SH7268/7269 Group

JPEG Codec Unit "JCU" Sample Driver

R01AN2338EJ0104
Rev. 1.04
Nov.27 2017

Introduction

This application note describes the specification of the JPEG codec unit (in the following, JCU) driver of SH7268/SH7269.

Target Device

SH7268/SH7269 Group

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

TABLE OF CONTENTS

1. Outline

1.1 environment

The following this driver's development environment and Evaluation board.

CPU

SH7269

Development environment

HEW (SuperH RISC engine microcomputer software integrated development environment) Version 4.09.01

Renesas SuperH RISC engine Standard Toolchain Version 9.4.1.0

- SH C/C++ Compiler Version 9.04.00
- SH Assembler Version 7.01.02
- SH C/C++ Standard Library Generator Version 3.00.03
- Optimizing Linkage Editor Version 10.00.01

Evaluation board

SH7269 CPU board (Part number: R0K572690C000BR)

SH7269 VDC4 board (Part number: R0K572690B000BR)

1.2 Functions

The following table describes functions of the JCU driver.

Table 1 Functions of JCU driver

Common functions	Specifications	JPEG baseline standard(ISO/IEC10918-1:1994). The JCUA driver does not support the following basic features: ● Scanning with two elements. ● Non-interleave scanning with multiple elements.
	Operational precision	Conforming to JPEG Part 2, ISO-IEC10918-2.
	Image input/output system	Block interleave method.
	Image data rate:	Max. 133.34 Mbytes/s (at 66.67-MHz operation)
	Subsampling	The buffer size can be reduced by using the mode in which data transfer is temporarily stopped each time the specified number of lines or the specified amount of data is transferred during image data or coded data input.
	Processing unit	8-byte address boundary units can be set
	Image sizes that can be processed:	Sizes divisible by the minimum coded unit (MCU): 8 lines by 16 pixels in YCbCr422 16 lines by 16 pixels in YCbCr420 Compression and decompression processing of images in 0-line or 0-pixel image sizes should be avoided. Following formats are not supported in this driver. YCbCr4:4:4, YCbCr4:1:1
Compression functions	Input pixel format	YCbCr422
	Output format	JPEG baseline standard (YCbCr4:2:2)
	Quantization table	4 quantization tables provided
	Huffman tables	4 Huffman tables provided. 2 tables for AC coefficients. 2 tables for DC coefficients.
	Markers supported	SOI/SOF0/SOS/DQT/DHT/DRI/RSTm/EOI
Decompression functions	Input format	JPEG baseline standard (YCbCr4:2:2 or YCbCr4:2:0) Decompression processing of images in unsupported pixel formats should be avoided.
	Output pixel format	YCbCr4:2:2, ARGB8888, RGB565

1.3 File structure

The following table describes file structure of the JCU driver. (OSPL file isn't included.)

Table 2 file structure

File Name	Description
jcu_api.c	Source file for JCU driver functions. (main)
jcu_para.c	Source file checking arguments.
jcu_reg.c	Source file controlling registers.
jcu_pl.c	Source file for JCU driver functions. (interrupt handlers, and porting layer)
jcu_misc.c	Source file for JCU driver functions. (other)
r_jcu_api.h	Header file including the prototype declarations for the JCU driver calls and definitions of constants.
r_jcu_local.h	Header file including local definitions.
r_jcu_pl.h	Header file including interrupt handlers and porting layer)
jcu_namecnv.h	Header file for old naming rule.
r_jcu_user.h	Header file for compilation option.

The following table describes file structure of the external header file for JCU.

Table 3 file structure of the external header file

File Name	Description
typedefine.h, r_typedefs.h	Header file including the typedef declarations for the basic types.
iodefine.h	Header file including IO definitions.
r_ospl.h	Header file including OS porting layer.

1.4 Program size and section

The following table describes the program size and section of the JCU driver.

Table 4 program size and section

Type	Section	Size[byte]	Description
ROM	P_JCU	10552	Program area
	C_JCU	1421	Constant area
	D_JCU	0	Initialized data area
RAM	B_JCU	104	Uninitialized data area
	(Stack)	664	Used stack size of the sample program

Note: "Size" doesn't include RAM of output data.
"Size" changes by the value of the optimization option.
This data is set by "Speed & size optimization enabled" of "Renesas SuperH RISC engine Standard Toolchain 9.3.2.0"

1.5 Concept

The JCU driver supports to encode/decode functions. These functions can be used exclusively.

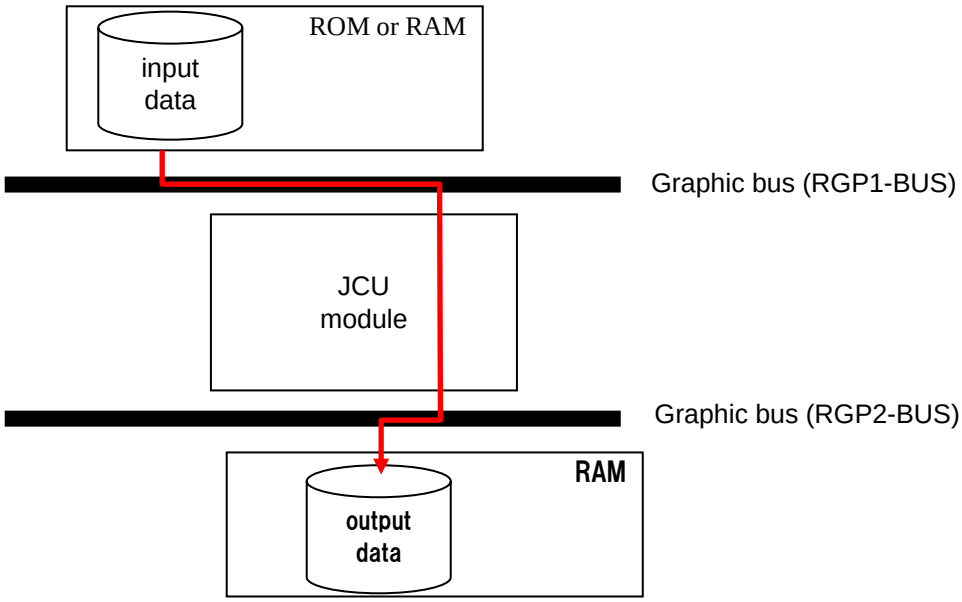


Figure 1 Basic processing

1.6 State transition

The JCU driver manages operating state and judges the propriety of the processing according to the state. This state is to use API, and it changes. The image describes state transition.

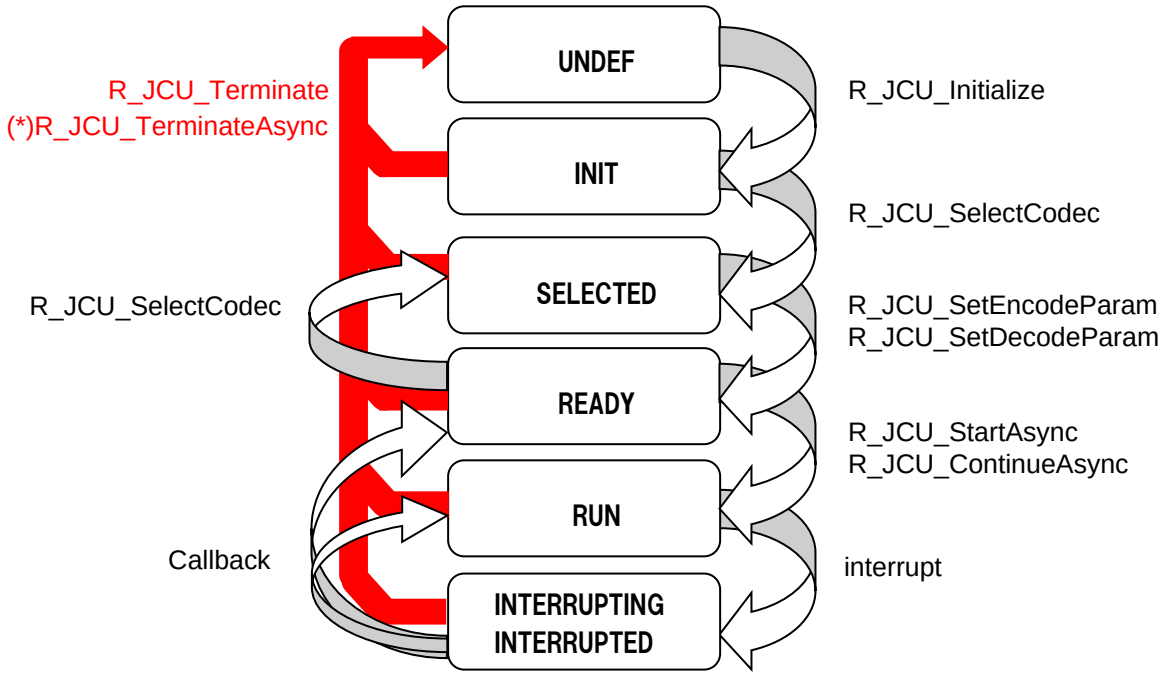


Figure 2 JCU driver state transition

The following table describes API function state transition.

Table 5 state transition table

API	The state that a calling is possible						The state after execution
	UNDEF	INIT	SELECTED	READY	RUN	INTERRUPTING INTERRUPTED	
for Encode or Decode							
R_JCU_Initialize	OK	NG	NG	NG	NG	NG	INIT
R_JCU_Terminate	OK	OK	OK	OK	OK	OK	UNDEF
R_JCU_TerminateAsync	OK	OK	OK	OK	OK*1	OK	UNDEF
R_JCU_SelectCodec	NG	OK	OK	OK	NG	NG	SELECTED
R_JCU_SetCountMode	NG	NG	OK	OK	NG	NG	(Stay)
R_JCU_Start	NG	NG	NG	OK	NG	NG	(Stay)
R_JCU_StartAsync	NG	NG	NG	OK	NG	NG	RUN
R_JCU_Continue	NG	NG	NG	OK	NG	NG	(Stay)
R_JCU_ContinueAsync	NG	NG	NG	OK	NG	NG	RUN
R_JCU_GetAsyncStatus	OK	OK	OK	OK	OK	OK	(Stay)
R_JCU_OnInterrupting*2	NG	NG	NG	NG	NG	OK	INTERRUPTED
R_JCU_OnInterrupted	NG	NG	NG	NG	NG	OK	READY or RUN
for Decode							
R_JCU_SetPauseForImageInfo	NG	NG	OK	OK	NG	NG	(Stay)
R_JCU_SetErrorFilter	NG	OK	OK	OK	NG	NG	(Stay)
R_JCU_SetDecodeParam	NG	NG	OK	OK	NG	NG	Ready
R_JCU_GetImageInfo	NG	NG	NG	OK*3	NG	NG	(Stay)
R_JCU_GetErrorInfo	NG	NG	OK	OK	NG	NG	(Stay)
for Encode							
R_JCU_SetEncodeParam	NG	NG	OK	OK	NG	NG	Ready
R_JCU_SetQuantizationTable	NG	NG	OK	OK	NG	NG	(Stay)
R_JCU_SetHuffmanTable	NG	NG	OK	OK	NG	NG	(Stay)
R_JCU_GetEncodedSize	NG	NG	NG	OK	NG	NG	(Stay)

*1: Just after execution in the RUN state, the status is stay. After it, the state transfers in UNDEF by the timing which interrupted (and callback function executed).

*2: It's called in the default callback of OSPL. Just before calling, the state transfer in INTERRUPTING state.

*3: After JCU_INT_GET_IMAGE_INFO interrupt occurred, it's possible to check image information.

1.7 Interrupt handler

The following table describes interrupt handlers.

Table 6 Interrupt handler

Interrupt	Vector		Handler
	Number	Address	
JEDI Compression/Decompression Process Interrupt Request	181	0x000002D4 ~ 0x000002D7	void INT_JCU_JEDI (void);
JDTI Data Transfer Interrupt Request	182	0x000002D8 ~ 0x000002DB	void INT_JCU_JDTI (void);

The user has to register the function as the interrupt handler for the user to use interrupt function of JCU. At the case of using OS, register the interrupt handler function by OS function. In other case, register the function in the vector table.

1.8 Compiler switch

In this driver, the compiler switch is defined by "jcu_user.h" file.

1.8.1 Parameter check

When "JCU_PARAMETER_CHECK" is defined, an argument of API is checked. When a parameter is wrong, an error code is returned. See 2.1.2(2) jcu_errorcode_t and 2.1.2(5) jcu_detail_error_t.

1.8.2 Interrupt handler definition

In this driver, the function for interrupt handler exists. (see 1.7Interrupt handler) When not using a RTOS, don't define "IS_USE_RTOS". By a declaration of "#pragma interrupt", the function becomes as an interrupt function. In this case, user has to register the handler function in an interrupt vector table statically.

When interrupt function handlers are registered by using OS function, define "IS_USE_RTOS".

1.9 Limitation

1.9.1 Reserved word

In this driver, prefix "JCU_" is added to the symbol name, to classify as other programs. Please don't use the symbol which starts from "JCU".

1.9.2 Stop during processing

In the H/W of JCU, there are no functions stopped during processing. After processing of the encoding or decoding which is executed, please begin to the next process.
Please don't stop during processing by software-reset or module standby.

1.9.3 Output subsampling processing.

The output subsampling function (encode or decode) was removed from the SH7269 H/W specification.

2. API

2.1 Data definition

2.1.1 Basic types

The following table describes basic type definition. The basic type definition is defined in " r_typedefs.h". See 1.3

Table 7 Basic types

Basic type	Description
int8_t	typedef signed char
uint8_t	typedef unsigned char
int16_t	typedef signed short
uint16_t	typedef unsigned short
int32_t	typedef signed int
uint32_t	typedef unsigned int
char_t	typedef char
bool_t	typedef int
int_fast32_t	typedef int
uint_fast32_t	typedef unsigned int

2.1.2 Constant

The following table describes variable type, constant value. Constant is #define macro or enum.

Table 8 Constant

Section	Constant
(1)	Version
(2)	jcu_errorcode_t
(3)	jcu_codec_t
(4)	jcu_continue_type_t
(5)	jcu_detail_error_t
(6)	jcu_int_detail_error_t
(7)	jcu_int_detail_errors_t
(8)	jcu_interrupt_line_t
(9)	jcu_interrupt_lines_t
(10)	jcu_swap_t
(11)	jcu_sub_sampling_t
(12)	jcu_decode_format_t
(13)	jcu_jpeg_format_t
(14)	jcu_huff_t
(15)	jcu_table_no_t
(16)	jcu_color_element_t
(17)	jcu_status_information_t
(18)	jcu_codec_status_t
(19)	jcu_sub_state_t
(20)	jcu_sub_status_t

(1) Version

Symbol	Value	Description
JCU_VERSION	103	JCU version number.
JCU_VERSION_STRING	"1.03"	Character string of the JCU version number.

(2) jcu_errorcode_t

This is the variable type of error code. The following constant value is used for a variable of jcu_errorcode_t type.

Symbol	Value	Description
JCU_ERROR_OK	0x0000	No error has occurred.
JCU_ERROR_PARAM	0x4501	A parameter provided to a function is incorrect.
JCU_ERROR_STATUS	0x4502	A function was called in an incorrect state.
JCU_ERROR_CODEC_TYPE	0x4503	A function was called in an incorrect mode.
JCU_ERROR_LIMITATION	0x4504	Limitations on JCU driver.

(3) jcu_codec_t

This is the constant value of "ENCODE" or "DECODE" process.

Symbol	Value	Description
JCU_ENCODE	0	Encode(Compression) process.
JCU_DECODE	1	Decode(De-compression) process.

(4) jcu_continue_type_t

This is the constant value of paused factor (continue mode).

Symbol	Value	Summary
JCU_INPUT_BUFFER	0	Resumes reading input image data.
JCU_OUTPUT_BUFFER	1	This value can't be used.
JCU_GET_IMAGE_INFO	2	Restart the process after reading the image information.

(5) jcu_detail_error_t

This is the variable type of error classification of the JCU driver. The following constant value is used for a variable of jcu_detail_error_t.

About details, refer to table 40.3 and 40.4 of the SH7268/7269 Group User's Manual: Hardware

Symbol	Value	Summary
JCU_JCDERR_OK	0x0000	Normal
JCU_JCDERR_SOI_NOT_FOUND	0x4521	SOI not detected: SOI not detected until EOI detected
JCU_JCDERR_INVALID_SOF	0x4522	SOF1 to SOFF detected
JCU_JCDERR_UNPROVIDED_SOF	0x4523	Unprovided pixel format detected
JCU_JCDERR_SOF_ACCURACY	0x4524	SOF accuracy error: Other than 8 detected
JCU_JCDERR_DQT_ACCURACY	0x4525	DQT accuracy error: Other than 0 detected

JCU_JCDERR_COMPONENT_1	0x4526	Component error 1: The number of SOF0 header components detected is other than 1, 3, or 4
JCU_JCDERR_COMPONENT_2	0x4527	Component error 2: The number of components differs between SOF0 header and SOS
JCU_JCDERR_NO_SOF0_DQT_DHT	0x4528	SOF0, DQT, and DHT not detected when SOS detected
JCU_JCDERR_SOS_NOT_FOUND	0x4529	SOS not detected: SOS not detected until EOI detected
JCU_JCDERR_EOI_NOT_FOUND	0x452A	EOI not detected (default)
JCU_JCDERR_RESTART_INTERVAL_NUM	0x452B	Restart interval data number error detected
JCU_JCDERR_IMAGE_SIZE	0x452C	Image size error detected*
JCU_JCDERR_LAST_MCU_NUM	0x452D	Last MCU data number error detected
JCU_JCDERR_BLOCK_NUM	0x452E	Block data number error detected

* When there are except for EOI marker behind the compressed data part (it has no huffman encoding segments and markers.), JCU_JCDERR_IMAGE_SIZE error sometimes occurs. When bits of the JCU_INT_ERROR_SEGMENT_TOTAL_DATA and the JCU_INT_ERROR_MCU_BLOCK_DATA is passed to the R_JCU_SetErrorFilter function (the part of JINTE0 register), JCU_JCDERR_IMAGE_SIZE error will not be detected any more.

(6) jcu_int_detail_error_t

This is the variable type of particular error code. The following constant value is used for a variable of jcu_int_detail_error_t type.

Symbol	Value	Summary
JCU_INT_ERROR_RESTART_INTERVAL_DATA	0x80u	The number of data in the restart interval of the Huffman-coding segment is not correct in de-compression.
JCU_INT_ERROR_SEGMENT_TOTAL_DATA	0x40u	The total number of data in the Huffman-coding segment is not correct in de-compression.
JCU_INT_ERROR_MCU_BLOCK_DATA	0x20u	The final number of MCU data in the Huffman-coding segment is not correct in de-compression.
JCU_INT_ERROR_ALL	0xE0	All errors.

(7) jcu_int_detail_errors_t

This is the variable type of bitwise OR operated bit flag values defined by jcu_int_detail_error_t. It's possible to use the value of jcu_int_detail_error_t type for a variable of jcu_int_detail_errors_t type.

```
typedef bit_flags_fast32_t  jcu_int_detail_errors_t;
```

(8) jcu_interrupt_line_t

This is the variable type of the kind of interrupt as the bit flag value. The following constant value is jcu_interrupt_line_t type used for a variable of jcu_interrupt_lines_t type.

Symbol	Value	Summary
JCU_INTERRUPT_LINE_JEDI	0x00000001u	Interrupt of JEDI.

JCU_INTERRUPT_LINE_JDTI	0x00000002u	Interrupt of JDTI.
JCU_INTERRUPT_LINE_ALL	0x00000003u	Interrupt of both of JEDI and JDTI.

(9) jcu_interrupt_lines_t

This is the variable type of bitwise OR operated bit flag values defined by jcu_interrupt_line_t. It's possible to use the value of jcu_interrupt_line_t type for a variable of jcu_interrupt_lines_t type.

```
typedef bit_flags_fast32_t  jcu_interrupt_lines_t;
```

(10) jcu_swap_t

This is the variable type of swap setting. The following constant value is used for a variable of jcu_swap_t type.

Symbol	Value	Summary
JCU_SWAP_NONE	0x00	No swap.
JCU_SWAP_BYTE	0x01	Byte swap.
JCU_SWAP_WORD	0x02	Word swap.
JCU_SWAP_WORD_AND_BYTE	0x03	Word-byte swap.
JCU_SWAP_LONG_WORD	0x04	Longword swap.
JCU_SWAP_LONG_WORD_AND_BYTE	0x05	Longword-byte swap.
JCU_SWAP_LONG_WORD_AND_WORD	0x06	Longword-word swap.
JCU_SWAP_LONG_WORD_AND_WORD_AND_BYTE	0x07	Longword-word-byte swap.

(11) jcu_sub_sampling_t

This is the variable type of subsampling parameter for decoding. The following constant value is used for a variable of jcu_sub_sampling_t type.

Symbol	Value	Summary
JCU_SUB_SAMPLING_1_1	0x00	No subsampling.
JCU_SUB_SAMPLING_1_2	0x01	Subsamples output data into 1/2.
JCU_SUB_SAMPLING_1_4	0x02	Subsamples output data into 1/4.
JCU_SUB_SAMPLING_1_8	0x03	Subsamples output data into 1/8.

(12) jcu_decode_format_t

This is the variable type of output pixel format of RAW image data. The following constant value is used for a variable of jcu_decode_format_t type.

Symbol	Value	Summary
JCU_OUTPUT_YCbCr422	0x00	YCbCr4:2:2
JCU_OUTPUT_ARGB8888	0x01	ARGB8888
JCU_OUTPUT_RGB565	0x02	RGB565

(13) jcu_jpeg_format_t

This is the variable type of pixel format in JPEG image data. The following constant value is used for a variable of jcu_jpeg_format_t type.

Symbol	Value	Summary
JCU_JPEG_YCbCr422	0x01	YCbCr4:2:2
JCU_JPEG_YCbCr420	0x02	YCbCr4:2:0

(14) jcu_huff_t

This is the variable type of Huffman table entropy type (AC or DC). The following constant value is used for a variable of jcu_huff_t type.

Symbol	Value	Summary
JCU_HUFFMAN_AC	0x00	AC
JCU_HUFFMAN_DC	0x01	DC

(15) jcu_table_no_t

This is the variable type of Quantization table number or Huffman table number. The following constant value is used for a variable of jcu_table_no_t type.

Symbol	Value	Summary
JCU_TABLE_NO_0	0x00	Quantization table No. 0 (JCQTBL0), or DC/AC Huffman table No. 0 (JCHTBD0 / JCHTBA0)
JCU_TABLE_NO_1	0x01	Quantization table No. 1 (JCQTBL1), or DC/AC Huffman table No. 1 (JCHTBD1 / JCHTBA1)
JCU_TABLE_NO_2	0x02	Quantization table No. 2 (JCQTBL2) It can't be used at the Huffman table.
JCU_TABLE_NO_3	0x03	Quantization table No. 3 (JCQTBL3) It can't be used at the Huffman table.

(16) jcu_color_element_t

This is the variable type of component identifier of Quantization table or Huffman table. The following constant value is used for a variable of jcu_color_element_t type.

Constant	Value	Summary
JCU_ELEMENT_Y	0x00	Y (luminance component) table.
JCU_ELEMENT_Cb	0x01	Cb (blue-difference chroma component) table.
JCU_ELEMENT_Cr	0x02	Cr (red-difference chroma component) table.

(17) jcu_status_information_t

This is the variable type of main status of the JCU driver. The following constant value is used for a variable of jcu_status_information_t type.

Symbol	Value	Summary
JCU_STATUS_UNDEF	0x00	The JCU is uninitialized status.
JCU_STATUS_INIT	0x01	The JCU is initialized status.
JCU_STATUS_SELECTED	0x02	The JCU mode is selected.
JCU_STATUS_READY	0x08	The JCU decode/encode is ready, or the JCU decode/encode has been completed.
JCU_STATUS_RUN	0x10	The JCU decode/encode being executed.
JCU_STATUS_INTERRUPTING	0x40	The state that interrupt occurred.
JCU_STATUS_INTERRUPTED	0x80	The state after interrupt function executed.

(18) jcu_codec_status_t

This is the variable type of mode selection information. The following constant value is used for a variable of jcu_codec_status_t.

Symbol	Value	Summary
JCU_CODEC_NOT_SELECTED	-1	The state of the JCU mode is not selected.
JCU_STATUS_ENCODE	0	The state of the JCU mode is JCU_ENCODE.
JCU_STATUS_DECODE	1	The state of the JCU mode is JCU_DECODE.

(19) jcu_sub_state_t

This is the variable type of JCU sub status. The following fixed number is used for a variable of jcu_sub_state_t.

Symbol	Value	Summary
JCU_SUB_INFOMATION_READY	0x00000008	The JCU decode paused, when the image size and pixel format can be read.
JCU_SUB_DECODE_OUTPUT_PAUSE	0x00000100	The JCU decode paused, when the last output image data is written in decompression.
JCU_SUB_DECODE_INPUT_PAUSE	0x00000200	The JCU decode paused, when the amount of input coded data is read in decompression.
JCU_SUB_ENCODE_OUTPUT_PAUSE	0x00001000	The JCU encode paused, when the last output jpeg data is written in decompression.
JCU_SUB_ENCODE_INPUT_PAUSE	0x00002000	The JCU encode paused, when the number of input image data lines is read in compression.
JCU_SUB_PAUSE_ALL	0x00003308	All logical sum of jcu_sub_state_t type

(20) jcu_sub_status_t

This is the variable type of bitwise OR operated bit flag values defined by jcu_sub_state_t. It's possible to use the value of jcu_sub_state_t type for a variable of jcu_sub_status_t type.

```
typedef uint_fast32_t  jcu_sub_status_t;
```

2.1.3 Structures

The following table describes structures.

Table 9 Structures

Section	Structure
(1)	jcu_count_mode_param_t
(2)	jcu_buffer_t
(3)	jcu_buffer_param_t
(4)	jcu_decode_param_t
(5)	jcu_image_info_t
(6)	jcu_encode_param_t
(7)	jcu_async_status_t
(8)	jcu_internal_information_t
(9)	jcu_i_lock_t

(1) jcu_count_mode_param_t

Parameters for the count mode (division process). "inputBuffer" means an input side in JCU, "outputBuffer" means an output side in JCU.

The output subsampling function (encode or decode) can't be used by this driver. outputBuffer.isEnable must be set "false".

```
typedef struct {
    struct {
        bool_t      isEnable;
        bool_t      isInitAddress;
        uint32_t*    restartAddress;
        uint32_t    dataCount;
    } inputBuffer;
    struct {
        bool_t      isEnable;
        bool_t      isInitAddress;
        uint32_t*    restartAddress;
        uint32_t    dataCount;
    } outputBuffer;
} jcu_count_mode_param_t;
```

Member	Summary
isEnable	false: Disable the division processing on input/output buffer. true: Enable the division processing on input buffer. **Output side must be false.**
isInitAddress	false: When decoding paused, the input address isn't initialized. true: When decoding paused, the input address is initialized by "inputBuffer.restartAddress".
restartAddress	If "IsInitAddress" is "true", the input data address is initialized by this value.
dataCount	The division size of the input buffer. In the case of decoding mode, when data of "dataCount" byte count is input to JCU, it pauses. In the case of encoding mode, when data of "dataCount" line count is input to JCU, it pauses.

	The "dataCount" must be a multiple of 8 bytes.

(2) jcu_buffer_t

Structure for the input/output buffer setting.

```
typedef struct {
    jcu_swap_t     swapSetting;
    uint32_t*      address;
} jcu_buffer_t;
```

Member	Summary
swapSetting	Byte/Word/Longword Swap.
address	Buffer address.

(3) jcu_buffer_param_t

Parameters for the input/output buffer setting in encode/decode.

```
typedef struct {
    jcu_buffer_t   source;
    jcu_buffer_t   destination;
    int16_t        lineOffset;
} jcu_buffer_param_t;
```

Member	Summary
source	Input buffer.
destination	Output buffer.
lineOffset	Line offset.

(4) jcu_decode_param_t

Parameters for the option setting in de-compression.

```
typedef struct {
    jcu_sub_sampling_t   verticalSubSampling;
    jcu_sub_sampling_t   horizontalSubSampling;
    jcu_decode_format_t  decodeFormat;
    uint8_t              alpha;
} jcu_decode_param_t;
```

Member	Summary
verticalSubSampling	Vertical subsampling.
horizontalSubSampling	Horizontal subsampling.
decodeFormat	The output pixel format of RAW image data.
alpha	Alpha value setting. If the pixel format isn't ARGB8888, the alpha value has to be zero.

(5) jcu_image_info_t

Structure for the image information of the decoded JPEG data.

```
typedef struct {
    uint32_t            width;
    uint32_t            height;
    jcu_jpeg_format_t   encodedFormat;
} jcu_image_info_t;
```

Member	Summary
width	The width of the image data.
height	The height of the image data
encodedFormat	The pixel format of original JPEG data.

(6) jcu_encode_param_t

Parameters for the option setting in compression.

```
typedef struct {
    jcu_jpeg_format_t    encodeFormat;
    int_t                QuantizationTable[JCU_COLOR_ELEMENT_NUM];
    int_t                HuffmanTable[JCU_COLOR_ELEMENT_NUM];
    uint32_t             DRI_value;
    uint32_t             width;
    uint32_t             height;
} jcu_encode_param_t;
```

Member	Summary
encodeFormat	The pixel format of compressed JPEG data. This value has to be JCU_JPEG_YCbCr422.
QuantizationTable	Quantization table.
HuffmanTable	Huffman table.
DRI_value	DRI (Define Restart Interval) value.
width	The width of the input image data.
height	The height of the input image data.

(7) jcu_async_status_t

The JCU driver state and interrupt status.

```
typedef struct st_jcu_async_status_t  jcu_async_status_t;
struct st_jcu_async_status_t {
    jcu_status_information_t   Status;
    jcu_sub_status_t           SubStatusFlags;
    bool_t                     IsPaused;
    bool_t                     IsEnabledInterrupt;
    r_ospl_flag32_t            InterruptEnables;
    r_ospl_flag32_t            InterruptFlags;
    r_ospl_flag32_t            CancelFlags;
};
```

Member	Summary
Status	Internal status of the JCU driver.
SubStatusFlags	Internal sub status of the JCU driver.
IsPaused	false: JCU driver is not paused true: JCU driver is paused.
IsEnabledInterrupt	false: JCU driver's I-lock object was already locked. true: JCU driver's I-lock object was unlocked.
InterruptEnables	Interruption of JCU is registered.
InterruptFlags	The flag managed in the JCU interrupt function.
CancelFlags	The flag referred in the JCU interrupt function.

(8) jcu_internal_information_t

Structure for the internal state of the JCU driver.

```
typedef struct {
    jcu_codec_status_t            Codec;
    bool_t                        IsCountMode;
    jcu_int_detail_errors_t       ErrorFilter;
    jcu_async_status_t            AsyncStatus;
    r_ospl_caller_t               InterruptCallbackCaller;
    jcu_i_lock_t*                 I_Lock;
    const r_ospl_i_lock_vtable_t* I_LockVTable;
    bool_t                        Is_I_LockMaster;
    r_ospl_async_t*               AsyncForFinalize;
} jcu_internal_information_t;
```

Member	Summary
Codec	Mode selection information.
IsCountMode	false: JCU driver is not count mode. true: JCU driver is count mode.
ErrorFilter	The valid decoding error code(jcu_int_detail_error_t) as the bit flag value.
AsyncStatus	The status of the interrupt and the asynchronous process.
InterruptCallbackCaller	The interrupt callback function registered with OSPL.
I_Lock	I-Lock status.
I_LockVTable	Indexes of the function which does I-Lock control.
IS_I_LockMaster	false: I_LockVTable is not set. true: I_LockVTable is set.
AsyncForFinalize	Parameter of OSPL.

(9) jcu_i_lock_t

Structure for the I-Lock state of the JCU driver.

```
struct st_jcu_i_lock_t {
  bool_t       IsLock;
  bool_t       IsRequestedFinalize;
};
```

Member	Summary
IsLock	false: JCU driver doesn't set I-Lock. true: JCU driver set I-Lock.
IsRequestedFinalize	false: JCU driver isn't requested finalize. true: JCU driver is requested finalize.

2.1.4 OS porting layer (OSPL)

In this driver, when calling the function of the OS porting layer(OSPL), the general-purpose type of OSPL is used as an argument and a return value.

The following table describes the general-purpose type of OSPL. Please refer to OS porting layer "OSPL" Application Note for SH7268/7269.

Table 10 general-purpose type of OSPL

Name	Description
errnum_t	Error Codes
r_ospl_async_t	Setting of notifications.
r_ospl_flag32_t	Flag having 32-bit.
r_ospl_interrupt_t	Structure related to interrupt source.
r_ospl_caller_t	Manager of an interrupt callback function.
r_ospl_i_lock_vtable_t	Structure related to I-Lock.
r_ospl_async_type_t	Kind of the asynchronous operation.

2.2 API Function

2.2.1 R_JCU_Initialize

API	jcu_errorcode_t R_JCU_Initialize(void* const NullConfig);	
Header	#include "r_jcu_api.h"	
Parameter	[in] void* const NullConfig	Please set a null.
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_STATUS	Error Code No error has occurred. A function was called in an incorrect state.
Description	In this function, the following processing executed. User defined function (R_JCU_OnInitialize) executes. Initialize of driver's management information. Initialize of the state inside the driver.	
Valid state	This API function is valid in the following state. -UNDEF Status	
Description	The state will be in the initialized status. Initializes the internal status(g_jcu_condition). The user defined function(R_JCU_OnInitialize) is called. Perform the following processing in the user defined function. 1. Clock supply to JCU. 2. Sets the priority of interrupt. 3. Sets the environment-depend process.	
Comment		

2.2.2 R_JCU_Terminate

API	jcu_errorcode_t R_JCU_Terminate(void);	
Header	#include "r_jcu_api.h"	
Parameter	void	
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM	Error Code No error has occurred. The return value of the user defined function is an error.
Description	In this function, the following processing executed (synchronous process). User defined function (R_JCU_OnFinalize) execute. The internal state of the driver transfers to JCU_STATUS_UNDEF.	
Valid state	This API function can execute every state. When "g_jcu_condition.status" is "JCU_STATUS_RUN", it waits until processing ends.	
Description	The processing which finishes a JCU driver. The function keeps executing until processing ends. The state will be in the uninitialized status. The user defined function(R_JCU_OnFinalize) is called. Perform the following processing in the user defined function. 1. Clock stopped to JCU. 2. Clear the priority of interrupt. 3. Sets the environment-depend process.	
Comment		

2.2.3 R_JCU_TerminateAsync

API	jcu_errorcode_t R_JCU_TerminateAsync(r_ospl_async_t* const async);

Header	#include "r_jcu_api.h"	
Parameter	[in] r_ospl_async_t* const async	General-purpose argument of OSPL. See. 2.1.4
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM	Error Code No error has occurred. The argument is NULL, other return value of the user defined function is an error.
Description	See. R_JCU_Terminate function. (asynchronous process).	
Valid state	This API function can execute every state. When the JCU Driver state used at JCU_STATUS_RUN, it won't be processed to complete immediately, and it return in the state which is just as it is. And when decoding or encoding ended, and JCU stopped, a processing terminated.	
Description	See. R_JCU_Terminate function.	
Comment	For argument, please refer to OS porting layer "OSPL" Application Note for SH7268/7269.	

2.2.4 R_JCU_SelectCodec

API	jcu_errorcode_t R_JCU_SelectCodec(const jcu_codec_t codec);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_codec_t codec	Codec
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	Error Code No error has occurred. An argument isn't right. A function was called in an incorrect state.
Description	In this function, the following processing executed. This function selects the JCU mode (Encode or Decode).	
Valid state	This API function is valid in the following state. INIT Status SELECTED Status READY Status	
Description	This function selects the JCU mode	
Comment	Please set again all parameters of decode, encode and count mode. Because when this function was called, these parameters were initialized.	

2.2.5 R_JCU_SetCountMode

API	jcu_errorcode_t R_JCU_SetCountMode(const jcu_count_mode_param_t* const buffer);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_count_mode_param_t* const buffer	count mode(division process).
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	Error Code No error has occurred. An argument isn't right. A function was called in an incorrect state.
Description	In this function, the following processing executed. Sets the count mode (division process).	
Valid state	This API function is valid in the following state. -SELECTED Status -READY Status	
Description	Sets the count mode. At SH7269 device, the output subsampling function (encode or decode) cannot be used. If the "JCU_PARAMETER_CHECK" symbol is defined and execute this function, it'll be an error. If undefined this symbol, a serious problem may occur.	

Before calling R_JCU_Start, Subsampling process is executed if user is registered in this API function.
The following table describes target data of jcu_count_mode_param_t for "inputBuffer" and "outputBuffer".

Mode	Target data of inputBuffer	Target data of outputBuffer
Encode	Input image data	-
Decode	Input JPEG data	-

The following table describes unit size for "inputBuffer" and "outputBuffer".

Mode	Unit size of inputBuffer	Unit size of outputBuffer
Encode	8Line unit	-
Decode	8byte unit	-

If "jcu_count_mode_param_t::inputBuffer.isEnable = false", the subsampling function is invalid.

Comment

2.2.6 R_JCU_SetPauseForImageInfo

API	jcu_errorcode_t R_JCU_SetPauseForImageInfo(const bool_t is_pause);	
Header	#include "r_jcu_api.h"	
Parameter	[in] const bool_t is_pause	true: It's made the setting which is paused. false: It's made the setting which is not paused.
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_STATUS	Error Code No error has occurred. A function was called in an incorrect state.
Description	When the image information can be acquired, it's made the setting which is paused.	
Valid state	In this function, the following processing executed. -SELECTED Status -READY Status And it's can be executed in case of the Decode mode.	
Description	When the image information can be acquired, it's made the setting which is paused by the R_JCU_GetImageInfo function.	
Comment		

2.2.7 R_JCU_SetErrorFilter

API	jcu_errorcode_t R_JCU_SetErrorFilter(jcu_int_detail_error_t filter);	
Header	#include "r_jcu_api.h"	
Parameter	[in] jcu_int_detail_error_t filter	The valid decoding error code(jcu_int_detail_error_t) as the bit flag value.
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	Error Code No error has occurred. An argument isn't right. A function was called in an incorrect state.
Description	The particular error code(jcu_int_detail_error_t) was set to valid.	
Valid state	In this function, the following processing executed. INIT Status SELECTED Status READY Status	
Description	The particular error code was set to valid.	

When the valid decoding error occurred, interrupt occurs internally and R_JCU_Start function returns error code.

Comment

2.2.8 R_JCU_Start

API	jcu_errorcode_t R_JCU_Start(void);	
Header	#include " r_jcu_api.h"	
Parameter	void	
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_STATUS	Error Code No error has occurred. A function was called in an incorrect state.
Description	Starts JCU process (synchronous process).	
Valid state	This API function is valid in the following state. READY Status	
Description	Starts JCU process. The function will not return until decoding or encoding ends or pauses. Using the R_JCU_SetDecoderParam API function or the R_JCU_SetEncoderParamSet API function, set the parameters before the JCU process starts You cannot stop the JCU process, after the JCU process starts.	
Comment	When the parameter at R_JCU_SetEncodeParam or R_JCU_SetDecodeParam function isn't right, this API function does not return an error.	

2.2.9 R_JCU_StartAsync

API	jcu_errorcode_t R_JCU_StartAsync(r_ospl_async_t* const async);	
Header	#include " r_jcu_api.h"	
Parameter	[in] r_ospl_async_t* const async	General-purpose argument of OSPL. See. 2.1.4
Return value	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_STATUS	Error Code No error has occurred. A function was called in an incorrect state.
Description	Starts JCU process (asynchronous process).	
Valid state	This API function is valid in the following state. READY Status	
Description	See. R_JCU_Start function. For argument, please refer to OS porting layer "OSPL" Application Note for SH7268/7269.	
Comment	When the parameter at R_JCU_SetEncodeParam or R_JCU_SetDecodeParam function isn't right, this API function doesn't return an error.	

2.2.10 R_JCU_Continue

API	jcu_errorcode_t R_JCU_Continue(const jcu_continue_type_t type);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_continue_type_t type	Paused factor(continue mode)
Return value	jcu_errorcode_t _t JCU_ERROR_OK JCU_ERROR_STATUS	Error Code No error has occurred. A function was called in an incorrect state.
Description	Resume the JCU process (synchronous process).	
Valid state	This API function is valid in the following state. READY Status	
Description	Processing of JCU which paused is resumed. The function will not return until decoding or encoding ends or pauses.	

	The parameter is a paused factor.
Comment	If the paused factor isn't right, this API function doesn't return an error.

2.2.11 R_JCU_ContinueAsync

API	jcu_errorcode_t R_JCU_Continue(const jcu_continue_type_t type , r_ospl_async_t* const async);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_continue_type_t type	Paused factor(continue mode)
	[in] r_ospl_async_t* const async	General-purpose argument of OSPL. See. 2.1.4
Return value	jcu_errorcode_t _t	Error Code
	JCU_ERROR_OK	No error has occurred.
	JCU_ERROR_STATUS	A function was called in an incorrect state.
Description	Resume the JCU process (asynchronous process).	
Valid state	This API function is valid in the following state.	
	READY Status	
Description	See. R_JCU_Continue function.	
	For argument, please refer to OS porting layer "OSPL" Application Note for SH7268/7269.	
Comment	If the paused factor isn't right, this API function doesn't return an error.	

2.2.12 R_JCU_GetAsyncStatus

API	void R_JCU_GetAsyncStatus(const jcu_async_status_t** const out_Status);	
Header	#include " r_jcu_api.h"	
Parameter	[out] const jcu_async_status_t** const out_Status	Pointer of a structure that indicates the state of the interrupt and asynchronous process.
Return value	None	
Description	Gets the pointer of a structure that indicates the state of the interrupt and asynchronous process.	
Valid state	This API function can execute every state.	
Description	Gets the pointer of a structure that indicates the state of the interrupt and asynchronous process.	
Comment	Pointer variable "out_Status" needs the const modifiers.	

2.2.13 R_JCU_OnInterrupting

API	errnum_t R_JCU_OnInterrupting(const r_ospl_interrupt_t* const InterruptSource);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const r_ospl_interrupt_t* const InterruptSource	Interruption sender. See.2.1.4.
Return value	errnum_t	Error information
	0	No error.
	E_OTHERS	Other error.
	E_STATE	Status error.
Description	Interrupt is accepted.	
Valid state	This function is not usually called from the user directly.	
	This function is callbacked from the interrupt callback function of the default.	
	This function sets the value of the interrupt status register to variable "gs_jcu_internal_information AsyncStatus.InterruptFlags". And, Interrupt request is cleared after it.	

	Default callback function "R_JCU_OnInterruptDefault" calls this function. It changes to INTERRUPTING status before calling. For the detail, see 3.1.7.
Description	A JCU driver notifies interrupts to oneself and clears by this function. Interrupt notice is used for a trigger of the resumption of the asynchronous process from event flag waiting.
Comment	Whether the R_JCU_OnInterrupted function is called continuously sets an event flag.

2.2.14 R_JCU_OnInterrupted

API	errnum_t R_JCU_OnInterrupted(void);	
Header	#include " r_jcu_api.h"	
Parameter	None	-
Return value	errnum_t	Error information
	0	No error.
	E_OTHERS	Other error.
	E_STATE	Status error.
	Each value of jcu_detail_error_t	Decode error
Description	Interrupt function is executed.	
Valid state	This function is not usually called from the user directly. This function is callbacked from the interrupt callback function of the default. Variable "gs_jcu_internal_information. AsyncStatus.InterruptFlags" the e function set in 1 is cleared in 0. And, interrupt function is executed. Default callback function "R_JCU_OnInterruptDefault" calls this function. It changes to INTERRUPTED status before calling. For the detail, see 2.2.13, 3.1.7.	
Description	The "interruption notice" from R_JCU_OnInterrupting function is cleared and interrupt handling operation is executed by this function. Interrupt handling operation does a trigger of the resumption of the asynchronous process from event flag waiting. When decoding error occurred, the value of jcu_detail_error_t type is returned in a return value of this function.	
Comment	When OSPL and callback processing was used by default, a return value of this API function is set to a ReturnValue member of an Async structure.	

2.2.15 R_JCU_SetDecodeParam

API	jcu_errorcode_t R_JCU_SetDecodeParam(const jcu_decode_param_t* const decode, const jcu_buffer_param_t* const buffer);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_decode_param_t* decode	Pointer to variable of decode parameter information.
	[in] const jcu_buffer_param_t* buffer	Pointer to variable of buffer.
Return value	jcu_errorcode_t	Error Code
	JCU_ERROR_OK	No error has occurred.
	JCU_ERROR_PARAM	An argument isn't right.
	JCU_ERROR_STATUS	A function was called in an incorrect state.
Description	Sets decoding parameter.	
Valid state	In this function, the following processing executed. SELECTED Status READY Status And it's can be executed in case of the Decode mode.	
Description	Sets decoding parameter.	
Comment	If the pixel format isn't ARGB8888, "decode.alpha" value has to set zero.	

2.2.16 R_JCU_GetImageInfo

API	jcu_errorcode_t R_JCU_GetImageInfo(jcu_image_info_t* const buffer);	
Header	#include " r_jcu_api.h"	
Parameter	[out] jcu_image_info_t* const buffer	Pointer to variable of image information.
Return value	jcu_errorcode_t	Error Code
	JCU_ERROR_OK	No error has occurred.
	JCU_ERROR_PARAM	An argument isn't right.
	JCU_ERROR_STATUS	A function was called in an incorrect state.
Description	Gets information on the JPEG data.	
Valid state	This API function is valid in the following state. READY Status And it's can be executed in case of the Decode mode.	
Description	Gets the image information(width, height, pixel format) of the decoded JPEG data. If the pixel format of the decoded JPEG data is outside of the jcu_jpeg_format_t, or the image size (wide or height) is zero, it's the error, so JCU can't decode.	
Comment	If data is read before the request which reads the image information, the data is not guaranteed.	

2.2.17 R_JCU_GetErrorInfo

API	jcu_errorcode_t R_JCU_GetErrorInfo(jcu_detail_error_t* const errorCode);	
Header	#include " r_jcu_api.h"	
Parameter	[out] jcu_detail_error_t* const errorCode	Pointer to variable of error information.
Return value	jcu_errorcode_t	Error Code
	JCU_ERROR_OK	No error has occurred.
	JCU_ERROR_PARAM	An argument isn't right.
	JCU_ERROR_STATUS	A function was called in an incorrect state.
Description	Gets information on the error data.	
Valid state	This API function is valid in the following state. READY Status And it's can be executed in case of the Decode mode.	
Description	When decoding error occurred, the reason of the error can be got from this function. For detail, see. 2.1.2(4) When a decoding error doesn't occur, the data is not guaranteed.	
Comment	This API function is the function equivalent to "JCU_GetErrorInfo" in the JCU driver before Ver0.09, it doesn't correspond to OSPL. This API function isn't necessary in the JCU driver after Ver0.10, that corresponded to OSPL. Because error information is returned from related API functions or is stored in a ReturnValue member of Async structure. For detail, see.2.1.4.	

2.2.18 R_JCU_SetEncodeParam

API	jcu_errorcode_t R_JCU_SetEncodeParam(const jcu_encode_param_t* const encode, const jcu_buffer_param_t* const buffer);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_encode_param_t* const encode	Pointer to variable of encode parameter information.
	[in] const jcu_buffer_param_t* const buffer	Pointer to variable of buffer.
Return value	jcu_errorcode_t	Error Code
	JCU_ERROR_OK	No error has occurred.
	JCU_ERROR_PARAM	An argument isn't right.
	JCU_ERROR_STATUS	A function was called in an incorrect state.

Description	Sets encoding parameter.
Valid state	This API function is valid in the following state. SELECTED Status READY Status And it's can be executed in case of the Encode mode.
Description	Sets Encoding parameter.
Comment	

2.2.19 R_JCU_SetQuantizationTable

API	jcu_errorcode_t R_JCU_SetQuantizationTable(const jcu_decode_format_t tableNo, const uint8_t* const table);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_decode_format_t tableNo	Quantization table number.
	[in] const uint8_t* const table	Quantization table.
Return value	jcu_errorcode_t	Error Code
	JCU_ERROR_OK	No error has occurred.
	JCU_ERROR_PARAM	An argument isn't right.
	JCU_ERROR_STATUS	A function was called in an incorrect state.
Description	Quantization table data. For the setting value of the quantization table data, see "SH7268, SH7269 Group User's Manual: Hardware" section 41.3.1(4), or use a quantization table generation tool of an accessory for a sample.	
Valid state	This API function is valid in the following state. SELECTED Status READY Status And it's can be executed in case of the Encode mode.	
Description	The data to which it was given at the table is set as the address of the chosen table number.	
Comment	Even when more than one picture data is encoded, this table data should be set once.	

2.2.20 R_JCU_SetHuffmanTable

API	jcu_errorcode_t R_JCU_SetHuffmanTable(const jcu_decode_format_t tableNo, const jcu_huff_t type, const uint8_t* const table);	
Header	#include " r_jcu_api.h"	
Parameter	[in] const jcu_decode_format_t tableNo	Huffman table number.
	[in] const jcu_huff_t type,	Type of Huffman table (AC or DC).
	[in] const uint8_t* const table	Huffman table
Return value	jcu_errorcode_t	Error Code
	JCU_ERROR_OK	No error has occurred.
	JCU_ERROR_PARAM	An argument isn't right.
	JCU_ERROR_STATUS	A function was called in an incorrect state.
Description	Sets the Huffman table. For the setting value of the Huffman table data, see "SH7268, SH7269 Group User's Manual: Hardware" section 41.3.1(4).	
Valid state	This API function is valid in the following state. SELECTED Status READY Status And it's can be executed in case of the Encode mode.	
Description	To the address selected by the table number and by the AC/DC data, Huffman table data is set/	
Comment	Even when more than one picture data is encoded, this table data should be set once.	

2.2.21 R_JCU_GetEncodedSize

API	jcu_errorcode_t R_JCU_GetEncodedSize (size_t* const out_Size);	
Header	#include " r_jcu_api.h"	
Parameter	[out] size_t* const out_Size	Pointer to variable of the data size.
Return value	jcu_errorcode_t JCU_ERROR_OK	Error Code No error has occurred.
Description	Gets the size of data to be compressed.	
Valid state	This API function can execute every state. If data is read before interrupt of encoding complete, the data is not guaranteed	
Description	Gets the size of JPEG data to be compressed. If data is read before interrupt of encoding complete, the data is not guaranteed.	
Comment		

3. Other function, define macro

3.1 User defined function

This driver's "jcu_pl.c" is a porting layer. Each function is possible to be modify as "User defined" functions. The following describes "User defined" functions.

3.1.1 R_JCU_OnInitialize

Function name	errnum_t R_JCU_OnInitialize(void);	
Header	#include "r_jcu_pl.h"	
Parameter	void	
Return value	errnum_t 0 E_OTHERS	Error information No error. Other error.
Description	Initializes the user defined process. By default, the following processing is executed. - Clock control - Set interrupt priority	
Valid state	This function is not usually called from the user directly. This function is callbacked from "R_JCU_Initialize" function. For detail, See2.2.1.	
Description	This function is user defined function. If necessary, add execute process properly.	
Comment		

3.1.2 R_JCU_OnFinalize

Function name	errnum_t R_JCU_OnFinalize(errnum_t e);	
Header	#include "r_jcu_pl.h"	
Parameter	errnum_t e	Error information.
Return value	errnum_t	Error information. The argument is set just as it is.
Description	Finalizes the user defined process. By default, the following processing is executed. - Clock control(stop)	
Valid state	This function is not usually called from the user directly. This function is callbacked from "R_JCU_Finalize" and "R_JCU_FinalizeAsync" functions. For detail, see2.2.2.	
Description	This function is user defined function. If necessary, add execute process properly.	
Comment		

3.1.3 R_JCU_SetDefaultAsync

Function name	void R_JCU_SetDefaultAsync(r_ospl_async_t* const Async, r_ospl_async_type_t AsyncType);	
Header	#include "r_jcu_pl.h"	
Parameter	r_ospl_async_t* const Async	General-purpose argument of OSPL. See. 2.1.4. "NULL" can't be used.
	r_ospl_async_type_t AsyncType	General-purpose argument of OSPL. See. 2.1.4.
Return value	void	
Description	Sets the default value of the variable of r_ospl_async_t type structure.	
Valid state	This function is not usually called from the user directly.	

	This function is callbacked from "R_JCU_TerminateAsync", "R_JCU_StartAsync", and "R_JCU_ContinueAsync" functions.
Description	This function is user defined function. This function is callbacked from asynchronous process of the JCU driver. This function executes the following processing. - Member of variable of r_ospl_async_t type structure corresponds to the "Flags" is set to the default-value, if the 'Flags' member of the variable is zero.
Comment	"ReturnValue" member of "r_ospl_async_t" type is not necessary to set in this function, because "ReturnValue" is initialized in caller asynchronous function,

3.1.4 R_JCU_SetInterruptCallbackCaller

Function name	errnum_t R_JCU_SetInterruptCallbackCaller(const r_ospl_caller_t* const Caller);
Header	#include "r_jcu_pl.h"

Parameter	const r_ospl_caller_t* const Caller	General-purpose argument of OSPL. See. 2.1.4.
Return value	errnum_t	Error information.
	0	No error.

Description	The object which the interrupt callback function is called is registered with driver's user defined functions.
Valid state	This function is not usually called from the user directly. This function is callbacked from "R_JCU_StartAsync" and "R_JCU_ContinueAsync" functions.
Description	This function is user defined function. This function is callbacked from asynchronous process of the JCU driver. This function executes the following processing. - Registers the value of the "Caller" argument of the R_OSPL_CallInterruptCallback function which is callbacked when this function was callbacked from the interrupt handler.
Comment	

3.1.5 R_JCU_OnEnableInterrupt

Function name	void R_JCU_OnEnableInterrupt(jcu_interrupt_lines_t const Enables);
Header	#include "r_jcu_pl.h"

Parameter	jcu_interrupt_lines_t const Enables	The kind of interrupt as the bit flag value.
Return value	void	

Description	It's made interrupt enabled.
Valid state	This function is not usually called from the user directly. This function is callbacked when OSPL does I-LOCK release.
Description	This function is user defined function. This function is callbacked from the processing which the JCU driver enables an interrupt. This function executes the following processing. -Interrupt-service of JCU is enabled to execute.
Comment	

3.1.6 R_JCU_OnDisableInterrupt

Function name	void R_JCU_OnDisableInterrupt(jcu_interrupt_lines_t const Disables1);
Header	#include "r_jcu_pl.h"

Parameter	jcu_interrupt_lines_t const Enables	The kind of interrupt as the bit flag value.
Return value	void	

Description It's made interrupt Disabled.
Valid state This function is not usually called from the user directly.
This function is callbacked when OSPL does I-LOCK set.
Description This function is user defined function.
This function is callbacked from the processing which the JCU driver disables an interrupt.
This function executes the following processing.
-Interrupt-service of JCU is disabled to execute.
Comment

3.1.7 R_JCU_OnInterruptDefault

Function name errnum_t R_JCU_OnInterruptDefault(const r_ospl_interrupt_t* const InterruptSource,
const r_ospl_caller_t* const Caller);
Header #include "r_jcu_pl.h"

Parameter	const r_ospl_interrupt_t* const InterruptSource	Pointer to variable of interrupt source information. see.2.1.4.
	const r_ospl_caller_t* const Caller	General-purpose argument of OSPL. See. 2.1.4.
Return value	void	

Description The interruption callback function of the default.
Valid state This function is not usually called from the user directly.
This function is callbacked from "R_OSPL_CallInterruptCallback" functions, in case of default.
Description This function is user defined function.
This function is callbacked from the interrupt service routine of the JCU driver.
This function executes the following processing.
- Call R_JCU_OnInterrupting function and R_JCU_OnInterrupted function
- Set the event registered with an Async structure.

Comment

3.2 The function of OS Porting Layer(OSPL)

In JCU driver, OS Porting Layer(OSPL) is used. The following table describes OSPL functions used by this driver. For detail, see. "OSPL" Application Note for SH7268/7269.

Table 11 OSPL functions

Function name	Description
R_OSPL_CALLER_Initialize	Call Initialize function.
R_OSPL_THREAD_GetCurrentId	Get running thread ID.
R_OSPL_DisableAllInterrupt	Disables all interrupts.
R_OSPL_EnableAllInterrupt	Releases all disabled interrupts.
R_OSPL_FLAG32_InitConst	Clears all flags in 32bit to 0.
R_OSPL_FLAG32_Set	Set one or some bits to 1.
R_OSPL_FLAG32_Clear	Set one or some bits to 0.
R_OSPL_FLAG32_Get	Get 32bit flags value.
R_OSPL_FLAG32_GetAndClear	Returns the value of flags and clears all bits to 0.
R_OSPL_EVENT_Wait	Waits for setting the flags in 16bit and clear received flags.
R_OSPL_EVENT_Set	Set one or some bits to 1.
R_OSPL_EVENT_Clear	Set one or some bits to 0.
R_OSPL_EVENT_Allocate	Allocate thread attached event
R_OSPL_EVENT_Free	Return thread attached event

3.3 Porting from old version(before Ver0.09)

This driver of the new version(after Ver0.10) changes the basic type name, constant name, variable type, struct name, and function name. When porting from old version, please change each definition with the following table.

If application sources of old version JCU driver were included "typedefine.h" and "jcu_namecnv.h" header file, they are able to use a definition of an old edition.

Table 12 Basic Type Name

new version	old version	define
int8_t	_SBYTE *1	typedef signed char
uint8_t	_UBYTE	typedef unsigned char
int16_t	_SWORD	typedef signed short
uint16_t	_UWORD	typedef unsigned short
int32_t *1	_SINT *1	typedef signed int
uint32_t *1	_UINT *1	typedef unsigned int
int32_t *1	_SDWORD	typedef signed long
uint32_t *1	_UDWORD	typedef unsigned long
char_t	_SBYTE *1	typedef char
bool_t	JCU_Boolean	typedef int
int_fast32_t	_SINT *1	typedef int
uint_fast32_t	_UINT *1	typedef unsigned int

*1 There are more than one types which is corresponded. For example, "_SBYTE" type of the old version corresponds to "int8_t" and "char_t" type of the new version.

Table 13 Constant type, Variable type

new version	old version
jcu_errorcode_t	JCU_ErrorCode
jcu_codec_t	JCU_codec
jcu_continue_type_t	JCU_ContinueType
jcu_detail_error_t	JCU_DetailError
jcu_int_detail_error_t	JCU_IntDetailError
jcu_interrupt_line_t	JCU_InterruptLine
jcu_interrupt_lines_t	JCU_InterruptLines
jcu_swap_t	JCU_Swap
jcu_sub_sampling_t	JCU_SubSampling
jcu_decode_format_t	JCU_DecodeFormat
jcu_jpeg_format_t	JCU_JpegFormat
jcu_huff_t	JCU_HuffType
jcu_table_no_t	JCU_TableNo
jcu_color_element_t	JCU_ColorElement
jcu_status_information_t	JCU_statusInformation
jcu_codec_status_t	JCU_codecStatus

Table 14 Structure Name

new version	old version
jcu_count_mode_param_t	JCU_CountModeParam
jcu_buffer_t	JCU_Buffer

jcu_buffer_param_t	JCU_BufferParam
jcu_decode_param_t	JCU_DecodeParam
jcu_image_info_t	JCU_ImageInfo
jcu_encode_param_t	JCU_EncodeParam
jcu_async_status_t	JCU_AsyncStatus
jcu_internal_information_t	JCU_InternalInformation

Table 15 Function Name

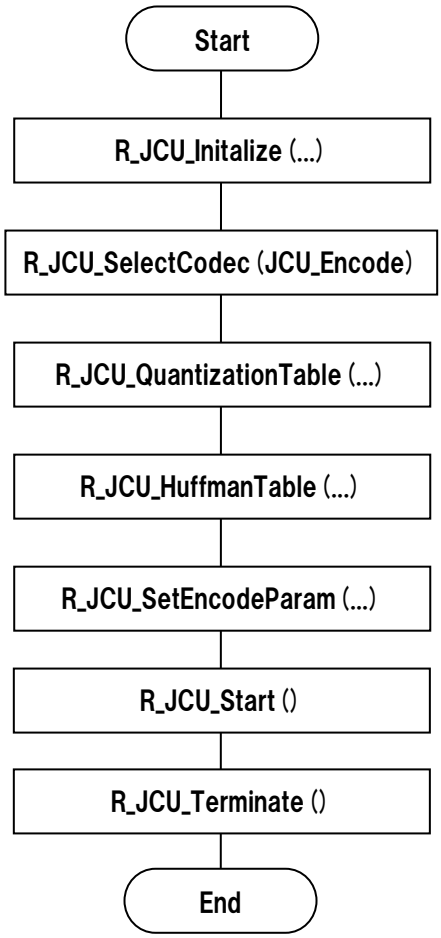
new version	old version
R_JCU_Initialize	JCU_Initialize
R_JCU_Terminate	JCU_Terminate
R_JCU_SelectCodec	JCU_SelectCodec
R_JCU_Start	JCU_Start
R_JCU_SetCountMode	JCU_SetCountMode
R_JCU_Continue	JCU_Continue
R_JCU_SetCallbackFunction*	JCU_SetCallbackFunction
R_JCU_SetDecodeParam	JCU_SetDecodeParam
R_JCU_GetImageInfo	JCU_GetImageInfo
R_JCU_GetErrorInfo	JCU_GetErrorInfo
R_JCU_SetQuantizationTable	JCU_SetQuantizationTable
R_JCU_SetHuffmanTable	JCU_SetHuffmanTable
R_JCU_GetEncodedSize	JCU_GetEncodedSize
R_JCU_SetEncodeParam	JCU_SetEncodeParam
R_JCU_TerminateAsync	JCU_TerminateAsync
R_JCU_GetAsyncStatus	JCU_GetAsyncStatus
R_JCU_StartAsync	JCU_StartAsync
R_JCU_SetPauseForImageInfo	JCU_SetPauseForImageInfo

* In the new version, there are no "JCU_SetCallbackFunction" functions which much the old version. When this function was used, please change the code using flowcharts in sample.

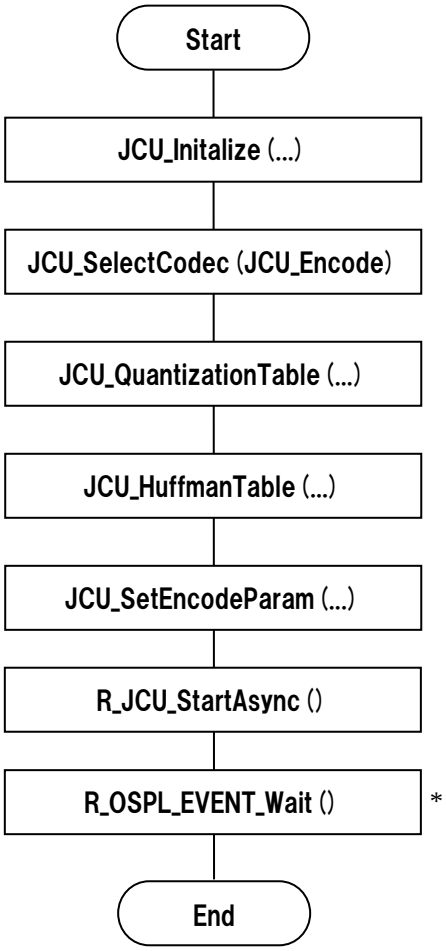
4. Sample

In this section, the flowchart of encode/decode function is illustrated.

4.1 Encode (synchronous process)

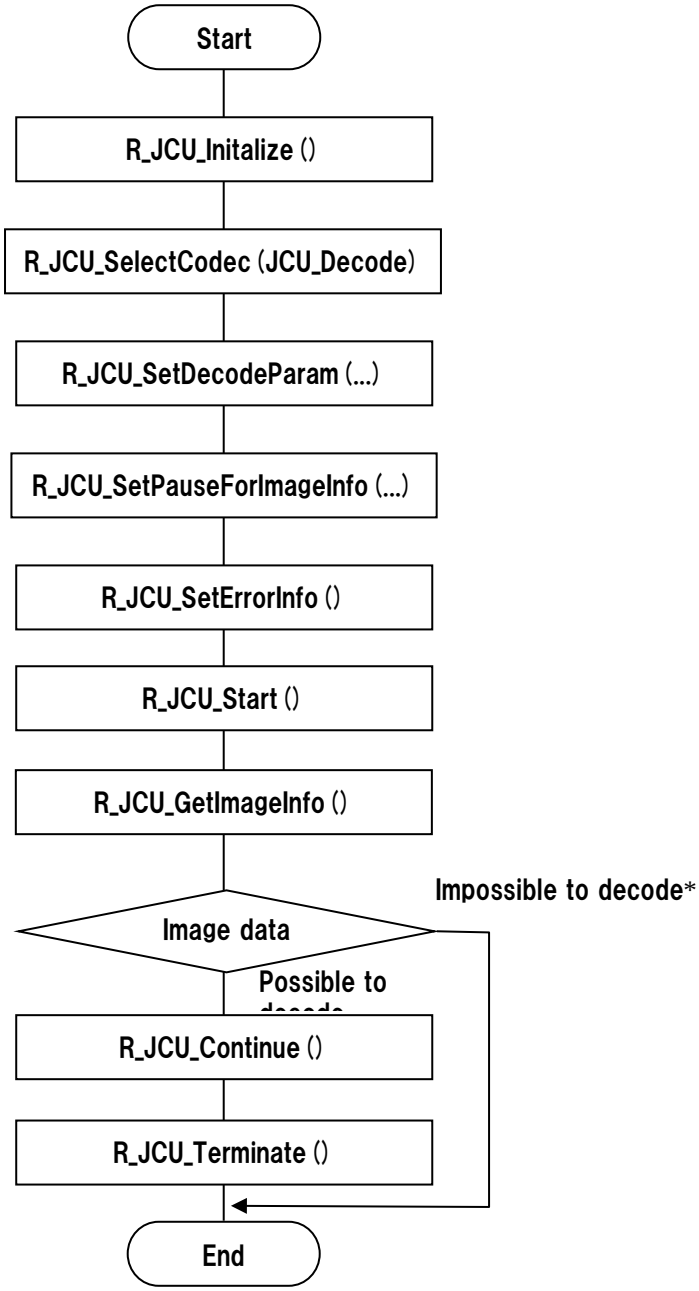


4.2 Encode (asynchronous process)



*This function is able to set time-out time, and this function does polling by no waiting.

4.3 Decode (synchronous process)



* It is impossible to decode, when the pixel format of the decoded JPEG data was not "YCbCr4:2:2" or "YCbCr4:2:0", or the image size (width or height) was zero.

Website and Support

Renesas Electronics website
http://www.renesas.com

Inquiries
http://www.renesas.com/contact/

Revision History

Rev.	date	Description
1.04	Nov.27, 2017	● Only version number was updated due to updating the JCU program. The following items are revision record of the code: ● Correction to change the bit of JCU clock supply in "R_JCU_OnInitialize" and "R_JCU_OnFinalize" function from sampling rate converter (SRC) clock supply.
1.03	Feb.29, 2016	Updated to OSPL version 0.96. Added calling API of allocation and free event flags.
1.00	Oct.9, 2014	1st version.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

(Rev.3.0-1 November 2016)

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0