RZ/A2M Group

Capture Engine Unit Sample Driver

Introduction

This document describes the functional specification of Capture Engine Unit (CEU) driver for RZ/A series RZ/A2M group MCU.

Target Device

RZ/A2M

Contents

1. Specifications

This driver uses the RZ/A2M group on-chip CEU to capture the video data and transfer the memory.

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

Table 1-1 Peripheral Functions to Be Used and Their Uses

Classification	Item	Implemented Function	Description	Remarks
Attachable camera	Sample sizes	5M pixels	2,560 pixels ×1,920 line	Horizontal: In 4 pixel units Vertical: In 4 line units (Note 1) Sizes of image that can be input Horizontal 2,560pixels to 128 pixels Vertical 1,920lines to 96 lines
		UXGA	1,600 pixels ×1,200 lines	
		SXGA	1,280 pixels ×1,024 lines	
		XGA	1,024 pixels × 768 lines	
		SVGA	800 pixels ×600 lines	
		VGA	640 pixels ×480 lines	
		Sub-QCIF	128 pixels × 96 lines	
	Input format	YCbCr422 8 bits	Cb0, Y0, Cr0, Y1…	Supports the 1-to-1 clock ratio.
			Cr0, Y0, Cb0, Y1…	
			Y0, Cb0, Y1, Cr0…	
			Y0, Cr0, Y1, Cb0…	
		YCbCr422 16 bits	{Y0, Cb0}, {Y1, Cr0}…	
			{Y0, Cr0}, {Y1, Cb0}…	
		Binary data	Data of the specified size is captured starting at an edge of the sync signal.	Written sequentially.
			Captured using the horizontal sync signal as the enable signal (not configurable on the RZ/A1H and RZ/A1M).	
	Horizontal/vertical sync signal polarity	Optional	Active high Active low	
	Capture start position	Optional	Specifiable in camera input clock units.	Horizontal: In 1 cycle units Vertical: In 1 HD (horizontal sync signal) units
	Number of captured pixels	Optional	Specifiable in units of 4 pixels horizontally and 4 lines vertically.	
	Interlace (Note 2)	Both field capture	Stored as field image.	Capture: In 2 VD (vertical sync signal) units
			Stored as frame image.	
		Single field capture	Top field and bottom field are specifiable.	Capture: In 1 VD units
Memory write	Output format (Note 2)	YCbCr422 YCbCr420	Simple thinning for YCbCr420	
Filter function	Same size, reduced size (Note 2)	Capture image reduction	Arbitrary magnification between 1/16 and 1(The size of the reduced image is VGA or less.)	
	Low-pass filter		High frequency component rejection	Applicable only in horizontal direction.

Note 1: Dependent on the device to be attached, its AC characteristics, the frame rate of the device to be attached, and the rate at which data is to be transferred to RAM.

Note 2: This driver does not support it.

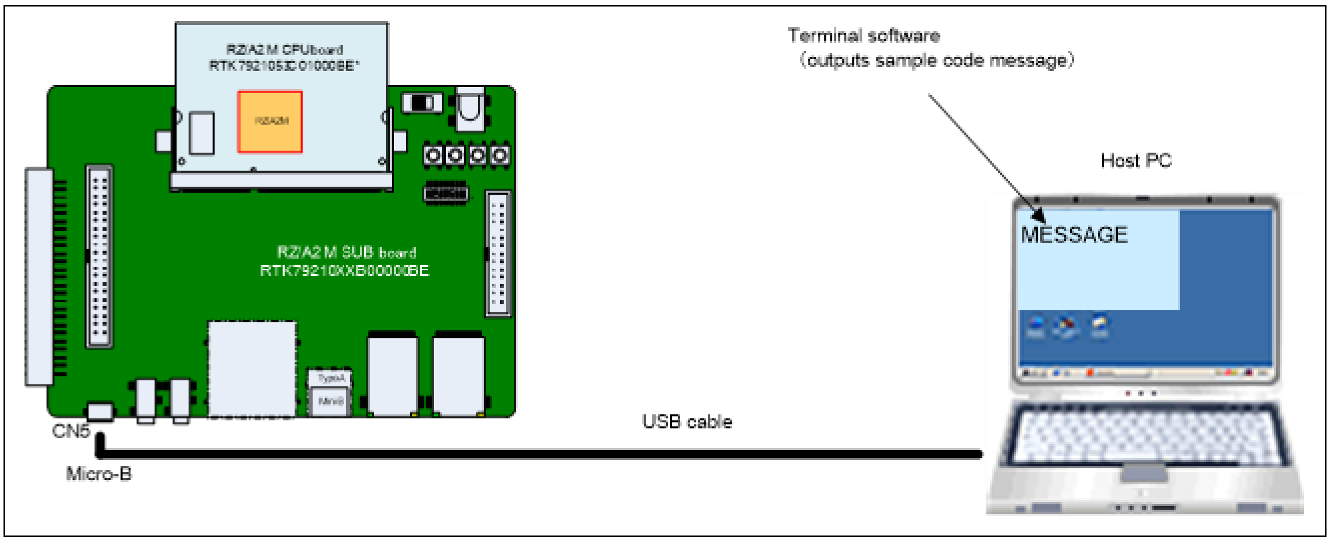

Figure 1.1 Operation check conditions

2. Operation Confirmation Conditions

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

Table 2.1 Peripheral device used(1/2)

Peripheral device	Usage
MCU Used	RZ/A2M
Operating frequency[MHz] (Note)	CPU Clock (Iφ) : 528MHz Image processing clock (Gφ) : 264MHz Internal Bus Clock (Bφ) : 132MHz Peripheral Clock 1 (P1φ) : 66MHz Peripheral Clock 0 (P0φ) : 33MHz QSPI0_SPCLK : 66MHz CKIO : 132MHz
Operating voltage	Power supply voltage (I/O): 3.3 V Power supply voltage (either 1.8V or 3.3V I/O (PVcc SPI)) : 3.3V Power supply voltage (internal): 1.2 V
Integrated development environment	e2 studio V7.4.0
C compiler	"GNU Arm Embedded Tool chain 6-2017-q2-update" compiler options(except directory path) Release: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Os -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -Wstack-usage=100 -fabi-version=0 Hardware Debug: -mcpu=cortex-a9 -march=armv7-a -marm -mlittle-endian -mfloat-abi=hard -mfpu=neon -mno-unaligned-access -Og -ffunction-sections -fdata-sections -Wunused -Wuninitialized -Wall -Wextra -Wmissing-declarations -Wconversion -Wpointer-arith -Wpadded -Wshadow -Wlogical-op -Waggregate-return -Wfloat-equal -Wnull-dereference -Wmaybe-uninitialized -g3 -Wstack-usage=100 -fabi-version=0

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

Table 2.2 Peripheral device used(2/2)

Operation mode	Boot mode 3 (Serial Flash boot 3.3V)
Terminal software communication settings	Communication speed: 115200bps Data length: 8 bits Parity: None Stop bits: 1 bit Flow control: None
Board to be used	RZ/A2M CPU board RTK7921053C00000BE RZ/A2M SUB board RTK79210XXB00000BE
Device (functionality to be used on the board)	Serial flash memory allocated to SPI multi-I/O bus space (channel 0) Manufacturer : Macronix Inc. Model Name : MX25L51245GXD RL78/G1C (This device communications the host PC by convert USB Communication and Serial Communication.)

3. Reference Application Notes

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

Nothing.

4. Hardware Description

4.1 Hardware Configuration

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

4.2 List of Pins to Be Used

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

Table 4-1 Pins to Be Used and Their Functions

Pin name	I/O	Description	RZ / A2M evaluation board connection
VIO7~VIO0	Input	CEU data bus	PE_6-1, PH_1-0
VIO_CLK	Input	CEU clock	P6_1
VIO_VD	Input	CEU vertical sync	P6_2
VIO_HD	Input	CEU horizontal sync	P6_3
VIO_FLD	Input	Field signal	NC

5. Software Description

This section describes each enum type definition in the code. For details on error codes, refer to "5.2 Error Codes".

5.1 Enumeration Definitions

(1) ceu_onoff_t

ceu_onoff_t defines the ON/OFF state of a CEU function. The CEU sample driver uses this constant to configure bit swapping among 32, 16, and 8 bits.

```
typedef enum
{
    CEU_OFF    = 0,
    CEU_ON     = 1
} ceu_onoff_t;
```

Enumerator	Value	Description
CEU_OFF	0	OFF (Disables the function.)
CEU_ON	1	ON (Enables the function.)

(2) ceu_jpg_t

ceu_jpg_t defines the CEU's capture mode.

```
typedef enum
{
    CEU_IMAGE_CAPTURE_MODE = 0,
    CEU_DATA_SYNC_MODE,
    CEU_DATA_ENABLE_MODE
} ceu_jpg_t;
```

Enumerator	Value	Description
CEU_IMAGE_CAPTURE_MODE	0	Image capture mode
CEU_DATA_SYNC_MODE	1	Data synchronous fetch mode
CEU_DATA_ENABLE_MODE	2	Data enable fetch mode

(3) ceu_dtif_t

ceu_dtif_t defines the CEU's input interface.

```
typedef enum
{
    CEU_8BIT_DATA_PINS = 0,
    CEU_16BIT_DATA_PINS
} ceu_dtif_t;
```

Enumerator	Value	Description
CEU_8BIT_DATA_PINS	0	8-bit interface
CEU_16BIT_DATA_PINS	1	16-bit interface

(4) ceu_sig_pol_t

ceu_sig_pol_t defines the sense polarity of the sync signal from the external module.

```
typedef enum
{
    CEU_HIGH_ACTIVE = 0,
    CEU_LOW_ACTIVE
} ceu_sig_pol_t;
```

Enumerator	Value	Description
CEU_HIGH_ACTIVE	0	Senses the sync signal from the external module as a high active signal.
CEU_LOW_ACTIVE	1	Senses the sync signal from the external module as a low active signal.

(5) ceu_dtary_t

ceu_dtary_t defines the order in which the luminance and color difference components are to be input.

```
typedef enum
{
    CEU_CB0_Y0_CR0_Y1 = 0,
    CEU_CR0_Y0_CB0_Y1,
    CEU_Y0_CB0_Y1_CR0,
    CEU_Y0_CR0_Y1_CB0
} ceu_dtary_t;
```

Enumerator	Value	Description
CEU_CB0_Y0_CR0_Y1	0	With the 8-bit interface • The image input data is fetched in the order of Cb0, Y0, Cr0, and Y1. With the 16-bit interface • The image input data is fetched in the order of {Cb0, Y0} and {Cr0, Y1}.
CEU_CR0_Y0_CB0_Y1	1	With the 8-bit interface • The image input data is fetched in the order of Cr0, Y0, Cb0, and Y1. With the 16-bit interface • The image input data is fetched in the order of {Cr0, Y0} and {Cb0, Y1}.
CEU_Y0_CB0_Y1_CR0	2	With the 8-bit interface • The image input data is fetched in the order of Y0, Cb0, Y1, and Cr0. With the 16-bit interface • The image input data is fetched in the order of {Y0, Cb0} and {Y1, Cr0}.
CEU_Y0_CR0_Y1_CB0	3	With the 8-bit interface • The image input data is fetched in the order of Y0, Cr0, Y1, and Cb0. With the 16-bit interface • The image input data is fetched in the order of {Y0, Cr0} and {Y1, Cb0}.

(6) ceu_int_type_t

ceu_int_type_t defines the types of CEU interrupt to be enabled. When using two or more types of interrupts, specify the following definitions separated by ORs in the function "6.6R_CEU_InterruptEnable()."

```
typedef enum
{
    CEU_INT_CPEIE   = (0x00000001u),
    CEU_INT_CFEIE   = (0x00000002u),
    CEU_INT_IGEWIE  = (0x00000010u),
    CEU_INT_HDIE    = (0x00000100u),
    CEU_INT_VDIE    = (0x00000200u),
    CEU_INT_CPBE1IE = (0x00001000u),
    CEU_INT_CPBE2IE = (0x00002000u),
    CEU_INT_CPBE3IE = (0x00004000u),
    CEU_INT_CPBE4IE = (0x00008000u),
    CEU_INT_CDTOFIE = (0x00010000u),
    CEU_INT_IGHSIE  = (0x00020000u),
    CEU_INT_IGVSIE  = (0x00040000u),
    CEU_INT_VBPIE   = (0x00100000u),
    CEU_INT_FWFIE   = (0x00800000u),
    CEU_INT_NHDIE   = (0x01000000u),
    CEU_INT_NVDIE   = (0x02000000u)
} ceu_int_type_t;
```

Error Code	Value	Description (Error type)
CEU_INT_CPEIE	0x00000001u	Enables end of 1-frame capture interrupts.
CEU_INT_CFEIE	0x00000002u	Enables end of 1-field capture interrupts.
CEU_INT_IGEWIE	0x00000010u	Enables register access during capture interrupts (error related).
CEU_INT_HDIE	0x00000100u	Enables HD interrupts.
CEU_INT_VDIE	0x00000200u	Enables VD interrupts.
CEU_INT_CPBE1IE	0x00001000u	Enables CPBE1 interrupts. (Bundle write related)
CEU_INT_CPBE2IE	0x00002000u	Enables CPBE2 interrupts. (Bundle write related)
CEU_INT_CPBE3IE	0x00004000u	Enables CPBE3 interrupts. (Bundle write related)
CEU_INT_CPBE4IE	0x00008000u	Enables CPBE4 interrupts. (Bundle write related)
CEU_INT_CDTOFIE	0x00010000u	Enables CDTOF interrupts. (Error related)
CEU_INT_IGHSIE	0x00020000u	Enables IGHS interrupts. (Error related)
CEU_INT_IGVSIE	0x00040000u	Enables IGVS interrupts. (Error related)
CEU_INT_VBPIE	0x00100000u	Enables VBP interrupts. (Error related)
CEU_INT_FWFIE	0x00800000u	Enables FWF interrupts. (Error related)
CEU_INT_NHDIE	0x01000000u	Enables non-HD interrupts. (Error related) (Note 1)
CEU_INT_NVDIE	0x02000000u	Enables non-VD interrupts. (Error related) (Note 1)

Note 1: This type of interrupts must be disabled in data enable fetch mode.

(7) ceu_edge_t

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

```
typedef enum
{
    CEU_EDGE_RISING    = 0,
    CEU_EDGE_FALLING
} ceu_edge_t;
```

Enumeration constant	Value	Description
CEU_EDGE_RISING	0	Rising edge
CEU_EDGE_FALLING	1	Falling edge

5.2 Error Codes

Table 5-1 shows the error code lists of the CEU driver.

Table 5-1 CEU driver error code list

Error Code	Value	Description (Error type)
CEU_OK	0	Normal termination
CEU_ERR_PARAM	1	Parameter error • Data enable fetch mode was specified on the RZ/A1H or RZ/A1M. • 16-bit interface was specified for a platform other than the RZ/A1H and RZ/A1M. • cap is set to NULL, or cap or clp value is out of valid range. • cayr/ cacr value is set to NULL. (Note 1) • cayr/cacr value is out of valid range. (Note 1) • chdw value is out of valid range.

Note 1: cacr is checked only in image capture mode.

5.3 Restrictions

(1) Reentrancy

The functions of the CEU sample driver are not reentrant. An unexpected driver operation may result if a CEU sample driver function is called asynchronously by two or more tasks or interrupt processing routines.

5.4 Functions

Table 5-2 shows the API function lists of the CEU driver.

Table 5-2 List of RVAPI Functions

Function Name	**Outline**	**Header file**
R_CEU_Initialize	Initialization processing	r_ceu.h
R_CEU_Open	CEU configuration	r_ceu.h
R_CEU_Execute	Frame capture startup processing	r_ceu.h
R_CEU_Stop	Stop the Continuous capture	r_ceu.h
R_CEU_Terminate	CEU termination processing	r_ceu.h
R_CEU_InterruptEnable	Interrupt enable setup	r_ceu.h
R_CEU_InterruptDisable	Interrupt disable setup	r_ceu.h
CEU_Isr	Interrupt handler	r_ceu.h
R_CEU_OnInitialize	Sample for releasing CEU standby state and registering the interrupt handler	r_ceu_user.h
R_CEU_OnFinalize	Sample for setting up CEU standby state and releasing the interrupt handler	r_ceu_user.h

6. Functions Reference

6.1 R_CEU_Initialize

R_CEU_Initialize			
Synopsis	Initialization processing		
Header	r_ceu.h		
Declaration	`void R_CEU_Initialize( void (* const init_func)( uint32_t ), const uint32_t user_num );`		
Arguments	[IN]	void (* init_func)(uint32_t)	: callback function to be registered Specify NULL if not necessary.
	[IN]	uint32_t user_num	: Argument to the callback function Set up according to the application.
Return value	None		
Remarks			

(1) Description

Since the CEU sample driver will perform neither CEU module standby release processing nor interrupt handler registration processing, it is necessary to add those processing using the callback function specified in this function. "6.9 R_CEU_OnInitialize()" is available as a sample function for adding those processing. Add the required processing while referring to that sample.

This function takes the following actions:

- To initialize the internal variables to be used by the sample driver.
- To call the callback function specified in the argument.

6.2 R_CEU_Open

R_CEU_Open	
Synopsis	CEU configuration
Header	r_ceu.h
Declaration	`ceu_error_t R_CEU_Open( 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, or cap or clp value is out of valid range.
Remarks	•

(1) Description

This function is used to select the CEU capture mode and size and to set up the interface with the external module. According to the capture mode, there are some parameters that do not need to be set. Table 6-1lists that parameters that need not be set up.

Table 6-1 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	✓	✓ (Note 1)	✓ (Note 1)
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.

This function takes the following actions:

- Selects the capture mode (jpg).
- Sets up the pins for inputting the digital image to be captured (dtif).
- Specifies the sensing polarity of the sync signal from the external module (vdpol/ hdpol).
- Specifies the order in which the luminance and color difference components are to be input (dtary).
- Sets the edge for fetching the image data from an external module(dsel).
- Sets the edge for capturing the field identification signal from an external module(fldsel).
- Sets the edge for capturing the horizontal sync signal from an external module(hdsel).
- Sets the edge for capturing the vertical sync signal from an external module(vdsel).
- Capture size setting (cap)
- Filter size clip setting (clp)
- 32/16/8-bit swap settings (cols/ cows/ cobs)

(2) Parameter details

(a) ceu_config_t

ceu_config_t structure is described below.

```
typedef struct
{
    ceu_cap_rect_t    * cap;
    ceu_clp_t         * clp;
    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_onoff_t         cols;
    ceu_onoff_t         cows;
    ceu_onoff_t         cobs;
} ceu_config_t;
```

Type/Member Name	Description
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_jpg_t jpg	Selects the capture size. The RZ/A1H or RZ/A1M do not allow the data enable fetch mode to be selected. • 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. The user can also select a 16-bit interface for the RZ/A1H and RZ/A1M. • 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 horizontal 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_Y1for 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_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. This member needs 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 (in 8 cycle units) : 5,120 cycles maximum • In data synchronous fetch mode (in 4 cycle units) : 2,560 cycles maximum (With the 16-bit interface) • In image capture mode (in 4 cycle units) : 2,560 cycles maximum • In data synchronous fetch mode (in 2 cycle units): 1,280 cycles maximum

(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.

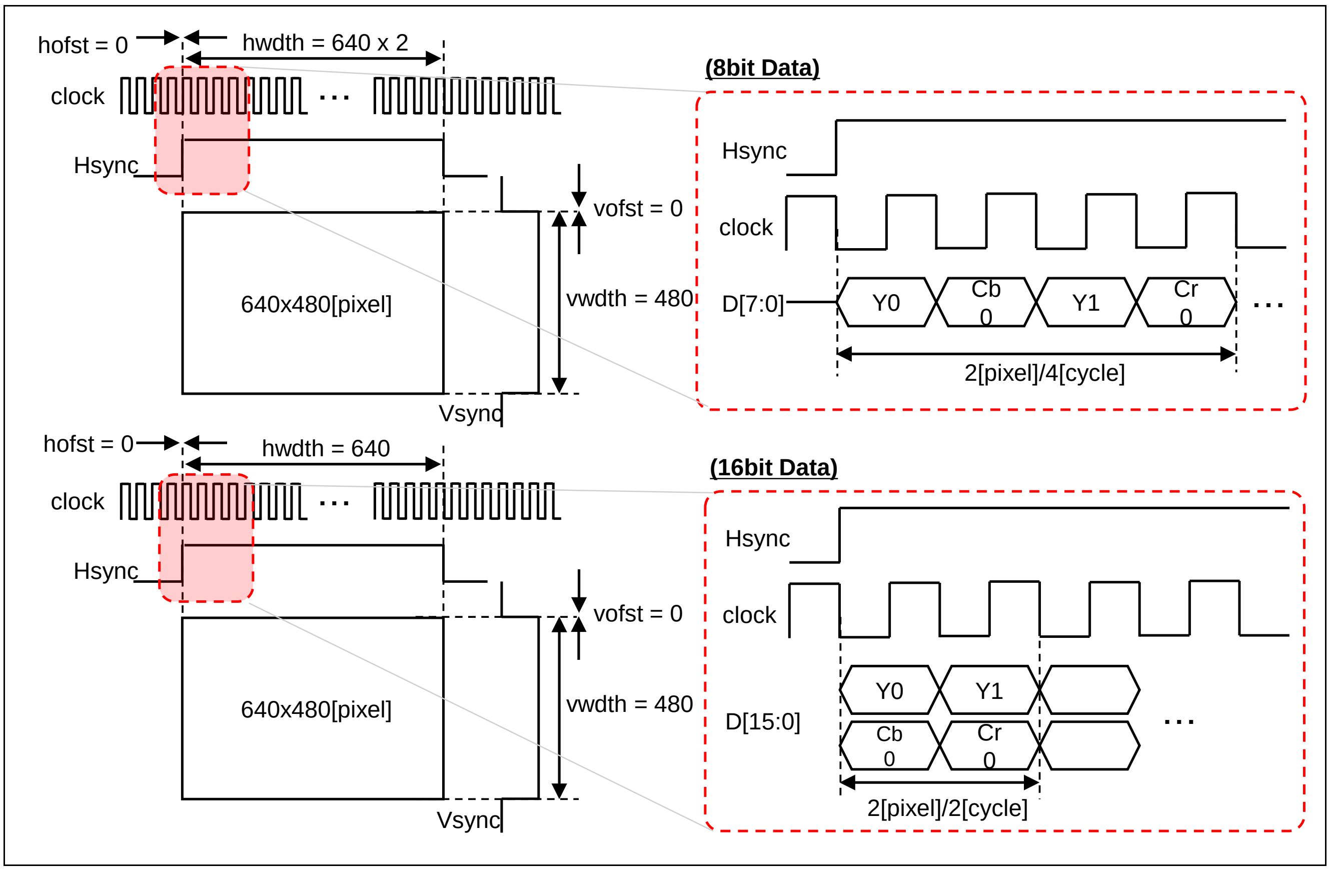

Figure 6-1 Timing of the Signals Output from the Camera

The timing of the camera-output signals is shown in Figure 6-1. 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 [cycles] (twice), the value of hwdth turns to 640 x 2 [cycles].

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

Figure 6-2 shows a configuration example for an 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 6-2 Sample R_CEU_Open Function's Parameter Settings (8-bit Interface)

Figure 6-3 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 6-3 Sample R_CEU_Open Function's Parameter Settings (16-bit Interface)

6.3 R_CEU_Execute

R_CEU_Execute			
Synopsis	Frame capture startup processing		
Header	r_ceu.h		
Declaration	`ceu_error_t R_CEU_Execute( const void * cayr, const void * cacr, uint32_t chdw ceu_onoff_t auto_capture);`		
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) • In 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 (in bytes) • In image capture mode Capture data buffer stride (in 4 byte units) • In 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.
	[IN]	ceu_onoff_t auto_capture	: Continuous capture
Return value	CEU_OK		: Normal termination
	CEU_ERR_PARAM		: cayr/ cacr set to NULL. (Note 1) : cayr/ cacr values are out of valid range. (Note 1) : chdw value is out of valid range.
Remarks	•		

[Note 1] cacr is checked only in image capture mode.

(1) Description

This function starts the image capture processing according to the settings made with the function described in Section 6.2, R_CEU_Open(). After starting image capturing, the end of 1-frame capture interrupt must be used to check for the completion of image capturing.

This function takes the following actions:

- To specify the address of the area for storing data.
- To specify the data buffer stride.

6.4 R_CEU_Stop

R_CEU_Stop		
Synopsis	Stop the Continuous capture	
Header	r_ceu.h	
Declaration	`ceu_error_t R_CEU_Stop(void);`	
Arguments	[IN] void	
Return value	CEU_OK	: Normal termination
Remarks		

(1) Description

In this function, clear the CE bit of the capture start register. Use this function to stop capture during continuous capture operation.

This function takes the following actions:

- Clear the CE bit of the capture start register

6.5 R_CEU_Terminate

R_CEU_Terminate			
Synopsis	CEU termination processing		
Header	r_ceu.h		
Declaration	`ceu_error_t R_CEU_Terminate( void (* const quit_func)( uint32_t ), const uint32_t user_num );`		
Arguments	[IN]	void (* quit_func)(uint32_t)	: Callback function to be registered Specify NULL if not necessary.
	[IN]	uint32_t user_num	: Argument to the callback function Set up according to the application.
Return value	CEU_OK		: Normal termination
Remarks			

(1) Description

This function performs a CEU software reset. Since the CEU sample driver performs neither CEU module standby configuration processing nor interrupt handler release processing, it is necessary to add those processing using the callback function specified in this function.

"6.10 R_CEU_OnFinalize ()" is available as a sample function for adding those processing. Add the required processing while referring to that sample.

This function takes the following actions:

- To perform a CEU software reset
- To call the callback function specified in the argument.

6.6 R_CEU_InterruptEnable

R_CEU_InterruptEnable

Synopsis	Interrupt enable setup		
Header	r_ceu.h		
Declaration	`void R_CEU_InterruptEnable( const ceu_int_type_t int_type, void (* const callback)(ceu_int_type_t) );`		
Arguments	[IN]	ceu_int_type_t int_type	: CEU interrupt type selection
	[IN]	void (*callback)(ceu_int_type_t)	: Callback function to be registered Specify NULL if not necessary.
Return value	None		
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 interrupt priorities are checked by the function described in Section 6.9, "R_CEU_OnInitialize()" which is introduced as an example of registering an interrupt handler. If two or more callback functions are registered, only the last registered one can be used, and others are ignored. .

- To enable the types of CEU interrupts specified as an argument
- To register the callback function registered as an argument

6.7 R_CEU_InterruptDisable

R_CEU_InterruptDisable		
Synopsis	Interrupt disable setup	
Header	r_ceu.h	
Declaration	`void R_CEU_InterruptDisable( void );`	
Arguments	[IN]	None
Return value	None	
Remarks		

(1) Description

This function takes the following actions:

- To disable all types of CEU interrupts.
- To clear the registered callback function.

6.8 CEU_Isr

CEU_Isr			
Synopsis	Interrupt handler		
Header	r_ceu.h		
Declaration	`void CEU_Isr( const uint32_t int_sense );`		
Arguments	[IN]	uint32_t int_sense	: Interrupt Request Edge/Level
Return value	None		
Remarks			

(1) Description

This function serves as the interrupt handler to be used by the CEU sample driver.

The function is registered as the CEU interrupt handler through the function described in Section 6.9, "R_CEU_OnInitialize()" which is introduced as an example of interrupt handler registration processing.

6.9 R_CEU_OnInitialize

R_CEU_OnInitialize			
Synopsis	Sample processing for releasing the CEU standby state and registering an interrupt handler		
Header	r_ceu_user.h		
Declaration	`void R_CEU_OnInitialize ( const uint32_t user_num );`		
Arguments	[IN]	uint32_t user_num	: User parameter
Return value	None		
Remarks			

(1) Description

This function is introduced as an example of releasing the CEU module standby state and registering an interrupt handler. This function takes the following actions:

- To perform CEU standby release processing
- To register an interrupt handler
- To set up interrupt priorities

6.10 R_CEU_OnFinalize

R_CEU_OnFinalize			
Synopsis	Sample for setting up CEU standby state and releasing the interrupt handler		
Header	r_ceu_user.h		
Declaration	`void R_CEU_OnFinalize( const uint32_t user_num );`		
Arguments	[IN]	uint32_t user_num	: User parameter
Return value	None		
Remarks			

(1) Description

This function is introduced as an example of setting up the CEU module standby state and releasing the interrupt handler. This function takes the following actions:

- To set up the CEU standby state
- To release the interrupt handler

7. Processing Flow of the Application Using the CEU Sample Driver

Figure 7-1 shows the processing flow of the application using the CEU sample driver.

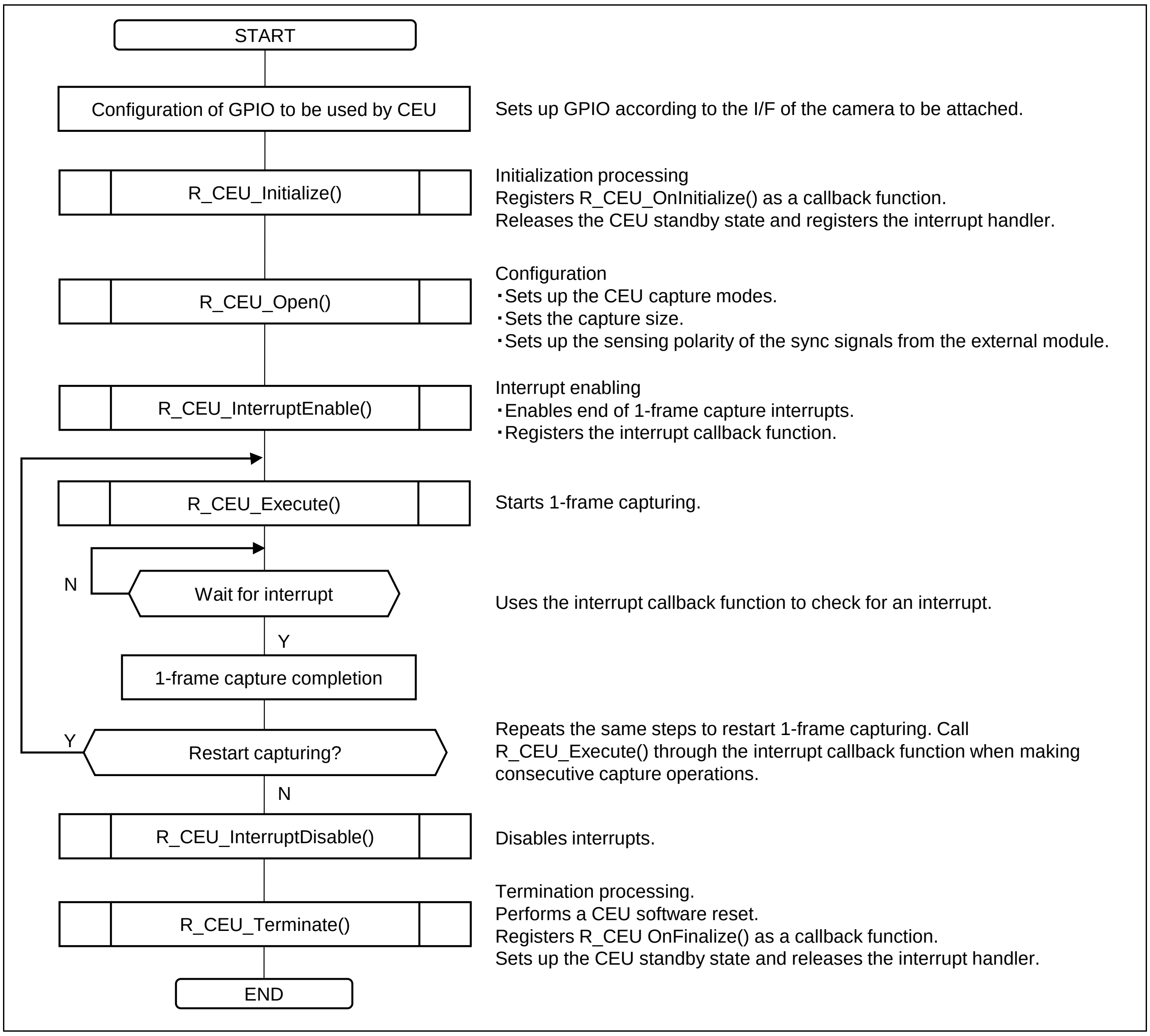
START
Configuration of GPIO to be used by CEU
Sets up GPIO according to the I/F of the camera to be attached.
R_CEU_Initialize()
Initialization processing
Registers R_CEU_OnInitialize() as a callback function.
Releases the CEU standby state and registers the interrupt handler.
R_CEU_Open()
Configuration
・Sets up the CEU capture modes.
・Sets the capture size.
・Sets up the sensing polarity of the sync signals from the external module.
R_CEU_InterruptEnable()
Interrupt enabling
・Enables end of 1-frame capture interrupts.
・Registers the interrupt callback function.
R_CEU_Execute()
Starts 1-frame capturing.
N
Wait for interrupt
Uses the interrupt callback function to check for an interrupt.
Y
1-frame capture completion
Y
Restart capturing?
Repeats the same steps to restart 1-frame capturing. Call R_CEU_Execute() through the interrupt callback function when making consecutive capture operations.
N
R_CEU_InterruptDisable()
Disables interrupts.
R_CEU_Terminate()
Termination processing.
Performs a CEU software reset.
Registers R_CEU OnFinalize() as a callback function.
Sets up the CEU standby state and releases the interrupt handler.
END

Figure 7-1 Processing Flow of the Application Using the CEU Sample Driver

The GPIO configuration of the pins to be used by the CEU must be accomplished by the user application since it is not carried out by the CEU sample driver.

8. How to Import the Driver

8.1 e^2 studio

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

8.2 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.

9. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware

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

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

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

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

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

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

The latest version can be downloaded from the ARM website.

ARM Cortex^{TM}-A9 (Revision: r4p1) Technical Reference Manual

The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification - Architecture version 2.0

The latest version can be downloaded from the ARM website.

ARM CoreLink^{TM} Level 2 Cache Controller L2C-310 (Revision: r3p3) Technical Reference Manual

The latest version can be downloaded from the ARM website.

Technical Update/Technical News

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

User's Manual: Development Tools

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.14.18	-	First edition issued
1.01	Dec.28.18	28	Addition “6.4 R_CEU_Stop”
1.02	Apr.15.19	11	Addition “ceu_edge_t”
		14	Addition the following parameters to Table 6-1. ・ceu_edge_t vdsel ・ceu_edge_t hdsel ・ceu_edge_t fldsel ・ceu_edge_t dsel
		15	Addition the processing content for the following parameters. ・ceu_edge_t vdsel ・ceu_edge_t hdsel ・ceu_edge_t fldsel ・ceu_edge_t dsel
		16,17	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
1.10	May.17.19	5	Table 9.1 Peripheral device used(1/2) Remove compiler option "-mthumb-interwork"
		30	Added “8.How to Import the Driver”

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

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

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

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

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

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

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

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

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

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

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