPI(void);`	
Arguments	[IN] None	:
Return value	None	
Remarks		

(1) Description

This function enables MIPI and VIN standby mode, disables interrupts, and releases the interrupt handler.

The following drivers are used within this function.

- R_MIPI_InterruptDisable ()
- R_MIPI_Close ()

8.3 R_RVAPI_OpenMIPI

R_RVAPI_OpenMIPI			
Synopsis	MIPI capture setup		
Header	r_rvapi_mipi.h		
Declaration	`e_mipi_error_t R_RVAPI_OpenMIPI(const st_mipi_param_t * const config);`		
Arguments	[IN]	const st_mipi_param_t * const config	: コンフィグレーションデータ NULL は設定しないでください
Return value	MIPI_OK		: Normal termination
	MIPI_PARAM_ERR		: Argument is NULL
Remarks			

(1) Description

This function sets up MIPI capture settings such as capture lane, capture format, PHY settings, and so on.

The following driver is used within this function.

- R_MIPI_Open ()

(2) Parameter details

st_mipi_param_t structure is described as below.

```
typedef struct
{
    uint8_t  mipi_lanenum;          /*!< Mipi Lane Num */
    uint8_t  mipi_vc;               /*!< Mipi Virtual Channel */
    uint8_t  mipi_interlace;        /*!< Interlace or Progressive */
    uint8_t  mipi_laneswap;         /*!< Mipi Lane Swap Setting */
    uint16_t mipi_frametop;         /*!< (for Interlace)Top Field Packet ID */
    uint16_t mipi_outputrate;       /*!< Mipi Data Send Speed(Mbit per sec) */
} st_mipi_param_t;
```

Type / Member Name	Description
uint8_t mipi_lanenum	Number of transfer lane (T.B.D: Fixed 1 at current ver.) (Note) 1: 1 lane operation 2: 2lane parallel operation
uint8_t mipi_vc	Virtual channel 0～3 Enabled virtual channel number
uint8_t mipi_interlace	Input method (T.B.D: Fixed MIPI_PROGRESSIVE at current ver.) (Note) MIPI_PROGRESSIVE: Progressive MIPI_INTERLACE: Interlace
uint8_t mipi_laneswap	Lane swapping (T.B.D: Fixed 0 at current ver.) (Note) 0：Disable lane swapping 1：Enable lane swapping
uint16_t mipi_frametop	Even field number 0x0000～0xFFFF This value is to detect top field of interlace image Set the ID of head line synchronous packet
uint16_t mipi_outputrate	MIPI transfer rate(MHz) (T.B.D: Fixed 80 at current ver.) (Note) 80～1000 Set the MIPI transfer rate

Note: These parameters are not supported at current driver version. Regarding each parameter, please use the fixed value which is indicated in the table.

Even-field number (mipi_frametop) is available when the input method (mipi_interlace) set as MIPI_INTERLACE.

Virtual channel (mipi_vc) means the channel which transfers data from camera.

8.4 R_RVAPI_InterruptEnableMIPI

R_RVAPI_InterruptEnableMIPI		
Synopsis	Interrupt enable setting	
Header	r_rvapi_mipi.h	
Declaration	e_mipi_error_t R_RVAPI_InterruptEnableMIPI(const st_mipi_int_t * const param);	
Arguments	[IN] const st_mipi_int_t * const param	: Interrupt setting Do not specify NULL
Return value	MIPI_OK	: Normal termination
	MIPI_PARAM_ERR	: Argument is NULL
Remarks		

(1) Description

This function takes the following actions. When using two or more types of interrupts, specify the correct e_mipi_interrupt_type_t type definitions separated by ORs. The types of interrupts specified in the argument of the callback function will become identifiable.

- Enable MIPI interrupt which specified by argument.
- Store callback function which specified by argument.

The following driver is used within this function.

- R_MIPI_InterruptEnable ()

(1) Parameter Details

(a) st_mipi_int_t
st_mipi_int_t structure is described as below.

```
typedef struct
{
    e_mipi_interrupt_type_t type;
    void (* p_mipiCallback) (e_mipi_interrupt_type_t interrupt_flag);
    void (* p_vinCallback)  (e_mipi_interrupt_type_t interrupt_flag);
    uint32_t line_num;
} st_mipi_int_t;
```

Type / Member Name	Description
e_mipi_interrupt_type_t type	Interrupt factor of MIPI and VIN Choice the MIPI and VIN interrupt factor needed.
void (* p_mipiCallback) (e_mipi_interrupt_type_t interrupt_flag)	MIPI interrupt callback function Callback function which is called when MIPI interrupt occurs. Do not specify NULL.
void (* p_vinCallback) (e_mipi_interrupt_type_t interrupt_flag)	VIN Interrupt callback function Callback function which is called when VIN interrupt occurs. Do not specify NULL.
uint32_t line_num	Line number for scan line interrupt 0x0000～0x07FF Set the line number in the case of type is VIN_INT_SCANLINE.

(b) e_mipi_interrupt_type_t
e_mipi_interrupt_type_t is an enumeration type for interrupt factor on MIPI and VIN.

```
typedef enum
{
    MIPI_INT_LESS_THAN_WC    = 0x00000001,
    MIPI_INT_AFIFO_OF        = 0x00000002,
    MIPI_INT_VD_START        = 0x00000004,
    MIPI_INT_VD_END          = 0x00000008,
    MIPI_INT_SHP_STB         = 0x00000010,
    MIPI_INT_FSFE            = 0x00000020,
    MIPI_INT_LNP_STB         = 0x00000040,
    MIPI_INT_CRC_ERR         = 0x00000080,
    MIPI_INT_HD_WC_ZERO      = 0x00000100,
    MIPI_INT_FRM_SEQ_ERR1    = 0x00000200,
    MIPI_INT_FRM_SEQ_ERR0    = 0x00000400,
    MIPI_INT_ECC_ERR         = 0x00000800,
    MIPI_INT_ECC_CRCT_ERR    = 0x00001000,
    MIPI_INT_ULPS_START      = 0x00002000,
    MIPI_INT_ULPS_END        = 0x00004000,
    MIPI_INT_ERRSOTHS        = 0x00008000,
    MIPI_INT_ERRSOTSYNCHS    = 0x00010000,
    MIPI_INT_ERRESC          = 0x00020000,
    MIPI_INT_ERRCONTROL      = 0x00040000,
    VIN_INT_FIELD2           = 0x00100000,
    VIN_INT_VSYNC_FALL       = 0x00200000,
    VIN_INT_VSYNC_RISE       = 0x00400000,
    VIN_INT_FIELD            = 0x00800000,
    VIN_INT_SCANLINE         = 0x01000000,
    VIN_INT_FRAME            = 0x02000000,
    VIN_INT_FIFO_OF          = 0x04000000
} e_mipi_interrupt_type_t;
```

Enumeration constant	Value	Description
MIPI_INT_LESS_THAN_WC	00000001H	Length of payload data of a long packet is less than the WC value
MIPI_INT_AFIFO_OF	00000002H	an overflow of the asynchronous FIFO, which stores the HS data sent from the PHY
MIPI_INT_VD_START	00000004H	Start of VD output from the CSI2 (a frame start interrupt)
MIPI_INT_VD_END	00000008H	End of VD output from the CSI2 (a frame end interrupt)
MIPI_INT_SHP_STB	00000010H	Short packet reception interrupt
MIPI_INT_FSFE	00000020H	Frame packet reception interrupt
MIPI_INT_LNP_STB	00000040H	Long packet reception interrupt
MIPI_INT_CRC_ERR	00000080H	CRC error interrupt
MIPI_INT_HD_WC_ZERO	00000100H	WC (word count) zero interrupt
MIPI_INT_FRM_SEQ_ERR1	00000200H	Frame sequence error 1 interrupt (Received an illegal Frame End packet)
MIPI_INT_FRM_SEQ_ERR0	00000400H	Frame sequence error 0 interrupt (Received an illegal Frame Start packet)
MIPI_INT_ECC_ERR	00000800H	ECC error interrupt
MIPI_INT_ECC_CRCT_ERR	00001000H	ECC 1-bit correction interrupt
MIPI_INT_ULPS_START	00002000H	Ultra-low power data transfer start interrupt
MIPI_INT_ULPS_END	00004000H	Ultra-low power data transfer end interrupt
MIPI_INT_ERRSOTHS	00008000H	Synchronized SOT (start of transfer) error interrupt during HS reception.
MIPI_INT_ERRSOTSYNCHS	00010000H	Non-synchronizable SOT (start of transfer) error interrupt during HS reception
MIPI_INT_ERRESC	00020000H	Escape mode entry error interrupt
MIPI_INT_ERRCONTROL	00040000H	PHY control error interrupt
VIN_INT_FIELD2	00100000H	Field interrupt
VIN_INT_VSYNC_FALL	00200000H	VSYNC falling edge detect interrupt
VIN_INT_VSYNC_RISE	00400000H	VSYNC rising edge detect interrupt
VIN_INT_FIELD	00800000H	Field switching interrupt
VIN_INT_SCANLINE	01000000H	Scanline interrupt
VIN_INT_FRAME	02000000H	End of frame interrupt
VIN_INT_FIFO_OF	04000000H	FIFO overflow interrupt

8.5 R_RVAPI_SetupMIPI

R_RVAPI_SetupMIPI			
Synopsis	VIN capture setup		
Header	r_rvapi_mipi.h		
Declaration	`e_mipi_error_t R_RVAPI_SetupMIPI(const st_vin_setup_t * const setup);`		
Arguments	[IN]	const st_vin_setup_t * const setup	: Configuration data Do not specify NULL
Return value	MIPI_OK		: Normal termination
	MIPI_PARAM_ERR		: The vin_setup is illegal or out of range.
Remarks			

(1) Description

This function set up the capture area such as clipping area and so on.

The following driver is used within this function.

- R_MIPI_Setup ()

(1) Parameter details

(a) st_vin_setup_t
st_vin_setup_t structure is described as below.

```
typedef struct
{
    st_vin_preclip_t    vin_preclip;
    uint8_t             vin_inputformat;
    uint8_t             vin_outputformat;
    uint8_t             vin_outputendian;
    uint8_t             vin_interlace;
    uint16_t            vin_stride;
    uint16_t            vin_ycoffset;
    e_vin_input_align_t vin_input_align;
    e_vin_output_swap_t vin_output_swap;
} st_vin_setup_t;
```

Type / Member Name	Description
st_vin_preclip_t vin_preclip	Pre-clip area Pre-clip area setting for capture image. Refer the “st_vin_preclip_t structure” for more detail.
uint8_t vin_inputformat	Input format VIN_INPUT_YCBCR422_8: YUY (=YCbCr422 8bit) VIN_INPUT_YCBCR422_8I: UYVY VIN_INPUT_RAW8: RAW 8bit
uint8_t vin_outputformat	Input format VIN_OUTPUT_YCBCR422_8: YUY (=YCbCr422 8bit) VIN_OUTPUT_Y8_CbCr: YC separation, YCbCr422(Y 8bit, Cb/Cr 8bit) VIN_OUTPUT_Y8: YC separation, Y data(8bit) VIN_OUTPUT_RAW8: RAW 8bit
uint8_t vin_outputendian	Endian type VIN_OUUPUT_EN_LITTLE：Little endian VIN_OUTPUT_EN_BIG：Big endian
uint8_t vin_interlace	Interlace mode VIN_INTERLACE_ODD: Odd-field capture mode VIN_INTERLACE_EVEN: Even-field capture mode VIN_INTERLACE_BOTH: Odd-/even-field capture mode VIN_PROGRESSIVE: Progressive capture mode
uint16_t vin_stride	Stride size of image More than 32 (Multiples of 32) Set the stride size of output image
uint16_t vin_ycoffset	UV data address offset (T.B.D: Fixed 0 at current ver.) (Note) 0～multiple of 128 Set the transfer offset address of UV data when the output format is set as YC separation.
vin_input_align_t vin_input_align	YCbCr422 input data alignment MIPI_Y_UPPER：Y in the upper bits and CbCr in the lower bits. MIPI_CB_UPPER：CbCr in the upper bits and Y in the lower bits.
vin_output_swap_t vin_output_swap	Output Data Byte Swap Mode VIN_SWAP_OFF：Bytes are not swapped in output data. VIN_SWAP_ON：Bytes are swapped in output data.

The endian type (vin_outputendian) is used when output the image data to outside memory.

Stride of image (vin_stride) should be set horizontal pre-clip size (vin_preclip_endx - vin_preclip_startx) or more. The horizontal pre-clip size is set by vin_preclip_endx and vin_preclip_startx of st_vin_preclip_t structure,

So, set the “vin_stride” that satisfy the following condition.

vin_stride >= vin_afterclip_size_x

Also, depending on the output format (vin_outputformat), it is necessary to set the parameters as follows about the stride size of image.

Output format	Setting unit (pixel)
VIN_OUTPUT_YCBCR422_8	64
VIN_OUTPUT_Y8_CbCr8	128
VIN_OUTPUT_Y8	128
VIN_OUTPUT_RAW8	64

The stride size of image is written to VnIS register by MIPI driver. In the case of output format is VIN_OUTPUT_RAW8, MIPI driver writes the value of the stride size of image divided by 2, to VnIS register due to hardware specification.

(b) st_vin_preclip_t

st_vin_preclip_t structure is described as below.

```
typedef struct
{
    uint16_t vin_preclip_starty;    /*!< Pre Area Clip Start Line */
    uint16_t vin_preclip_endy;      /*!< Pre Area Clip End Line */
    uint16_t vin_preclip_startx;    /*!< Pre Area Clip Start Column */
    uint16_t vin_preclip_endx;      /*!< Pre Area Clip End Column */
} st_vin_preclip_t;
```

Type / Member Name	Description
uint16_t vin_preclip_starty	Start line (vertical direction) 0～2046 (In the case of scaling: 0～2044) The value 0 means the first valid line.
uint16_t vin_preclip_endy	End line (vertical direction) 1～2047 (In the case of scaling: 3～2047)
uint16_t vin_preclip_startx	Start pixel (horizontal direction) Even value between 0 to 2042
uint16_t vin_preclip_endx	End pixel (horizontal direction) Odd value between 5 to 2047

The number of lines of vertical direction should be more than 2 lines in pre-clipped area, so, set the “vin_preclip_endy” and “vin_preclip_starty” that satisfy the following conditions.

(vin_preclip_endy - vin_preclip_starty) >= 1

In the case of vertical or horizontal scaling specified, set the “vin_preclip_endy” and “vin_preclip_starty” that satisfy the following conditions.

(vin_preclip_endy - vin_preclip_starty) >=3

The number of pixels of horizontal direction should be even value greater than 6 in pre-clipped area, so, set the “vin_preclip_endx” and “vin_preclip_startx” that satisfy the following conditions. And result of following should be odd-value.

(vin_preclip_endx - vin_preclip_startx) >=5

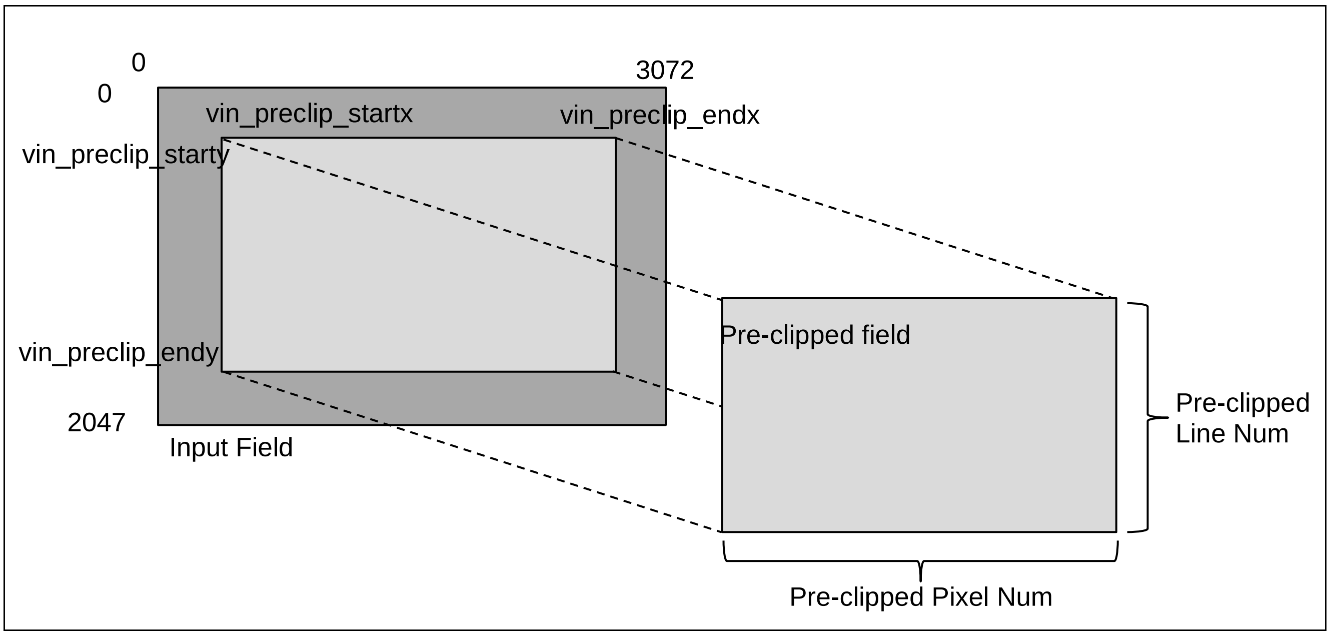

Figure 8-1 Image of pre-clipped area

8.6 R_RVAPI_SetBufferMIPI

R_RVAPI_SetBufferMIPI			
Synopsis	Capture buffer setting		
Header	r_rvapi_mipi.h		
Declaration	`e_mipi_error_t R_RVAPI_SetBufferMIPI(const uint8_t buffer_no, const uint8_t * const buffer);`		
Arguments	[IN]	const uint8_t buffer_no	: MB register number 0: MB1, 1:MB2, 2:MB3
		const uint8_t * const buffer	: Capture buffer address
Return value	MIPI_OK		: Normal termination
	MIPI_PARAM_ERR		: Argument is NULL
	MIPI_STATUS_ERR		: Driver internal status is illegal.
Remarks			

(1) Description

This function sets the buffer address to MB1-MB3 of VIN according the first argument.

The following driver is used within this function.

- R_MIPI_SetBufferAdr ()

8.7 R_RVAPI_CaptureStartMIPI

R_RVAPI_CaptureStartMIPI			
Synopsis	Capture start		
Header	r_rvapi_mipi.h		
Declaration	`e_mipi_error_t R_RVAPI_CaptureStartMIPI(void);`		
Arguments	[IN]	none	
Return value	MIPI_OK		: Normal termination
	MIPI_STATUS_ERR		: Driver internal status is illegal.
Remarks			

(1) Description

This function starts continuous capturing. Since this function is of asynchronous type, it is necessary to use function described in “8.4 R_RVAPI_InterruptEnableMIPI” to identify the completion of the frame capturing.

The following driver is used within this function.

- R_MIPI_CaptureStart()

8.8 R_RVAPI_CaptureStopMIPI

R_RVAPI_CaptureStopMIPI		
Synopsis	Capture stop	
Header	r_rvapi_mipi.h	
Declaration	`e_mipi_error_t R_RVAPI_CaptureStopMIPI(void);`	
Arguments	None	
Return value	MIPI_OK	: Normal termination
	MIPI_STATUS_ERR	: Driver internal status is illegal.
Remarks		

(1) Description

This function stops capturing one frame.

The following driver is used within this function.

- R_MIPI_CaptureStop ()

9. Function Reference（SPEA）

9.1 R_RVAPI_GraphCreateSurfaceSPEA

R_RVAPI_GraphCreateSurfaceSPEA			
Synopsis	Display area generation(SPEA)		
Header	r_rvapi_spea.h		
Declaration	`vdc_error_t R_RVAPI_GraphCreateSurfaceSPEA( const vdc_channel_t ch, const gr_surface_disp_config_t * const gr_disp_cnf);`		
Arguments	[IN]	vdc_channel_t ch	: VDC channel ● VDC_CHANNEL_0
	[IN]	gr_surface_disp_config_t * gr_disp_cnf	: Graphics display area settings
Return value	VDC_OK:		: Normal termination
	VDC_ERR_PARAM_CHANNEL		: Channel invalid error
	VDC_ERR_PARAM_LAYER_ID		: Invalid layer ID error Not setting VDC_LAYER_ID_0_RD
	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-range error
	VDC_ERR_PARAM_CONDITION		: Unauthorized condition error
	VDC_ERR_RESOURCE_LAYER		: Layer resource error
Remarks			

(1) Description

This function makes settings for displaying the memory contents allocated in the buffer.

The following drivers are used within this function.

- R_VDC_ReadDataControl ()
- R_VDC_StartProcess()

9.2 R_RVAPI_WindowOffsetSPEA

R_RVAPI_WindowOffsetSPEA			
Synopsis	Setting offset position for SPEA Window		
Header	r_rvapi_spea.h		
Declaration	`spea_error_t R_RVAPI_WindowOffsetSPEA( const vdc_layer_id_t layer_id, const uint16_t offset_x, const uint16_t offset_y);`		
Arguments	[IN]	vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD • Not setting VDC_LAYER_ID_0_RD
	[IN]	uint16_t offset_x	: Set offset_x 0 or more and 2047 or less in units of 2[pixels].
	[IN]	uint16_t offset_y	: Set offset_y 0 or more and 8191 or less.
Return value	SPEA_OK:		: Normal termination.
	SPEA_ERR_PARAM_LAYER_ID		: Invalid layer ID error
	SPEA_ERR_PARAM		: No permission condition error
Remarks			

(1) Description

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

Sets the arrangement of VDC(layers 2 and 3) display areas for the SPEA virtual frame.

The following driver is used within this function.

- R_SPEA_WindowOffset()

9.3 R_RVAPI_SetWindowSPEA

R_RVAPI_SetWindowSPEA			
Synopsis	Setting parameter for SPEA Window		
Header	r_rvapi_spea.h		
Declaration	`spea_error_t R_RVAPI_SetWindowSPEA( const vdc_layer_id_t layer_id, const spea_window_id_t window_id, const spea_onoff_t sken, const spea_sklym_t * size, const spea_skpsm_t * pos, const void * buffer);`		
Arguments	[IN]	vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD • Not setting VDC_LAYER_ID_0_RD
	[IN]	spea_window_id_t window_id	: SPEA ID • WINDOW_00 ~ WINDOW15:Window ID
	[IN]	spea_onoff_t sken	: SPEA Window ON/OFF • SPEA_ON • SPEA_OFF
	[IN]	spea_sklym_t * size	: Window size • Set offset_x 0 or more and 2047 or less in units of 2[pixels]. • Set offset_y 0 or more and 8191 or less.
	[IN]	spea_skpsm_t * pos	: Window start position Set offset_x in units of 2[pixels]. And, Error occur if the result of adding offset_x set in R_RVAPI_WindowOffsetSPEA to pos.x is not 0 or more but 2047 or less. Error occur if the result of adding offset_y set in R_RVAPI_WindowOffsetSPEA to pos.y is not 0 or more but 8191 or less.
	[IN]	void * buffer	: Window read buffer address Specify 8 byte alignment address.
Return value	SPEA_OK:		: Normal termination
	SPEA_ERR_PARAM_LAYER_ID		: Invalid layer ID error
	SPEA_ERR_PARAM		: No permission condition error
Remarks			

(1) Description

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

Display / Hide SPEA Window

SPEA Window start position, size, setting of read buffer

VDC frame buffer burst transfer mode setting (SPEA_ON:128bytes SPEA_OFF:32bytes transfer)

The following driver is used within this function.

- R_SPEA_SetWindow()

9.4 R_RVAPI_WindowUpdateSPEA

R_RVAPI_WindowUpdateSPEA			
Synopsis	SPEA Window parameter update request		
Header	r_rvapi_spea.h		

Declaration	`spea_error_t R_RVAPI_WindowUpdateSPEA(`
	`const vdc_layer_id_t layer_id);`		

Arguments	[IN]	vdc_layer_id_t layer_id	: Layer ID • VDC_LAYER_ID_2_RD • VDC_LAYER_ID_3_RD • Not setting VDC_LAYER_ID_0_RD
Return value	SPEA_OK:		: Normal termination
	SPEA_ERR_PARAM_LAYER_ID		: Invalid layer ID error
Remarks			

(1) Description

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

SPEA Window parameter update request

The following driver is used within this function.

- R_SPEA_WindowUpdate()

9.5 R_RVAPI_GraphCreateSurfaceRLE

R_RVAPI_GraphCreateSurfaceRLE			
Synopsis	Display area generation(RLE)		
Header	r_rvapi_spea.h		
Declaration	`vdc_error_t R_RVAPI_GraphCreateSurfaceRLE( const vdc_channel_t ch, const gr_surface_disp_config_t * const gr_disp_cnf);`		
Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	gr_surface_disp_config_t * gr_disp_cnf	: Graphics display area setting
Return value	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
	VDC_ERR_RESOURCE_LAYER		: Layer resource error
Remarks			

(1) Description

This function makes settings for displaying the memory contents allocated in the buffer.

The following drivers are used within this function.

- R_VDC_ReadDataControl ()
- R_VDC_StartProcess()

9.6 R_RVAPI_SetWindowRLE

R_RVAPI_SetWindowRLE

Synopsis	Setting and updating RLE parameters		
Header	r_rvapi_spea.h		

Declaration

```
vdc_error_t R_RVAPI_SetWindowRLE(
                  const vdc_channel_t ch,
                  const rle_onoff_t_sken,
                  const rle_cfg_t * rle_cfg,
                  const void * buffer,
                  const uint8_t * g_rle_image,
                  const uint32_t size_of_image);
```

Arguments	[IN]	vdc_channel_t ch	: VDC channel • VDC_CHANNEL_0
	[IN]	rle_onoff_t sken	: RLE ON/OFF • RLE_ON • RLE_OFF
	[IN]	rle_cfg_t * rle_cfg	: Setting NULL（TBD）
	[IN]	void * buffer	: Window read buffer address Specify 8 byte alignment address.
	[IN]	uint8_t * g_rle_image	: Targa format image data
	[IN]	uint32_t size_of_image	: Targa format image file size

Return value	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
	VDC_ERR_RESOURCE_LAYER	: Layer resource error
Remarks		

(1) Description

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

RLE Enable/Disable

Setting RLE parameter

RLE parameter update request

The following drivers are used within this function.

- R_RLE_SetWindow()
- R_RLE_WindowUpdate()
- R_VDC_ChangeReadProcess()

10. How to Import the Driver

10.1 e^2 studio

Please refer to the RZ/A2M Smart Configurator User's Guide: e^2 studio R20AN0583EJ for details on how to import drivers into projects in e2 studio using the Smart Configurator tool.

10.2 For Projects created outside e^2 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.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.14.18	-	First edition issued
1.01	Dec.28.18	75 82~87	Additional parameter of R_RVAPI_SetupMIPI. Additional “9. Function Reference(SPEA)”
1.02	Apr.15.19	59	Addition the following parameters to Table 7-1. ・ceu_edge_t vdsel ・ceu_edge_t hdsel ・ceu_edge_t fldsel ・ceu_edge_t dsel
		59	The following function delete because they are unused. ・R_CEU_InterruptEnable()
		60,61	Addition the following parameters and the explanation of each parameter to the ceu_config_t structure member. ・ceu_edge_t vdsel ・ceu_edge_t hdsel ・ceu_edge_t fldsel ・ceu_edge_t dsel
		68	Modified the function used “R_CEU_Stop()”->” R_CEU_InterruptEnable ()”.
1.10	May.17.19	5	Table 10.1 Peripheral device used(1/2) Remove compiler option "-mthumb-interwork"
		88	Added “10.How to Import the Driver”

Corporate Headquarters

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

Contact information

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

Trademarks