RZ/A1H Group

JPEG Codec Unit "JCU" Driver Example

R01AN1862EJ0201
Rev.2.01
Feb.2, 2018

Introduction

This application note describes the sample driver which is decoded from the JPEG image data and encoded to the JPEG image data.

The JPEG Codec Unit(JCU) driver example offers the following features:

- The JPEG image data is converted to a raw image data of the RGB565, ARGB8888, and YCbCr422 formats.
- The raw image data of the YCbCr format is converted to a JPEG image data.
- The interrupt information is used for notice of conversion completion.

Target Device

RZ/A1H Group

RZ/A1M Group

RZ/A1LU 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.

Limitations

The count mode (division process) of this JCU driver must not be used. The mode must be conduct an extensive evaluation, if the mode is used.

Table of contents

1. Specifications

Table 1.1 lists the Peripheral Functions and Their Applications, and Figure 1.1 shows the Operation Overview.

Table 1.1 Peripheral Functions to be Used and their Uses

Peripheral functions	Uses
JPEG Codec Unit(JCU)	Converts image data.
Interrupt controller(INTC)	The processor will receive interrupts when decoding or encoding is completed, failed, or paused.
Serial Communication Interface with FIFO(SCIF) Ch2	Output sample code message.

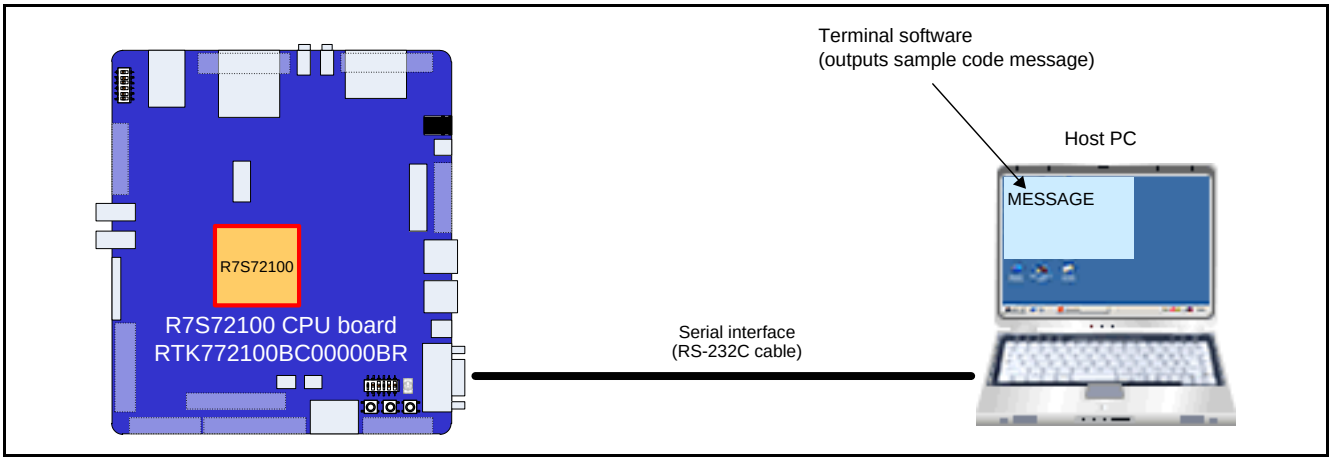

Figure 1.1 **Operation Overview**

2. Operation Check Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Check Conditions

Item		Contents
MCU used		RZ/A1H
Operating frequency*		CPU clock (Iϕ): 400MHz Image processing clock (Gϕ): 266.67MHz Internal bus clock (Bϕ): 133.33MHz Peripheral clock 1 (P1ϕ): 66.67MHz Peripheral clock 0 (P0ϕ): 33.33MHz
Operating voltage		Power supply voltage (I/O): 3.3V Power supply voltage (Internal): 1.18V
ARM	Integrated development environment	ARM® integrated development environment ARM Development Studio 5 (DS-5™) Version 5.16
	C compiler	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
IAR	Integrated development environment	IAR Embedded Workbench for ARM 7.80.4.12495
	C compiler	
Renesas	Integrated development environment	e2 studio (Version: 5.3.0.023)
	C compiler	GNUARM-NONE-EABI v16.01
Operating mode		Boot mode 0 (CS0-space 16-bit booting)
Communication setting of terminal software		• Communication speed: 115200bps • Data length: 8 bits • Parity: None • Stop bit length: 1 bit • Flow control: None
Board used		GENMAI Board • RTK772100BC00000BR (R7S72100 CPU board) • RTK77210000B00000BR (R7S72100 Option board)
Device used		LCD. (Only the sample should be used) • Serial interface (D-sub 9-pin connector)

3. Reference Application Note(s)

For additional information associated with this document, refer to the following application note(s).

RZ/A1H Example of Initialization (R01AN1646EJ)
RZ/A1H Definition of I/O Register file "iodefine.h" (R01AN1860JJ)
RZ/A1H Group OS porting layer "OSPL" Sample Program (R01AN1887EJ)

4. Peripheral Functions

The basic functions of the JCU are described in the RZ/A1H Group User's Manual: Hardware.

5. Description of Hardware

5.1 Hardware Configuration

Figure 5-1 shows examples of hardware devices connected.

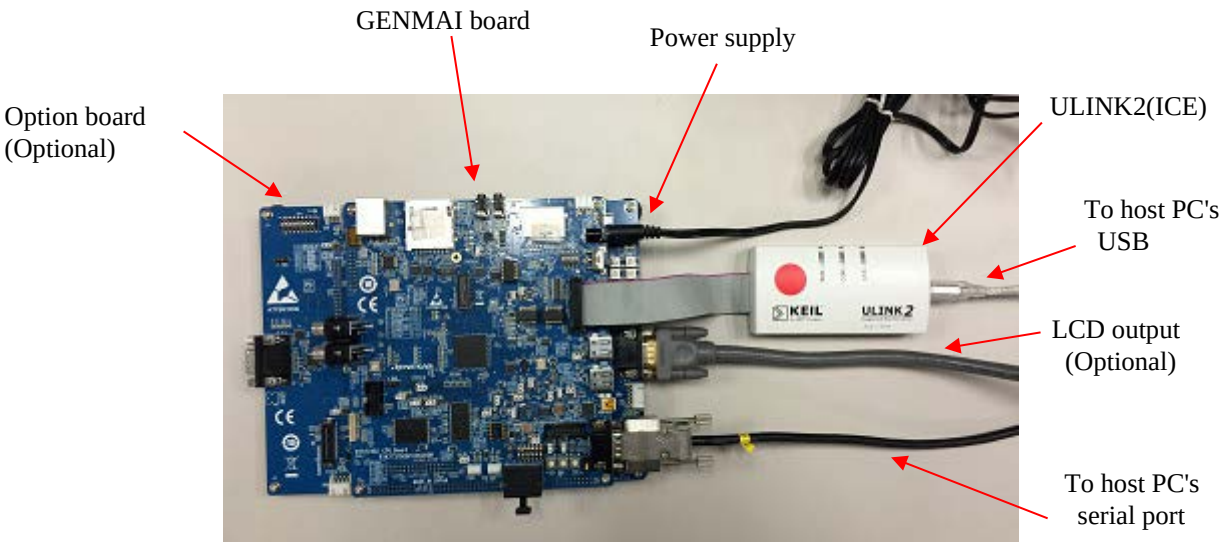

Figure 5-1 Examples of Hardware Devices Connected

5.2 List of Pins to be Used

Table 5-1 lists the pins to be used and their functions.

Table 5-1 Pins to be Used and their Functions

Pin name	I/O	Description
None		

6. Description of Software

6.1 Operation Outline

Figure 6-1 shows the sequence of image data converted using the synchronous function.

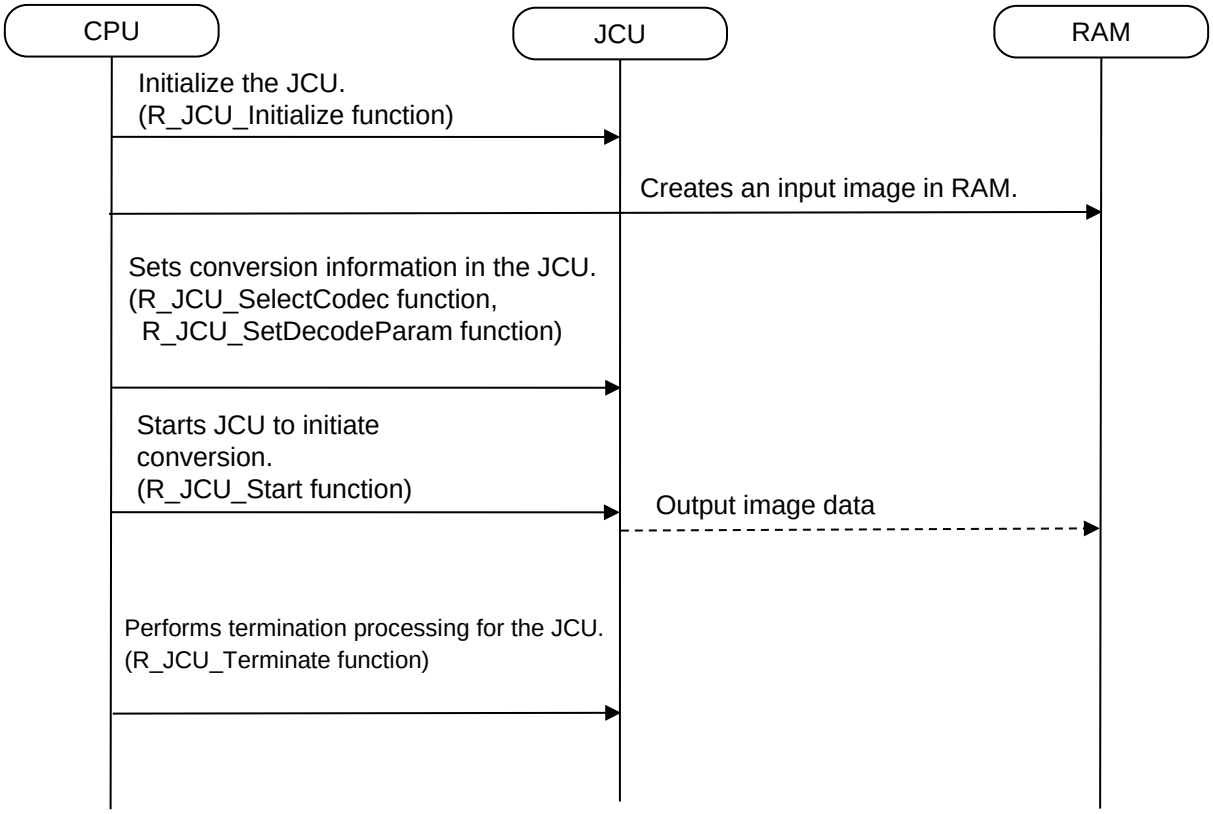

Figure 6-1 Sequence of image data conversion (using the synchronous function).

If you use the asynchronous function, refer to OS transplantation layer OSPL for RZ/A1H group PFV, JCU (R01AN1887EJ).

This sample program has processing of 3 kinds, "decoding processing of a JPEG picture"(R_JCU_SampleDecode function), "decoding and encoding processing of a JPEG picture"(R_JCU_SampleDecodeEncode function) and "the processing indicated after decoding of a JPEG picture"(R_JCU_SampleDecodeAndShow function).

6.1.1 Preparations

The following preparations in Sample Code.

1. Terminal software is started in a host PC and it's established as follows. (In the case of Tera Term)

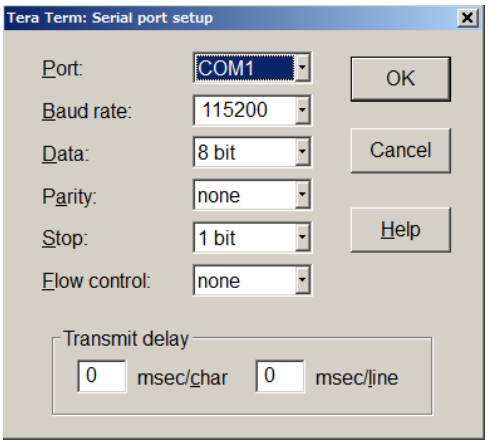

Figure 6.2 Setting of a serial port

2. When a sample program is executed, a message is output at a terminal as follows.

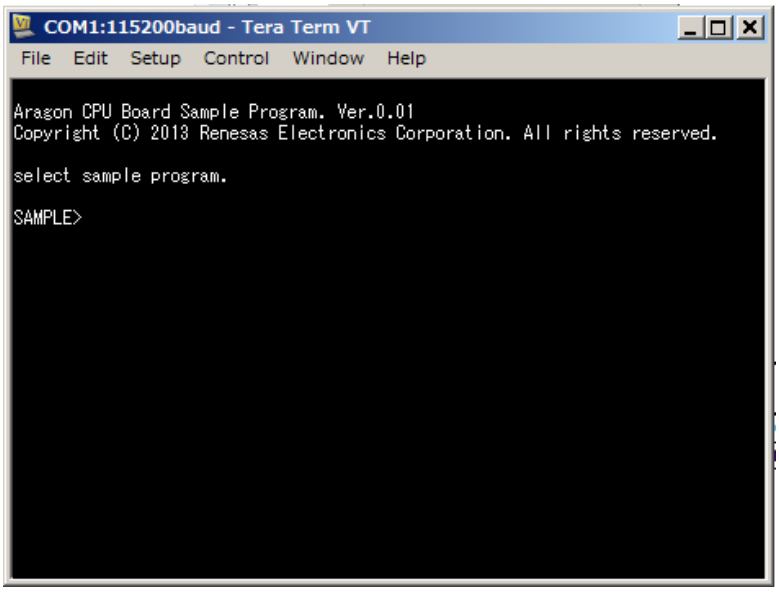

Figure 6.3 Message output at sample program execution

6.2 Memory Mapping

Figure 6.4 shows the Address Space of the RZ/A1H group and the Memory Mapping of the GENMAI Board RTK772100BC00000BR.

In this sample code, the code and data used in the ROM area is located in the NOR flash memory connected to the CS0 space, and the code and data used in the RAM area is located in the large-capacity on-chip RAM.

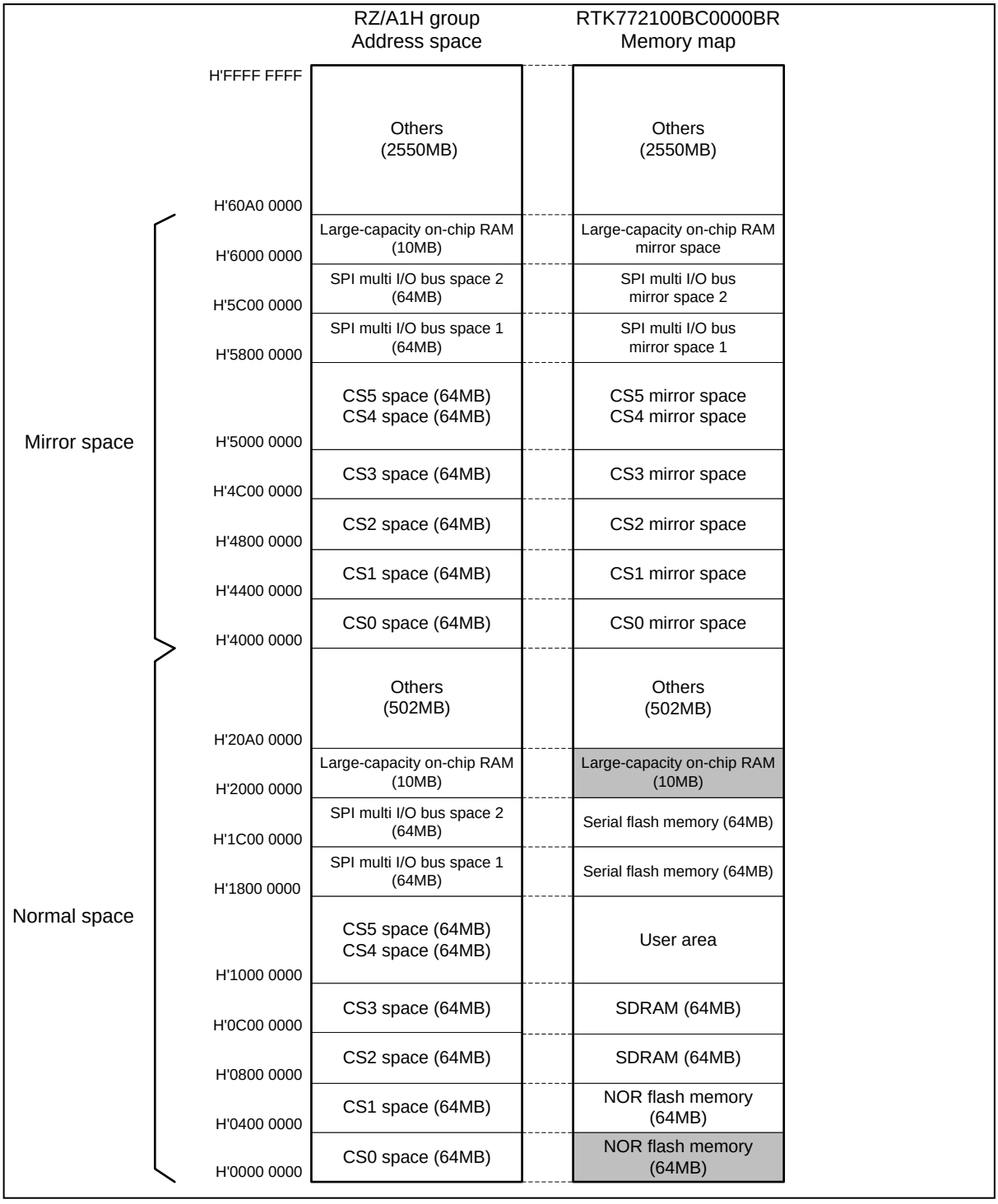

Figure 6.5 Memory Mapping

6.2.1 Section Assignment in Sample Code

In this sample code, the exception processing vector table and the IRQ interrupt handler are assigned to the large-capacity on-chip RAM, and they are executed in such RAM to speed up the interrupt processing. The transfer processing from the NOR flash memory area which is the program code of the exception processing vector table and the IRQ interrupt handler to the large-capacity on-chip RAM area, the clear to zero processing for the data selection without initial data, and the initialization for the data selection with initial data are executed by using the scatter-loading function. Refer to "Image structure and generation" in "ARM Compiler Toolchain Using the Linker" provided by the ARM for more information about the scatter-loading function.

Table 6.1 and Table 6.2 list the Sections to be Used in this sample code. Figure 6.6 shows the Section Assignment for the initial condition of the sample code and the condition after using the scatter-loading function.

Table 6.1 Sections to be Used (1/2)

Area Name	Description	Type	Loading Area	Execution Area
VECTOR_TABLE	Exception processing vector table	Code	FLASH	FLASH
RESET_HANDLER	Program code area of reset handler processing This area consists of the following sections. • INITCA9CACHE (L1 cache setting) • INIT_TTB (MMU setting) • RESET_HANDLER (Reset handler)	Code	FLASH	FLASH
CODE_BASIC_SETUP	Program code area to optimize operating frequency and flash memory	Code	FLASH	FLASH
InRoot	This area consists of the sections located in the root area such as C standard library.	Code and RO Data	FLASH	FLASH
CODE_FPU_INIT	Program code area for NEON and VFP initializations This area consists of the following sections. • CODE_FPU_INIT • FPU_INIT	Code	FLASH	FLASH
CODE_RESET	Program code area for hardware initialization This area consists of the following sections. • CODE_RESET (Startup processing) • INIT_VBAR (Vector base setting)	Code	FLASH	FLASH
CODE_IO_REGRW	Program code area for read/write functions of I/O register	Code	FLASH	FLASH
CODE	Program code area for defaults All the Code type sections which do not define section names with C source are assigned in this area.	Code	FLASH	FLASH
CONST	Constant data area for defaults All the RO Data type sections which do not define section names with C source are assigned in this area.	RO Data	FLASH	FLASH

Table 6.2 Sections to be Used (2/2)

Area Name	Description	Type	Loading Area	Execution Area
VECTOR_MIRROR_TABLE	Exception processing vector table (Section to transfer data to large-capacity on-chip RAM)	Code	FLASH	LRAM
CODE_HANDLER_JMPTBL	Program code area for user-defined functions of IRQ interrupt handler	Code	FLASH	LRAM
CODE_HANDLER	Program code area of IRQ interrupt handler This area consists of the following sections. • CODE_HANDLER • IRQ_FIQ_HANDLER	Code	FLASH	LRAM
DATA_HANDLER_JMPTBL	Registration table data area for user-defined functions of IRQ interrupt handler	RW Data	FLASH	LRAM
ARM_LIB_STACK	Application stack area	ZI Data	-	LRAM
IRQ_STACK	IRQ mode stack area	ZI Data	-	LRAM
FIQ_STACK	FIQ mode stack area	ZI Data	-	LRAM
SVC_STACK	Supervisor (SVC) mode stack area	ZI Data	-	LRAM
ABT_STACK	Abort (ABT) mode stack area	ZI Data	-	LRAM
TTB	MMU translation table area	ZI Data	-	LRAM
ARM_LIB_HEAP	Application heap area	ZI Data	-	LRAM
DATA	Data area with initial value for defaults All the RW Data type sections which do not define section names with C source are assigned in this area.	RW Data	FLASH	LRAM
BSS	Data area without initial value for defaults All the ZI Data type sections which do not define section names with C source area assigned in this area.	ZI Data	-	LRAM

Notes: 1. "FLASH" and "LRAM" shown in Loading Area and Execution Area indicate the NOR flash memory area and the large-capacity on-chip RAM area respectively.

2. Basically, the section name is set to be the same as the region's, however it consists of some sections in the areas of RESET_HANDLER, InRoot, CODE_FPU_INIT, CODE_RESET, CODE, CONST, CODE_HANDLER, DATA, and BSS. Refer to the ARM compiler toolchain manual about the region and the section.

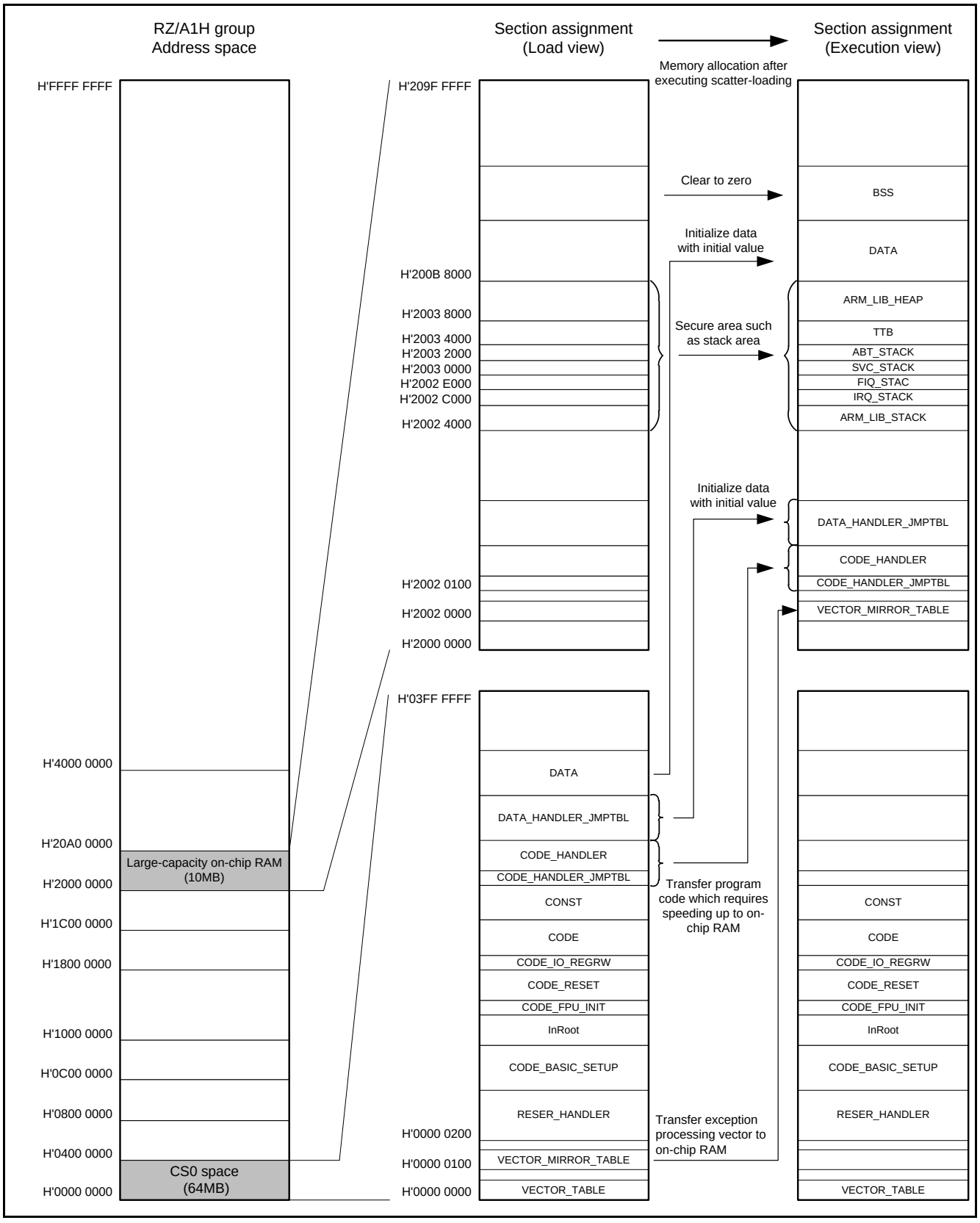

Figure 6.6 Section Assignment

6.2.2 Setting for MMU

The MMU is set to manage the 4 GB area in 1MB unit from the address H'0000 0000 in response to the memory map of the hardware resource used for the GENMAI Board. (Set by the ttb_init.s file.) The minimum unit should be 1MB when customizing the MMU based on the system.

Table 6.3 lists the Setting for MMU.

Table 6.3 Setting for MMU

Definition Name	Contents	Address	Size	Memory Type
M_SIZE_NOR	CS0 and CS1 spaces (NOR flash memory)	H'0000 0000 to H'07FF FFFF	128MB	L1 cache enable, Normal memory
M_SIZE_SDRAM	CS2 and CS3 spaces (SDRAM)	H'0800 0000 to H'0FFF FFFF	128MB	L1 cache enable, Normal memory
M_SIZE_CS45	CS4 and CS5 spaces	H'1000 0000 to H'17FF FFFF	128MB	Strongly-ordered memory (L1 cache disable)
M_SIZE_SPI	SPI multi IO bus space 1 and 2 (serial flash memory)	H'1800 0000 to H'1FFF FFFF	128MB	L1 cache enable, Normal memory
M_SIZE_RAM	Large-capacity on-chip RAM space	H'2000 0000 to H'209F FFFF	10MB	L1 cache enable, Normal memory
M_SIZE_IO_1	On-chip peripheral module and reserved area	H'20A0 0000 to H'3FFF FFFF	502MB	Strongly-ordered memory (L1 cache disable)
M_SIZE_NOR_M	CS0 and CS1 mirror spaces	H'4000 0000 to H'47FF FFFF	128MB	L1 cache disable, Normal memory
M_SIZE_SDRAM_M	CS2 and CS3 mirror spaces	H'4800 0000 to H'4FFF FFFF	128MB	L1 cache disable, Normal memory
M_SIZE_CS45_M	CS4 and CS5 mirror spaces	H'5000 0000 to H'57FF FFFF	128MB	Strongly-ordered memory (L1 cache disable)
M_SIZE_SPI_M	SPI multi IO bus mirror space 1 and 2	H'5800 0000 to H'5FFF FFFF	128MB	L1 cache disable, Normal memory
M_SIZE_RAM_M	Large-capacity on-chip RAM mirror space	H'6000 0000 to H'609F FFFF	10MB	L1 cache disable, Normal memory
M_SIZE_IO_2	On-chip peripheral module and reserved area	H'60A0 0000 to H'FFFF FFFF	2550MB	Strongly-ordered memory (L1 cache disable)

6.2.3 Exception Processing Vector Table

The RZ/A1H has seven types of exception processing (reset, undefined instruction, software interrupt, prefetch abort, data abort, IRQ, and FIQ). In the case of boot mode 0, the exception processing vector table is assigned to the area from H'0000 0000 to the area of 32 bytes (from H'0000 0000 to H'0000 001F) after the reset cancellation.

Figure 6.7 shows the contents of the sample code exception processing vector table as a description example.

```
vector_table
  LDR pc, =reset_handler       ; 0x0000_0000 : Reset exception
  LDR pc, =undefined_handler   ; 0x0000_0004 : Undefined instructions
exception
  LDR pc, =svc_handler         ; 0x0000_0008 : Software interrupts exceptions
  LDR pc, =prefetch_handler    ; 0x0000_000c : Prefetch abort exception
  LDR pc, =abort_handler       ; 0x0000_0010 : Data abort exception
  LDR pc, =reserved_handler    ; 0x0000_0014 : Reserved
  LDR pc, =irq_handler         ; 0x0000_0018 : IRQ exception
  LDR pc, =fiq_handler         ; 0x0000_001c : FIQ exception
```

Figure 6.7 Description Example of Exception Processing Vector Table

6.3 Interrupt

Table 6.4 shows Interrupts using by sample code.

Table 6.4 Interrupts using by sample code

Interrupt (Source ID)	Priority	Summary
JEDI	JCU_INT_PRI(=2)	Compression/Decompression process.
JDTI	JCU_INT_PRI(=2)	Data transfer process

6.4 Required Memory Size

Table 6.4 shows required memory size.

Table 6.5 Required memory size

Memory	Size (bytes)	Note
ROM	11912	JCU - Code
ROM	68	JCU - Read Only Data
ROM, RAM	16	JCU - Read Write Data
RAM	76	JCU - Zero Initialized Data
RAM	1060	Used stack size of the sample program

Note: These required memory sizes vary depending on the C compiler version and the compile option.

6.5 Basic Types

Table 6.6 lists the Basic types using in the example code.

Table 6.6 Basic types using in the example code

Symbol	Description
char_t	8-bit character
bool_t	Logical data type. The value is true (1) or false (0).
int_t	The signed integer for this library is a 32-bit signed integer.
int8_t	8-bit signed integer
int16_t	16-bit signed integer
int32_t	32-bit signed integer
uint8_t	8-bit unsigned integer
uint16_t	16-bit unsigned integer
uint32_t	32-bit unsigned integer
int_fast32_t	Fastest 32-bit minimum-width signed integer
uint_fast8_t	Fastest 8-bit minimum-width unsigned integer
uint_fast16_t	Fastest 16-bit minimum-width unsigned integer
uint_fast32_t	Fastest 32-bit minimum-width unsigned integer
uintptr_t	Same as pointer bit width unsigned integer as physical address
size_t	Same as pointer bit width unsigned integer as byte size
bit_flags_fast32_t	Same as uint_fast32_t bit flags (bit field)
bit_flags32_t	Same as uint32_t bit flags (bit field)

6.6 Constants, Enumerations and Error code

Table 6.7 Constants Used in Sample Code

Section	Symbol	Description
6.6.1	-	Version information.
6.6.2	errnum_t	Error information.
6.6.3	jcu_errorcode_t	Error code. 0 = No error.
6.6.4	jcu_codec_t	Mode selection (Compression or De-compression).
6.6.5	jcu_continue_type_t	Paused factor (continue mode).
6.6.6	jcu_detail_error_t	Error classification of the JCU driver.
6.6.7	jcu_int_detail_error_t	Particular error code.
6.6.8	jcu_int_detail_errors_t	Bitwise OR of the bit flag value jcu_int_detail_error_t
6.6.9	jcu_swap_t	Swap setting.
6.6.10	jcu_sub_sampling_t	Sub sample of the decoded image data.
6.6.11	jcu_decode_format_t	Output pixel format of RAW image data.
6.6.12	jcu_jpeg_format_t	Pixel format of JPEG image data.
6.6.13	jcu_huff_t	Type of Huffman table (AC or DC).
6.6.14	jcu_table_no_t	Quantization table number or Huffman table number.
6.6.15	jcu_status_information_t	Internal state of the JCU driver.
6.6.16	jcu_sub_state_t	Internal sub state of the JCU driver
6.6.17	jcu_sub_status_t	Bitwise OR of the bit flag value jcu_sub_state_t
6.6.18	jcu_codec_status_t	Mode selection information.
6.6.19	jcu_cbcr_offset_t	Cb/Cr range setting. If the pixel format isn't YCbCr, the JCU_CBCR_OFFSET_0 must be used.
6.6.20	jcu_interrupt_line_t	The kind of interrupt as the bit flag value.
6.6.21	jcu_interrupt_lines_t	Bitwise OR of the bit flag value jcu_interrupt_line_t.
6.6.22	-	Others

6.6.1 Version

Table 6.8 Version information

Symbol	Value	Description
JCU_VERSION	200	JCU version number.
JCU_VERSION_STRING	"2.00"	Character string of the JCU version number.

6.6.2 errnum_t

Table 6.9 Error information

Symbol	Value	Description
0	0	No error.
E_OTHERS	1	Other error.
E_FEW_ARRAY	2	The fixed-length arrays size is smaller than data size.
E_FEW_MEMORY	3	Out of memory error occurred when ensure a heap memory.

E_FIFO_OVER	4	Error when enqueue to FIFO is failed.
E_NOT_FOUND_SYMBOL	5	Error of undefined symbol.
E_NO_NEXT	6	There are no next.
E_ACCESS_DENIED	7	Error of Read/Write denied.
E_NOT_IMPLEMENT_YET	9	Not implemented.
E_ERRNO	0x0E (=14)	See "errno".
E_LIMITATION	0x0F (=15)	Temporary limitation.
E_STATE	0x10 (=16)	The error which can't be executed in this state
E_NOT_THREAD	0x11 (=17)	It is not thread. The function cannot be called from interrupt context.
E_PATH_NOT_FOUND	0x12 (=18)	The error by which a file and a folder aren't found.
E_BAD_COMMAND_ID	0x16 (=22)	The command id number is the outside of the range.
E_TIME_OUT	0x17 (=23)	Time out
E_STACK_OVERFLOW	0x1C (=28)	Stack overflow
E_NO_DEBUG_TLS	0x1D (=29)	There is no debug work area.
E_EXIT_TEST	0x1E (=30)	Stop request of the test.

6.6.3 jcu_errorcode_t

Table 6.10 Error code

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.

6.6.4 jcu_codec_t

Table 6.11 Mode selection (Compression or De-compression).

Symbol	Value	Description
JCU_ENCODE	0	Compression process.
JCU_DECODE	1	De-compression process.
JCU_NOT_SELECTED	2	Not selected state

6.6.5 jcu_continue_type_t

Table 6.12 Paused factor (continue mode).

Symbol	Value	Description
JCU_INPUT_BUFFER	0	Resumes reading input image data.
JCU_OUTPUT_BUFFER	1	Resumes writing output image data.
JCU_GET_IMAGE_INFO	2	Clears the process-stopped state caused by requests to read the image information.

6.6.6 jcu_detail_error_t

Table 6.13 Error classification of the JCU driver.

Symbol	Value	Description
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	0x452B	Restart interval data number error detected.
JCU_JCDERR_IMAGE_SIZE	0x452C	Image size error detected.
JCU_JCDERR_LAST_MCU_DATA	0x452D	Last MCU data number error detected.
JCU_JCDERR_BLOCK_DATA	0x452E	Block data number error detected.

6.6.7 jcu_int_detail_error_t

Table 6.14 Particular error code.

Symbol	Value	Description
JCU_INT_ERROR_RESTART_INTERVAL_DATA	0x80	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	0x40	The total number of data in the Huffman-coding segment is not correct in de-compression.
JCU_INT_ERROR_MCU_BLOCK_DATA	0x20	The final number of MCU data in the Huffman-coding segment is not correct in de-compression.

6.6.8 jcu_int_detail_errors_t

Table 6.15 Bitwise OR of the bit flag value jcu_int_detail_error_t

Symbol	Value	Description
JCU_INT_ERROR_ALL	0xE0	All errors

6.6.9 jcu_swap_t

Table 6.16 Swap setting.

Symbol	Value	Description
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.

6.6.10 jcu_sub_sampling_t

Table 6.17 Sub sample of the decoded image data.

Symbol	Value	Description
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.

6.6.11 jcu_decode_format_t

Table 6.18 Output pixel format of RAW image data.

Symbol	Value	Description
JCU_OUTPUT_YCbCr422	0x00	YCbCr422
JCU_OUTPUT_ARGB8888	0x01	ARGB8888
JCU_OUTPUT_RGB565	0x02	RGB565

6.6.12 jcu_jpeg_format_t

Table 6.19 Pixel format of JPEG image data.

Symbol	Value	Description
JCU_JPEG_YCbCr444	0x00	YCbCr444
JCU_JPEG_YCbCr422	0x01	YCbCr422
JCU_JPEG_YCbCr420	0x02	YCbCr420
JCU_JPEG_YCbCr411	0x06	YCbCr411

6.6.13 jcu_huff_t

Table 6.20 Type of Huffman table (AC or DC).

Symbol	Value	Description
JCU_HUFFMAN_AC	0x00	AC
JCU_HUFFMAN_DC	0x01	DC

6.6.14 jcu_table_no_t

Table 6.21 Quantization table number or Huffman table number.

Symbol	Value	Description
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)
JCU_TABLE_NO_3	0x03	Quantization table No. 3 (JCQTBL3)

6.6.15 jcu_status_information_t

Table 6.22 Internal state of the JCU driver.

Symbol	Value	Description
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. Next or current running function is "R_JCU_OnInterrupted".

6.6.16 jcu_sub_state_t

Table 6.23 Sub status of the JCU driver.

Symbol	Value	Description
JCU_SUB_INFOMATION_READY	0x0008	This bit is set to 1, when image size or pixel format can be gotten.
JCU_SUB_DECODE_OUTPUT_PAUSE	0x0100	This bit is set to 1, when number of writing lines were specified value of "jcu_count_mode_param_t::outputBuffer.dataCount " in decoding output image data.
JCU_SUB_DECODE_INPUT_PAUSE	0x0200	This bit is set to 1, when number of reading data were specified value of "jcu_count_mode_param_t::inputBuffer.dataCount" in decoding input encoded data.
JCU_SUB_ENCODE_OUTPUT_PAUSE	0x1000	This bit is set to 1, when number of writing data were specified value of "jcu_count_mode_param_t::outputBuffer.dataCount " in encoding output encoded data.
JCU_SUB_ENCODE_INPUT_PAUSE	0x2000	This bit is set to 1, when number of reading lines were specified value of "jcu_count_mode_param_t::inputBuffer.dataCount" in encoding input image data.

6.6.17 jcu_sub_status_t

Table 6.24 Bitwise OR of the bit flag value jcu_sub_status_t

Symbol	Value	Description
JCU_SUB_PAUSE_ALL	0x3308	all cause regarding pausing

6.6.18 jcu_codec_status_t

Table 6.25 Mode selection information.

Symbol	Value	Description
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.

6.6.19 jcu_cbcr_offset_t

Table 6.26 Cb/Cr range setting.

Symbol	Value	Description
JCU_CBCR_OFFSET_0	0	Range from -128 to 127
JCU_CBCR_OFFSET_128	1	Range from 0 to 255

If the pixel format isn't YCbCr, the JCU_CBCR_OFFSET_0 must be used.

6.6.20 jcu_interrupt_line_t

Table 6.27 The kind of interrupt as the bit flag value.

Symbol	Value	Description
JCU_INTERRUPT_LINE_JEDI	0x00000001u	Interrupt of JEDI.
JCU_INTERRUPT_LINE_JDTI	0x00000002u	Interrupt of JDTI.

6.6.21 jcu_interrupt_lines_t

Table 6.28 Bitwise OR of the bit flag value jcu_interrupt_line_t.

Symbol	Value	Description
JCU_INTERRUPT_LINE_ALL	0x00000003u	Interrupt of both of JEDI and JDTI.

6.6.22 Others

Table 6.29 Others

Symbol	Value	Description
JCU_NUMBER_OF_QUANTIZATION_TABLE_DATA	64	Size of quantization table
JCU_NUMBER_OF_HUFFMAN_TABLE_DATA_DC	28	Size of huffman table DC component
JCU_NUMBER_OF_HUFFMAN_TABLE_DATA_AC	178	Size of huffman table AC component
JCU_MULTI_THREAD	0 or 1	Multi-thread support =1, not support=0

6.7 Structures

Table 6.30 Structures

Section	Symbol	Outline
6.7.1	jcu_count_mode_param_t	Parameters for the count mode (division process).
6.7.2	jcu_buffer_t	Structure for the input/output buffer setting.
6.7.3	jcu_buffer_param_t	Parameters for the input/output buffer setting in de-compression.
6.7.4	jcu_decode_param_t	Parameters for the option setting in de-compression.
6.7.5	jcu_image_info_t	Structure for the image information of the decoded JPEG data.
6.7.6	jcu_encode_param_t	Parameters for the option setting in compression.
6.7.7	jcu_internal_information_t	The internal state of the JCU driver.
6.7.8	jcu_async_status_t	Status of asynchronous operation of the JCU driver
6.7.9	jcu_context_t	Context
6.7.10	jcu_thread_t	Thread information
6.7.11	jcu_config_t	Parameter on initialize and finalize
6.7.12	jcu_contexts_config_t	Parameter of "R_JCU_CONTEXTS_Initialize"
6.7.13	jcu_operator_config_t	Parameter of operation thread

6.7.1 jcu_count_mode_param_t

Outline: Parameters for the count mode (division process).

Header: r_jcu_api.h

Description

Member variable

bool_t inputBuffer.isEnable	false: Disable the division processing on input buffer. true: Enable the division processing on input buffer.
bool_t inputBuffer.isInitAddress	false: When decoding paused, the input address isn't initialized. true: When decoding paused, the input address is initialized by "inputBuffer.restartAddress".
uint32_t* inputBuffer.restartAddress	If "IsInitAddress" is "true", the input data address is initialized by this value.
uint32_t inputBuffer.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.
bool_t outputBuffer.isEnable	false: Disable the division processing on output buffer. true: Enable the division processing on output buffer.
bool_t outputBuffer.isInitAddress	false: When decoding paused, the input address isn't initialized. true: When decoding paused, the output address is initialized by "outputBuffer.restartAddress".
uint32_t* outputBuffer.restartAddress	If "IsInitAddress" is "true", the output data address is initialized by this value.
uint32_t outputBuffer.dataCount	The division size of the output buffer. In the case of decoding mode, when JCU outputs data of "dataCount" line (when data of YCbCr420 was decoded, two times of "dataCount" lines), it pauses. In the case of encoding mode, when JCU outputs data of "dataCount" byte, it pauses.

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

6.7.2 jcu_buffer_t

Outline: Structure for the input/output buffer setting.
Header: r_jcu_api.h
Description:
Member variable:

jcu_swap_t swapSetting	Byte/Word/Longword Swap.
uint32_t* address	Buffer address.

6.7.3 jcu_buffer_param_t

Outline: Parameters for the input/output buffer setting in de-compression.
Header: r_jcu_api.h
Description:
Member variable:

jcu_buffer_t source	Input buffer.
jcu_buffer_t destination	Output buffer.
int16_t lineOffset	Line offset.

6.7.4 jcu_decode_param_t

Outline: Parameters for the option setting in de-compression.
Header: r_jcu_api.h
Description:
Member variable:

jcu_sub_sampling_t verticalSubSampling	Vertical subsampling.
jcu_sub_sampling_t horizontalSubSampling	Horizontal subsampling.
jcu_decode_format_t decodeFormat	The output pixel format of RAW image data.
jcu_cbcr_offset_t outputCbCrOffset	Cb/Cr range setting. If the pixel format isn't YCbCr, the Cb/Cr value must be JCU_CBCR_OFFSET_0.
uint8_t alpha	Alpha value setting. If the pixel format isn't ARGB8888, the alpha value must be zero.

6.7.5 jcu_image_info_t

Outline: Structure for the image information of the decoded JPEG data.
Header: r_jcu_api.h
Description:
Member variable:

uint32_t width	The width of the image data.
uint32_t height	The height of the image data
jcu_jpeg_format_t encodedFormat	The pixel format of original JPEG data.

6.7.6 jcu_encode_param_t

Outline: Parameters for the option setting in compression.
Header: r_jcu_api.h
Description:
Member variable:

jcu_jpeg_format_t encodeFormat	The pixel format of compressed JPEG data. This value must be JCU_JPEG_YCbCr422.
int32_t QuantizationTable[]	Quantization table.
int32_t HuffmanTable[]	Huffman table.
uint32_t DRI_value	DRI (Define Restart Interval) value.
uint32_t width	The width of the input image data.

uint32_t height	The height of the input image data
jcu_cbcr_offset_t inputCbCrOffset	Cb/Cr range setting.

6.7.7 jcu_internal_information_t

Outline: The internal state of the JCU driver.
Header: r_jcu_api.h
Description:
Member variable:

jcu_codec_status_t Codec	Mode selection information.
bool_t isCountMode	false: JCU driver is not count mode. true: JCU driver is count mode.
jcu_int_detail_errors_t ErrorFilter	Filter whether an error is raised, when there is invalid thing.
jcu_async_status_t AsyncStatus	See 6.7.8. jcu_async_status_t.
r_ospl_caller_t InterruptCallbackCaller	Structure managing interrupt callback function.
jcu_i_lock_t* I_Lock	I-lock management structure of OSPL
r_ospl_i_lock_vtable_t* I_LockVTable	V-Table of I-lock of OSPL
bool_t Is_I_LockMaster	Whether this module is master of I-lock
r_ospl_async_t* AsyncForFinalize	Structure of regarding asynchronous finalize.

6.7.8 jcu_async_status_t

Outline: Status of asynchronous operation of the JCU driver
Header: r_jcu_api.h
Description:
Member variable:

jcu_status_information_t Status	Main state
bit_flags_fast32_t SubStatusFlags	Causes of pausing. "jcu_sub_status_t" type.
bool_t IsPaused	Whether JCU is paused.
bool_t IsEnabledInterrupt	Whether all interrupt lines were enabled in related channel
r_ospl_flag32_t InterruptEnables	Enabled interrupt lines in related channel
r_ospl_flag32_t InterruptFlags	Copy of interrupt status register. Symbols related with each bit are defined in the driver. These symbols of some drivers are not public.
r_ospl_flag32_t CancelFlags	Internal flags of status regarding finalize operation from begin to end.

6.7.9 jcu_context_t

However, the number is 1, if thread and interrupt share one channel and do exclusive control.

Context of JCU. It is defined, only if "JCU_MULTI_THREAD = 1".

There is one operator thread, if "JCU_MULTI_THREAD = 1". Only the operator thread accesses JCU registers by notification from the JCU API called from each thread.

Member variables are used internally. Do not access them.

6.7.10 jcu_thread_t

Information of thread managed by JCU driver. The structure is defined, if "JCU_MULTI_THREAD = 1".

Member variables are used internally. Do not access them.

6.7.11 jcu_config_t

Outline	Parameter on initialize and finalize.	
Header	r_jcu_api.h	
Description	This structure is defined, if "JCU_MULTI_THREAD = 1". Usually Flags = 0. To allow to call the JCU API from interrupt, specify "Flags = F_JCU_OwnerThread, OwnerThread = R_OSPL_THREAD_NULL" at the argument of "R_JCU_Initialize" and "R_JCU_TerminateEx" called from thread.	
Member variable	bit_flags_fast32_t Flags	See section 6.11.2, Flagged Structure Parameters. (mandatory) F_JCU_OwnerThread F_JCU_OwnerContext
	r_ospl_thread_id_t OwnerThread	It is usually omitted on initialization. A thread that calls API functions other than "R_JCU_Initialize" and "R_JCU_TerminateEx". R_OSPL_THREAD_NULL = interrupt context. If it was omitted, current thread.
	jcu_context_t* OwnerContext	It is usually omitted on initialization. A using context. If it was omitted and "OwnerThread" member variable was already attached with a context, the argument is the context. If it was omitted and no context was attached, new context will be created internally.

6.7.12 jcu_contexts_config_t

Outline	Parameter of "R_JCU_CONTEXTS_Initialize"	
Header	r_jcu_api.h	
Description	This structure is defined, if "JCU_MULTI_THREAD = 1".	
Member variable	bit_flags_fast32_t Flags	See section 6.11.2, Flagged Structure Parameters. (mandatory) F_JCU_ThreadMemory F_JCU_ThreadMemoryArray
	r_ospl_table_t* ThreadMemory	Array that contains thread information managed by JCU driver. It must be initialized as array index table. The number of elements must be equal or greater than the maximum number of thread that calls the JCU API. The number of threads = (the number of called "R_JCU_Initialize") - (the number of called "R_JCU_Terminate").
	jcu_thread_t* ThreadMemoryArray	First address of array as jcu_thread_t type (6.7.10). No initialization is required. The number of elements must be equal or greater than the number of elements that is contained in array index table that specified at "ThreadMemory" argument.

6.7.13 jcu_operator_config_t

Outline	Parameter of operation thread	
Header	r_jcu_api.h	
Description	This structure is for internal use. This structure is defined, if "JCU_MULTI_THREAD = 1".	
Member variable	bit_flags_fast32_t Flags	See section 6.11.2, Flagged Structure Parameters. (mandatory) F_JCU_ReturnValue F_JCU_MultiThreadQueue

	F_JCU_ReadyOrFinishedEvent
errnum_t ReturnValue	Return value of the operator thread function. This will be set by the operator thread.
r_ospl_queue_id_t MultiThreadQueue	Queue of notification from the thread calling the API to the operator thread.
r_ospl_async_t ReadyOrFinishedEvent	Setting of notification from the operator thread to the thread calling the API.

6.8 List of Variables

Table 6.31 Global Variables

Type	Variable Name	Contents
jcu_internal_information_t	gs_jcu_internal_information	Internal status of the JCU driver.
jcu_operator_config_t	gs_OperatorConfig	Status of the operator thread

6.9 List of Functions

Table 6.32 API functions

Section	Function Name	Outline
6.10.1	R_JCU_Initialize	Initializes the JCU driver.
6.10.2	R_JCU_Terminate	Performs termination processing for the JCU driver (synchronous process).
6.10.3	R_JCU_TerminateAsync	Performs termination processing for the JCU driver (asynchronous process).
6.10.4	R_JCU_TerminateEx	Performs termination processing for the JCU driver (with parameters).
6.10.5	R_JCU_SelectCodec	Sets the JCU mode.
6.10.6	R_JCU_SetCountMode	Sets the count mode.
6.10.7	R_JCU_SetPauseForImageInfo	When the image information can be acquired, it's made the setting which is paused.
6.10.8	R_JCU_SetErrorFilter	The particular error code(jcu_int_detail_error_t) was set to valid.
6.10.9	R_JCU_Start	Starts JCU process (synchronous process).
6.10.10	R_JCU_StartAsync	Starts JCU process (asynchronous process).
6.10.11	R_JCU_Continue	Resume the JCU process (synchronous process).
6.10.12	R_JCU_ContinueAsync	Resume the JCU process (asynchronous process).
6.10.13	R_JCU_SetDecodeParam	Sets decoding parameter.
6.10.14	R_JCU_GetImageInfo	Gets information on the JPEG data.
6.10.15	R_JCU_SetEncodeParam	Sets encoding parameter.
6.10.16	R_JCU_SetQuantizationTable	Sets the Quantization table.
6.10.17	R_JCU_SetHuffmanTable	Sets the Huffman table.
6.10.19	R_JCU_Set2ndCacheAttribute	Sets attribute of L2 cache.
6.10.18	R_JCU_GetEncodedSize	Gets the size of data to be compressed.
6.10.20	R_JCU_GetAsyncStatus	Gets the pointer of a structure that indicates the state of the interrupt and asynchronous process.
6.10.21	R_JCU_OnInterrupting	Interrupt is accepted.
6.10.22	R_JCU_OnInterrupted	Interrupt function is executed.
6.10.23	R_JCU_CONTEXTS_Initialize	Initializes the thread manager
6.10.24	R_JCU_OPERATOR_Thread	The operator thread function
6.10.25	R_JCU_OPERATOR_Dispatch	Execute a part of the operator thread
6.10.26	R_JCU_OPERATOR_FinalizeStart	Requests to finish the operator thread
6.10.27	R_JCU_SetContextInInterrupt	Sets active context for interrupt

Table 6.33 User defined functions

Section	Function Name	Outline
6.10.28	R_JCU_OnInitialize	Initializes the user defined process.
6.10.29	R_JCU_OnFinalize	Finalizes the user defined process.
6.10.30	R_JCU_SetDefaultAsync	Sets the default value of the variable of r_ospl_async_t type structure.
6.10.31	R_JCU_SetInterruptCallbackCaller	The object which the interrupt callback function is called is registered with driver's transplantation layer.
6.10.32	R_JCU_OnEnableInterrupt	Callbacks on request of interrupt enabling.
6.10.33	R_JCU_OnDisableInterrupt	Callbacks on request of interrupt disabling.

6.10.34	R_JCU_OnInterruptDefault	Default interrupt callback function.
6.10.35	R_JCU_CreateOperatorThread	Creates the operator thread
6.10.36	R_JCU_DestroyOperatorThread	Deletes the operator thread
6.10.37	R_JCU_OnAllocateContext	Allocates memory area for context
6.10.38	R_JCU_OnFreeContext	Frees memory area for context

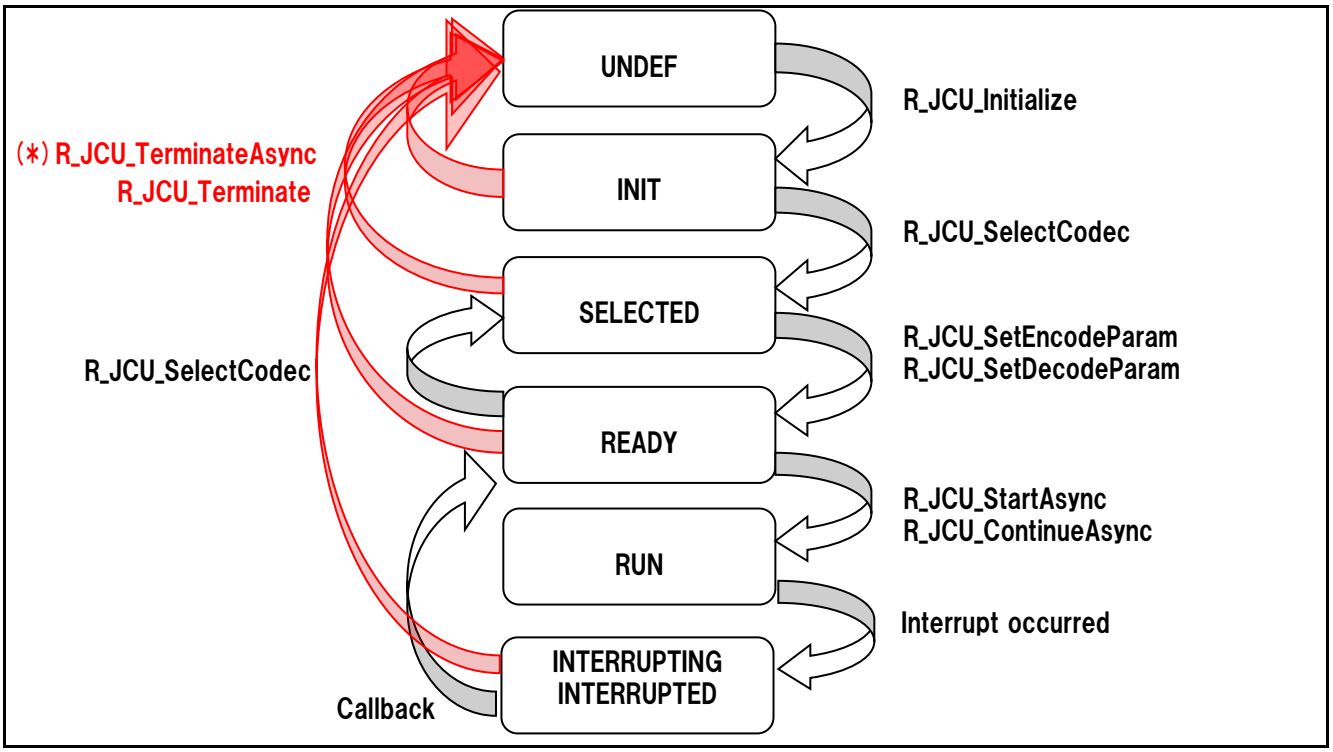

Figure 6.8 State transition diagram.

(*) When executing R_JCU_TerminateAsync in the Run state, the state doesn't transfer. After interrupt occurred (the callback function is executed), the state transfers to "Undef".

Table 6.34 State transition table

API	Status						Transition destination
	Undef	Init	Selected	Ready	Run	Interrupting Interrupted	
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*	OK	Undef
R_JCU_SelectCodec	NG	OK	OK	OK	NG	NG	Selected
R_JCU_SetCountMode	NG	NG	OK	OK	NG	NG	No change
R_JCU_Start	NG	NG	NG	OK	NG	NG	No change
R_JCU_StartAsync	NG	NG	NG	OK	NG	NG	Run
R_JCU_Continue	NG	NG	NG	OK	NG	NG	No change
R_JCU_ContinueAsync	NG	NG	NG	OK	NG	NG	Run
R_JCU_GetAsyncStatus	OK	OK	OK	OK	OK	OK	No change
R_JCU_OnInterrupting	NG	NG	NG	NG	NG	OK	Interrupted
R_JCU_OnInterrupted	NG	NG	NG	NG	NG	OK	Ready or Run
Decompression API							
R_JCU_SetPauseForImageInfo	NG	NG	OK	OK	NG	NG	No change
R_JCU_SetErrorFilter	NG	OK	OK	OK	NG	NG	No change
R_JCU_SetDecodeParam	NG	NG	OK	OK	NG	NG	Ready
R_JCU_GetImageInfo	NG	NG	NG	OK	NG	NG	No change
Compression API							
R_JCU_SetEncodeParam	NG	NG	OK	OK	NG	NG	Ready
R_JCU_SetQuantizationTable	NG	NG	OK	OK	NG	NG	No change
R_JCU_SetHuffmanTable	NG	NG	OK	OK	NG	NG	No change
R_JCU_GetEncodedSize	NG	NG	NG	OK	NG	NG	No change

(*) When executing R_JCU_TerminateAsync in the Run state, the state doesn't transfer. After interrupt occurred (the callback function is executed), the state transfers to "Undef".

6.10 Description of function

The specification of sample code functions is following below:

6.10.1 R_JCU_Initialize

Outline	Initializes the JCU driver.	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_Initialize (jcu_config_t* in_out_Config);	
Description	The state will be in the initialized status. Initializes the internal status(gs_jcu_internal_information). 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.	
Arguments	jcu_config_t* in_out_Config	NULL or parameter (6.7.11)
Return value	Error code.	

6.10.2 R_JCU_Terminate

Outline	Performs termination processing for the JCU driver (synchronous process).	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_Terminate(void);	
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. When "jcu_context_t::StatusOfContext" is "JCU_STATUS_RUN", it waits until processing ends. If the JCU API was called from interrupt, call "R_JCU_TerminateEx" instead of "R_JCU_Terminate".	
Arguments	None	
Return value	Error code.	

6.10.3 R_JCU_TerminateAsync

Outline	Performs termination processing for the JCU driver (asynchronous process).	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_TerminateAsync(r_ospl_async_t* const async);	
Description	The processing which finishes a JCU driver. The function which is executing is suspended before processing ends. For the detail of the argument "async", see the explanation of R_DRIVER_TransferAsync function in OS transplantation layer OSPL for RZ/A1H group PFV,JCU (R01AN1887EJ). About others, see R_JCU_Terminate.	
Arguments	r_ospl_async_t* const async	Synchronization setting.

Return value	Error code.

6.10.4 R_JCU_TerminateEx

Outline	Performs termination processing for the JCU driver (with parameters).	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_TerminateEx(jcu_config_t* in_out_Config);	
Description		
Arguments	jcu_config_t* in_out_Config	NULL or parameter (6.7.11)
Return value	Error code.	

6.10.5 R_JCU_SelectCodec

Outline	Sets the JCU mode.	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_SelectCodec(const jcu_codec_t codec);	
Description	This function selects the JCU mode(Compression or De-compression). All parameters of decode, encode and count mode must be set again. Because when this function was called, these parameters were initialized.	
Arguments	const jcu_codec_t codec	JCU mode(Compression or De-compression)
Return value	Error code.	

6.10.6 R_JCU_SetCountMode

Outline	Sets the count mode.	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_SetCountMode(const jcu_count_mode_param_t *const buffer);	
Description	Sets the count mode(division process). The division processing on input buffer can't be used simultaneously with the division processing on output buffer.	
Arguments	const jcu_count_mode_param_t *const buffer	Count mode
Return value	Error code.	

6.10.7 R_JCU_SetPauseForImageInfo

Outline	When the image information can be acquired, it's made the setting which is paused.	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_SetPauseForImageInfo(const bool_t is_pause)	
Description	When the image information can be acquired, it's made the setting which is paused by the R_JCU_GetImageInfo function.	
Arguments	const bool_t is_pause	TRUE: It's made the setting which is paused. FALSE: It's made the setting which isn't paused.
Return value	Error code.	

6.10.8 R_JCU_SetErrorFilter

Outline	The particular error code(jcu_int_detail_error_t) was set to valid.
Header	r_jcu_api.h

Declaration	jcu_errorcode_t R_JCU_SetErrorFilter(jcu_int_detail_errors_t filter);	
Description	The particular error code was set to valid. When the valid decoding error occurred, interrupt occurs.	
Arguments	jcu_int_detail_errors_t filter	The valid decoding error code(jcu_int_detail_error_t) as the bit flag value.
Return value	Error code.	

6.10.9 R_JCU_Start

Outline	Starts JCU process (synchronous process).	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_Start(void);	
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.	
Arguments	None.	
Return value	Error code.	

6.10.10 R_JCU_StartAsync

Outline	Starts JCU process (asynchronous process).	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_StartAsync(r_ospl_async_t* const async);	
Description	Starts JCU process. The function will return before decoding or encoding ends or pauses. For the detail of the argument "async", see the explanation of R_DRIVER_TransferAsync function in OS transplantation layer OSPL for RZ/A1H group PFV,JCU (R01AN1887EJ). About others, see R_JCU_Start.	
Arguments	r_ospl_async_t* const async	Synchronization setting.
Return value	Error code.	

6.10.11 R_JCU_Continue

Outline	Resumes the JCU process (synchronous process).	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_Continue(const jcu_continue_type_t type);	
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.	
Arguments	const jcu_continue_type_t type	Paused factor(continue mode)
Return value	Error code.	

6.10.12 R_JCU_ContinuetAsync

Outline	Resumes the JCU process (asynchronous process).

Header	r_jcu_api.h
Declaration	jcu_errorcode_t R_JCU_ContinueAsync(const jcu_continue_type_t type, r_ospl_async_t* const async);
Description	Starts JCU process. The function will return before decoding or encoding ends or pauses. About others, see R_JCU_Start.

Arguments		
	jcu_continue_type_t type	Mode of restarting JCU
	r_ospl_async_t* const async	Synchronization setting. NULL is not permitted
Return value	Error code.	

6.10.13 R_JCU_SetDecodeParam

Outline	Sets decoding parameter.
Header	r_jcu_api.h
Declaration	jcu_errorcode_t R_JCU_SetDecodeParam(const jcu_decode_param_t *const decode, const jcu_buffer_param_t *const buffer, const uint32_t interruptKind);
Description	Sets decoding parameter. If the pixel format isn't ARGB8888, the alpha value must be zero. If the pixel format isn't YCbCr, the Cb/Cr value must be JCU_CBCR_OFFSET_0.

Arguments		
	const jcu_decode_param_t *const decode	Pointer to variable of decode parameter information.
	const jcu_buffer_param_t *const buffer	Pointer to variable of buffer.
Return value	Error code.	

6.10.14 R_JCU_GetImageInfo

Outline	Gets information on the JPEG data.
Header	r_jcu_api.h
Declaration	jcu_errorcode_t R_JCU_GetImageInfo(jcu_image_info_t *const buffer);
Description	Gets the image information (width, height, pixel format) of the decoded JPEG data. If data is read before the request which reads the image information, the data is not guaranteed. If the pixel format of the decoded JPEG data is outside of the jcu_jpeg_format_t, it's the error, so JCU can't decode.

Arguments		
	jcu_image_info_t *const buffer	Pointer to variable of image information.
Return value	Error code.	

6.10.15 R_JCU_SetEncodeParam

Outline	Sets encoding parameter.
Header	r_jcu_api.h
Declaration	jcu_errorcode_t R_JCU_SetEncodeParam(const jcu_encode_param_t *const encode, const jcu_buffer_param_t *const buffer, const uint32_t interruptKind);
Description	Sets Encoding parameter.

Arguments		
	const jcu_encode_param_t *const encode	Pointer to variable of encode parameter information.

	const jcu_buffer_param_t *const buffer	Pointer to variable of buffer.
Return value	Error code.	

6.10.16 R_JCU_SetQuantizationTable

Outline	Sets the Quantization table.	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_SetQuantizationTable(const jcu_table_no_t tableNo, const uint8_t *const table);	
Description	Quantization table data. For the setting value of the quantization table data, see "RZ/A1H Group User's Manual: Hardware" section 45.3.1 (4), (a) Quantization Table Specification. Attached "QuantizationTable_Generator.html" file can calculate an example of quantization table.	
Arguments	const jcu_table_no_t tableNo	Quantization table number.
	const uint8_t *const table	Quantization table.
Return value	Error code.	

6.10.17 R_JCU_SetHuffmanTable

Outline	Sets the Huffman table.	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_SetHuffmanTable(const jcu_table_no_t tableNo, const jcu_huff_t type, const uint8_t *const table);	
Description	Huffman table data. For the setting value of the Huffman table data, see "RZ/A1H Group User's Manual: Hardware" section 45.3.1 (4), (b) Huffman Table Specification.	
Arguments	const jcu_table_no_t tableNo	Huffman table number.
	const jcu_huff_t type	Type of Huffman table (AC or DC).
	const uint8_t *const table	Huffman table
Return value	Error code.	

6.10.18 R_JCU_Set2ndCacheAttribute

Outline	Sets attribute of L2 cache.	
Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_Set2ndCacheAttribute(r_ospl_axi_cache_attribute_t const read_cache_attribute, r_ospl_axi_cache_attribute_t const write_cache_attribute);	
Description		
Arguments	r_ospl_axi_cache_attribute_t read_cache_attribute	Signal of ARCACHE[3:0]. See r_ospl_axi_cache_attribute_t.
	r_ospl_axi_cache_attribute_t write_cache_attribute	Signal of AWCACHE[3:0] See r_ospl_axi_cache_attribute_t.
Return value	Error code. No error = 0.	

6.10.19 R_JCU_GetEncodedSize

Outline	Gets the size of data to be compressed.

Header	r_jcu_api.h	
Declaration	jcu_errorcode_t R_JCU_GetEncodedSize(size_t *const out_Size);	
Description	Gets the size of data to be compressed. If data is read before interrupt of encoding complete, the data is not guaranteed.	
Arguments	size_t *const out_Size	Pointer to variable of the data size.
Return value	Error code.	

6.10.20 R_JCU_GetAsyncStatus

Outline	Gets the pointer of a structure that indicates the state of the interrupt and asynchronous process.	
Header	r_jcu_api.h	
Declaration	R_JCU_GetAsyncStatus(const jcu_async_status_t** const out_Status)	
Description	Pointer variable "out_Status" needs the const modifiers.	
Arguments	jcu_async_status_t** out_Status	(Output) Pointer of a structure that indicates the state of the interrupt and asynchronous process.
Return value	Error code. No error = 0.	

6.10.21 R_JCU_OnInterrupting

Outline	The function called when the interrupt is accepted.	
Header	r_jcu_api.h	
Declaration	errnum_t R_JCU_OnInterrupting(const r_ospl_interrupt_t* const InterruptSource);	
Description	This function is usually called automatically from the interrupt callback function of the default. This function sets the value of the interrupt status register to variable "jcu_async_status_t::InterruptFlags". And, Interrupt request is cleared after it. For the detail, see the explanation of R_DRIVER_OnInterrupting function in OS transplantation layer OSPL for RZ/A1H group PFV,JCU (R01AN1887EJ).	
Arguments	r_ospl_interrupt_t* InterruptSource	Interruption sender
Return value	Error code. No error = 0.	

6.10.22 R_JCU_OnInterrupted

Outline	The function called when the interrupt function is executed.	
Header	r_jcu_api.h	
Declaration	errnum_t R_JCU_OnInterrupted(void)	
Description	This function is usually called automatically from the interrupt callback function of the default. Variable "jcu_async_status_t::InterruptFlags" the e function set in 1 is cleared in 0. And, interrupt function is executed. For the detail, see the explanation of R_DRIVER_OnInterrupted function in OS transplantation layer OSPL for RZ/A1H group PFV,JCU (R01AN1887EJ).	
Arguments	None.	
Return value	Error code. No error = 0.	

6.10.23 R_JCU_CONTEXTS_Initialize

Outline	Initializes the thread manager
Header	r_jcu_api.h
Declaration	errnum_t R_JCU_CONTEXTS_Initialize(jcu_contexts_config_t* in_out_Config);

Description	This function initializes a relationship between the thread and the context.	
Arguments	jcu_contexts_config_t* in_out_Config	Parameter (6.7.12)
Return value	Error code. No error = 0.	

6.10.24 R_JCU_OPERATOR_Thread

Outline	The operator thread function	
Header	r_jcu_api.h	
Declaration	void R_JCU_OPERATOR_Thread(void* in_Argument);	
Description		
Arguments	void* in_Argument	Parameter (6.7.13) Value of void* type that is casted from jcu_operator_config_t* type.
Return value	None.	

6.10.25 R_JCU_OPERATOR_Dispatch

Outline	Executes a part of the operator thread	
Header	r_jcu_api.h	
Declaration	errnum_t R_JCU_OPERATOR_Dispatch(r_ospl_thread_id_t in_ThreadID);	
Description	This function is defined, if "R_OSPL_IS_PREEMPTION=0". This function is called from a thread (other than an operator thread) that waits for the operator thread.	
Arguments	r_ospl_thread_id_t in_ThreadID	Thread ID of the operator thread
Return value	Error code. No error = 0.	

6.10.26 R_JCU_OPERATOR_FinalizeStart

Outline	Requests to finish the operator thread	
Header	r_jcu_api.h	
Declaration	errnum_t R_JCU_OPERATOR_FinalizeStart(r_ospl_async_t* ref_Async);	
Description		
Arguments	r_ospl_async_t* ref_Async	Setting of notification when the operator thread is terminated.
Return value	Error code. No error = 0.	

6.10.27 R_JCU_SetContextInInterrupt

Outline	Sets active context for interrupt	
Header	r_jcu_api.h	
Declaration	jcu_context_t* R_JCU_SetContextInInterrupt(jcu_context_t* in_NewCurrentContext);	
Description	Only the function in an interrupt can call this function. Thread must not call this. The driver's API must be called after calling this function. Also, this function must be called at the last of the interrupt function. Its argument should be specified a context that is current context at the start of the interrupt function. The reason for returning the context in this way is to support to multiple interrupts.	
Arguments	jcu_context_t* in_NewCurrentContext	Setting new context
Return value	Old context	

6.10.28 R_JCU_OnInitialize

Outline	Initializes the user defined process.	
Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_OnInitialize (void);	
Description	The user-defined function executed by an initializing process of the JCU driver. If necessary, execute the following processing. - Clock control - Set interrupt priority - Environment-depend process	
Arguments	None.	
Return value	Error code. No error = 0.	

6.10.29 R_JCU_OnFinalize

Outline	Finalizes the user defined process.	
Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_OnFinalize (errnum_t e);	
Description	The user-defined function executed by a finalizing process of the JCU driver. If necessary, execute the following processing. - Clock stops - Clear interrupt priority - Environment-depend process	
Arguments	errnum_t e	Error code. Use it for a return value.
Return value	Error code. No error = 0.	

6.10.30 R_JCU_SetDefaultAsync

Outline	Sets the default value of the variable of r_ospl_async_t type structure.	
Header	r_jcu_pl.h	
Declaration	void R_JCU_SetDefaultAsync(r_ospl_async_t* const ref_Async, r_ospl_async_type_t in_AsyncType)	
Description	The user-defined function executed by all asynchronous process of the JCU driver. Execute 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. ● "ReturnValue" member of the variable is cleared by asynchronous caller's processing.	
Arguments	r_ospl_async_t* ref_Async	Synchronization setting. "NULL" is not permitted.
	r_ospl_async_type_t in_AsyncType	Type of asynchronous operation
Return value	Error code.	

6.10.31 R_JCU_SetInterruptCallbackCaller

Outline	The object which the interrupt callback function is called is registered with driver's transplantation layer.
Header	r_jcu_pl.h
Declaration	errnum_t R_JCU_SetInterruptCallbackCaller(const r_ospl_caller_t* const Caller)
Description	The user-defined function executed by all asynchronous process of the JCU driver. Execute the following processing.

- The setting which calls the R_OSPL_CallInterruptCallback function(value of the argument "Caller" is designated) from the interrupt handler.
For the detail, see the explanation of R_DRIVER_OnInterrupted function in OS transplantation layer OSPL for RZ/A1H group PFV,JCU (R01AN1887EJ).
Driver caller's function manages the body of the structure an argument "Caller" indicates.

Arguments	r_ospl_caller_t* Caller	The value sent to the R_OSPL_CallInterruptCallback function
Return value	Error code. No error = 0.	

6.10.32 R_JCU_OnEnableInterrupt

Outline	Callbacks on request of interrupt enabling.
Header	r_jcu_pl.h
Declaration	void R_JCU_OnEnableInterrupt(jcu_interrupt_lines_t const Enables)
Description	The user-defined function executed by the processing which enable interrupt of the JCU driver. Execute the following processing. -Interrupt-service of JCU is enabled to execute.

Arguments	jcu_interrupt_lines_t const Enables	The kind of interrupt as the bit flag value. The flag is set to "1" for enabled interrupt.
Return value	None.	

6.10.33 R_JCU_OnDiableInterrupt

Outline	Callbacks on request of interrupt disabling.
Header	r_jcu_pl.h
Declaration	void R_JCU_OnDisableInterrupt(jcu_interrupt_lines_t const Disables)
Description	The user-defined function executed by the processing which disable interrupt of the JCU driver. Execute the following processing. -Interrupt-service of JCU is disabled to execute.

Arguments	jcu_interrupt_lines_t const Disables	The kind of interrupt as the bit flag value. The flag is set to "1" for disabled interrupt.
Return value	None.	

6.10.34 R_JCU_OnInterruptDefault

Outline	Default interrupt callback function.
Header	r_jcu_pl.h
Declaration	errnum_t R_JCU_OnInterruptDefault(const r_ospl_interrupt_t* const InterruptSource, const r_ospl_caller_t* const Caller);
Description	This function is user defined function called at the timing of Interrupt handling. This function is registered, if "InterruptCallback" member variable in "Async" argument of "R_JCU_StartAsync" function was not specified.

Arguments	r_ospl_interrupt_t* InterruptSource	Interrupt Source
	r_ospl_caller_t* Caller	Argument passed to "R_OSPL_CallInterruptCallback"
Return value	Error code. No error = 0.	

6.10.35 R_JCU_CreateOperatorThread

Outline	Creates the operator thread.

Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_CreateOperatorThread();	
Description	This function must be defined, if "JCU_MULTI_THREAD=1". "R_JCU_CONTEXTS_Initialize" function must be called before creating new thread. Created thread must call "R_JCU_OPERATOR_Thread" function.	
Arguments	None	
Return value	Error code. No error = 0.	

6.10.36 R_JCU_DestroyOperatorThread

Outline	Deletes the operator thread.	
Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_DestroyOperatorThread();	
Description	This function must be defined, if "JCU_MULTI_THREAD=1". This function requests to finish the operator thread by calling "R_JCU_OPERATOR_FinalizeStart" and wait until the operator thread is destroyed.	
Arguments	None	
Return value	Error code. No error = 0.	

6.10.37 R_JCU_OnAllocateContext

Outline	Allocates memory area for context.	
Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_OnAllocateContext(jcu_context_t** out_AddressOfContext);	
Description	This function must be defined, if "JCU_MULTI_THREAD=1".	
Arguments	jcu_context_t** out_AddressOfContext	Output: allocated address
Return value	Error code. No error = 0.	

6.10.38 R_JCU_OnFreeContext

Outline	Frees memory area for context.	
Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_OnFreeContext(jcu_context_t** in_out_AddressOfContext);	
Description	This function must be defined, if "JCU_MULTI_THREAD=1".	
Arguments	jcu_context_t** in_out_AddressOfContext	Input: Freeing address Output: NULL
Return value	Error code. No error = 0.	

6.11 Supplementary Information

6.11.1 The way to call the API from interrupt, when multi-thread is supported

In the case of "JCU_MULTI_THREAD=1", to call the JCU API from an interrupt other than JCU interrupt, the following code must be written.

- Call "R_JCU_Initialize" from a thread with value to allow calling from an interrupt before an interrupt
- Set a context for an interrupt by calling "R_JCU_SetContextInInterrupt" before calling the JCU API from interrupt
- Set the value of the context prior to the calling current interrupt function
- To terminate the JCU driver, call "R_JCU_TerminateEx" from a thread with value to allow calling from an interrupt

```
void  main()
{
   jcu_config_t  jcu_config_for_interrupt;
   jcu_config_for_interrupt.Flags = F_JCU_OwnerThread;
   jcu_config_for_interrupt.OwnerThread = R_OSPL_THREAD_NULL; /* Interrupt */

   jcu_error = R_JCU_Initialize( &jcu_config_for_interrupt );
         IF( jcu_error != JCU_ERROR_OK ) { e=E_OTHERS; goto fin; }
   thread_config->JCU_Config = &jcu_config_for_interrupt;

   /* Main code */

fin:
   jcu_error = R_JCU_TerminateEx( &jcu_config_for_interrupt );
}

void  OnOtherInterrupt()
{
   jcu_context_t*  this_level_context =
      thread_config->JCU_Config->OwnerContext;
   jcu_context_t*  low_level_context =
      R_JCU_SetContextInInterrupt( this_level_context );

   /* Call JCU API */

fin:
   R_JCU_SetContextInInterrupt( low_level_context );
}
```

6.11.2 Flagged Structure Parameters

Flags member variables in the structure are used as bit flags, and if a bit is 1, the corresponding member variable is enabled according to the coding pattern. If a bit is 0, the value of the member variable is assumed to be the default value. Even if the version is upgraded so that its structure contains additional members, the old and new versions can be binary compatible.

```
FuncA_ConfigClass  config;

config.Flags  = F_FuncA_Param1 | F_FuncA_Param2;
config.Param1 = 10;
config.Param2 = 2;
FuncA( &config );
```

Because there is not Flags |= F_FuncA_Param3, config.Param3 is the default value.

7. Sample Codes

The sample codes can be downloaded from the Renesas Electronics website.

8. Documents for Reference

User's Manual: Hardware

RZ/A1H Group User's Manual: Hardware

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

R7S72100 RTK772100BC00000BR (GENMAI) User's Manual

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

R7S72100 CPU (GENMAI) Optional Board RTK7721000B00000BR 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 Generic Interrupt Controller Architecture Specification Architecture version 1.0

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

ARM Software Development Tools (ARM Compiler toolchain, ARM DS-5 etc.) can be downloaded from the ARM website.

The latest version can be downloaded from the ARM website.

Website and Support

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

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

Revision History

Rev.	Date	Description
2.01	Feb. 2, 2018	Added to support multi-thread Added to support cancel count mode of decoding (split decoding) Example application supports frame buffer on cached area. Updated to OSPL version 1.60. The following history is for changing the code: ● To stop when 2 interrupts at the end of split encoding ● To stop when "GIC_EndInterrupt" was called for RTX ● To stop at the end of decoding (modified from INS6 to DBTF|CBTF) ● Incomplete reset (standby request of JCU) ● Wrong value of "IsPaused" variable ● Few status check, extra error
1.03	Feb. 29, 2016	Updated to example of initialization version 1.01. Updated to OSPL version 0.96. Supported a case of pausing by two or more causes at the same time. Added "R_JCU_Set2ndCacheAttribute". Added a tool which calculate an example of quantization table. Added a sample program of animation by decoding JPEG. Added a sample program of encoding image which is input from video input to JPEG and decoding JPEG.
1.00	Jun.20.2014	First edition issued

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 claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

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

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

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

(Rev.4.0-1 November 2017)

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.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

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: +44-1628-651-700, Fax: +44-1628-651-804

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, 200333 P. R. China
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 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.0