RZ/A2M Group

JPEG Codec Unit "JCU" Driver Example

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.

Target Device

RZ/A2M

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 must not be used. The mode must be conducted an extensive evaluation, if the mode is used.

Contents

1. Specifications

Table 1.1 Peripheral device used lists the Peripheral Functions and Their Applications, and Figure 1.1 Operation check conditions shows the Operation Overview.

Table 1.1 Peripheral device used

Peripheral device	Usage
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.

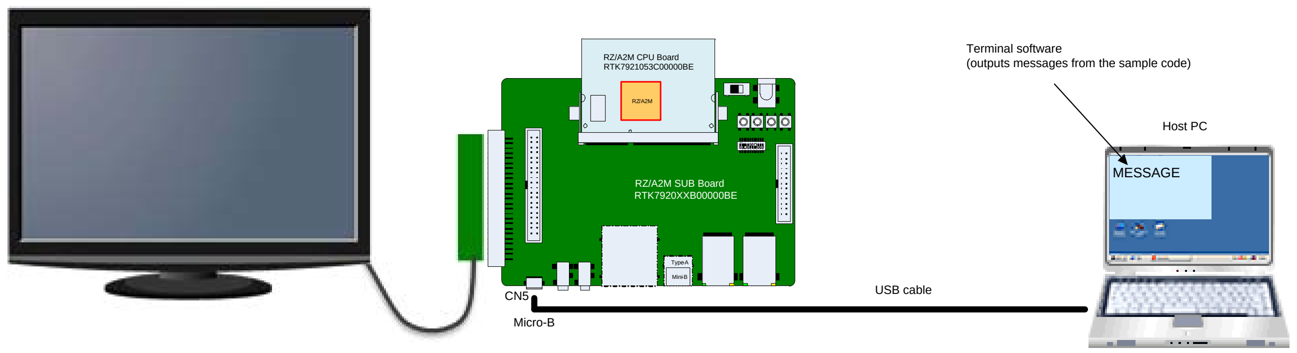


Figure 1.1 Operation check conditions

2. Operation Confirmation Conditions

Table 2.1 Operation Confirmation Conditions(1/2)

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

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

Table 2.2. Operation Confirmation Conditions (2/2)

Operation mode	Boot mode 3 (Serial Flash boot 3.3V)
Terminal software communication settings	• Communication speed: 115200bps • Data length: 8 bits • Parity: None • Stop bits: 1 bit • Flow control: None
Board to be used	RZ/A2M CPU board RTK7921053C00000BE RZ/A2M SUB board RTK79210XXB00000BE
Device (functionality to be used on the board)	• Serial flash memory allocated to SPI multi-I/O bus space (channel 0) Manufacturer : Macronix Inc. Model Name : MX25L51245GXD • RL78/G1C (Convert between USB communication and serial communication to communicate with the host PC.) • LED1

3. Description of Software

3.1 Operation Outline

Figure 3-1 shows the sequence of image data converted using the synchronous function.

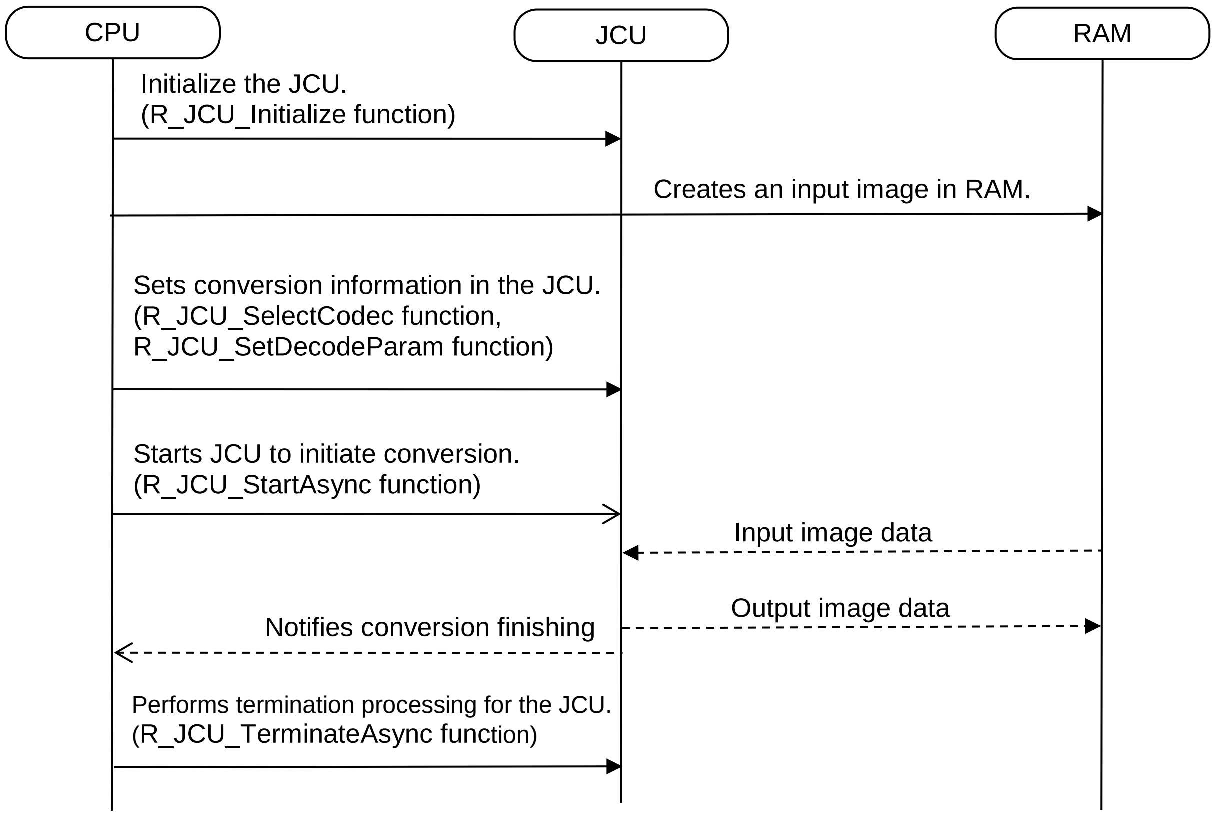

CPU
JCU
RAM
Initialize the JCU.
(R_JCU_Initialize function)
Creates an input image in RAM.
Sets conversion information in the JCU.
(R_JCU_SelectCodec function,
R_JCU_SetDecodeParam function)
Starts JCU to initiate conversion.
(R_JCU_StartAsync function)
Input image data
Output image data
Notifies conversion finishing
Performs termination processing for the JCU.
(R_JCU_TerminateAsync function)

Figure 3-1 Sequence of image data conversion

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).

3.2 Interrupt

Table 3.1 shows Interrupts using by sample code.

Table 3.1 Interrupts using by sample code

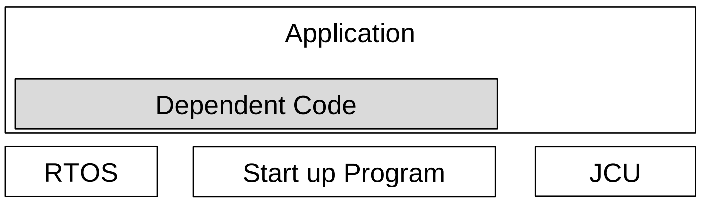
Interrupt (Source ID)	**Priority**	**Summary**
JEDI	JCU_INT_PRIORITY(=30)	Compression/Decompression process.
JDTI	JCU_INT_PRIORITY(=30)	Data transfer process

3.3 Constants, Type, Classes and Functions

Refer the HTML file attached the project.

3.4 Porting Guide

If you changed RTOS or start up program from a JCU running environment, the code in the application that depends on RTOS or start up program must be changed.



Callback function that specified to JCU API function that starts asynchronous operation depends on RTOS. Attached example application do the polling at while statement then CPU load will be 100%.

```
is_event = false;

e= R_JCU_StartAsync( (r_co_function_t) gs_SetTrue,  &is_event );
    if(e){goto fin;}
while ( ! is_event )
    { }  /* Pooling */
e= R_JCU_GetAsyncError(); if(e){goto fin;}
```

In order to avoid the CPU load becoming 100%, the following code must change to use RTOS binary semaphore, event group, thread attached event and so on.

- Specify API function to stop waiting instead of "gs_SetTrue" specified to the argument of asynchronous operating function. This function will be called from interrupt or thread
- Specify synchronizing object of RTOS instead of "&is_event" specified to the argument of asynchronous operating function
- Change polling "while" statement to calling RTOS API to wait
- Specify a different synchronization object for the synchronization object specified for "R_JCU_StartAsync" and the synchronization object for "R_JCU_TerminateAsync". "R_JCU_StartAsync" and "R_JCU_ContinueAsync" can be specified with the same synchronization object.

When calling from middleware, change the following code to use the porting layer of middleware.

- Instead of specifying "gs_SetTrue" as the argument of the asynchronous processing function, specify the function of the porting layer which abstracts to cancel the wait of RTOS. This function is called from an interrupt or thread
- Instead of specifying "&is_event" as an argument of the asynchronous processing function, specify a pointer indicating an instance of the middleware
- Change the "while" statement that does polling to the code of calling the function of porting layer which abstracts to wait in RTOS. Specify a pointer indicating an instance of the middleware to the argument
- Specify a different function for the function specified for "R_JCU_StartAsync" and for the function specified for "R_JCU_TerminateAsync". AAAA and CCCC can be specified with the same function

Mirror area and physical address of RAM depends on the setting of MMU that defined in start up program. The address to store in the pointer is virtual address. The address to access from JCU hardware is physical address. These addresses relationship depends on the setting of MMU.

4. Functions

Table 4.1 API functions

Section	Function Name	Outline
4.1	R_JCU_Initialize	Initializes the JCU driver.
4.2	R_JCU_TerminateAsync	Performs termination processing for the JCU driver (asynchronous process).
4.3	R_JCU_SelectCodec	Sets the JCU mode.
4.4	R_JCU_SetPauseForImageInfo	When the image information can be acquired, it's made the setting which is paused.
4.5	R_JCU_SetErrorFilter	The particular error code(jcu_int_detail_error_t) was set to valid.
4.6	R_JCU_StartAsync	Starts JCU process (asynchronous process).
4.7	R_JCU_GetAsyncError	Returns result of asynchronous process.
4.8	R_JCU_ContinuetAsync	Resume the JCU process (asynchronous process).
4.9	R_JCU_SetDecodeParam	Sets decoding parameter.
4.10	R_JCU_GetImageInfo	Gets information on the JPEG data.
4.11	R_JCU_SetEncodeParam	Sets encoding parameter.
4.12	R_JCU_SetQuantizationTable	Sets the Quantization table.
4.13	R_JCU_SetHuffmanTable	Sets the Huffman table.
4.14	R_JCU_GetEncodedSize	Gets the size of data to be compressed.
4.15	R_JCU_OnInterrupting	Inerrupt service routine (ISR)

Table 4.2 User defined functions

Section	Function Name	Outline
4.16	R_JCU_OnInitialize	Initializes the user defined process.
4.17	R_JCU_OnFinalize	Finalizes the user defined process.
4.18	R_JCU_EnableInterrupt	Callbacks on request of interrupt enabling.
4.19	R_JCU_DisableInterrupt	Callbacks on request of interrupt disabling.

4.1 R_JCU_Initialize

Outline	Initializes the JCU driver.
Header	r_jcu.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
Return value	Error code.	

4.2 R_JCU_TerminateAsync

Outline	Performs termination processing for the JCU driver.
Header	r_jcu.h
Declaration	jcu_errorcode_t R_JCU_TerminateAsync(r_co_function_t in_OnFinalized, volatile void* in_OnFinalizedArgument);
Description	The processing which finishes a JCU driver. This function is asynchronous function that will return before processing ends. The state will be changed. Perform the following processing in the user defined function. - Clock stopped to JCU. - Clear the priority of interrupt. - Sets the environment-depend process. 0 R_JCU_GetAsyncError must be called after finishing this asynchronous operation.

Arguments	r_co_function_t in_OnFinalized	Callback function called when interrupt was signaled. This function will be called from interrupt or thread. If any error was raised, this function will be not called.
	volatile void* in_OnFinalizedArgument	Argument of callback function called when interrupt was signaled
Return value	Error code.	

4.3 R_JCU_SelectCodec

Outline	Sets the JCU mode.
Header	r_jcu.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.	

4.4 R_JCU_SetPauseForImageInfo

Outline	When the image information can be acquired, it's made the setting which is paused.	
Header	r_jcu.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.	

4.5 R_JCU_SetErrorFilter

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

4.6 R_JCU_StartAsync

Outline	Starts JCU process.	
Header	r_jcu.h	
Declaration	jcu_errorcode_t R_JCU_StartAsync(r_co_function_t in_OnFinished, volatile void* in_OnFinishedArgument);	
Description	Starts JCU process. The function is asynchronous function that will return before 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. 0 R_JCU_GetAsyncError must be called after finishing this asynchronous operation.	
Arguments	r_co_function_t in_OnFinalized	Callback function called after interrupt handling. This function will be called from interrupt or thread. If any error was raised, this function will be not called.
	volatile void* in_OnFinalizedArgument	Argument of callback function called after interrupt handling
Return value	Error code.	

4.7 R_JCU_GetAsyncError

Outline	Returns error raised in asynchronized process.	
Header	r_jcu.h	
Declaration	jcu_errorcode_t R_JCU_GetAsyncError(void);	
Description		
Arguments	None	
Return value	Error code.	

4.8 R_JCU_ContinuetAsync

Outline	Resumes the JCU process (asynchronous process).
Header	r_jcu.h
Declaration	jcu_errorcode_t R_JCU_ContinueAsync(const jcu_continue_type_t type, r_co_function_t in_OnFinished, volatile void* in_OnFinishedArgument);
Description	Processing of JCU which paused is resumed. The function is asynchronous function that will return before decoding or encoding ends or pauses. 0 R_JCU_GetAsyncError must be called after finishing this asynchronous operation.

Arguments

jcu_continue_type_t type	Mode of restarting JCU
r_co_function_t in_OnFinalized	Callback function called after interrupt handling. This function will be called from interrupt or thread. If any error was raised, this function will be not called.
volatile void* in_OnFinalizedArgument	Argument of callback function called after interrupt handling

Return value

Error code.

4.9 R_JCU_SetDecodeParam

Outline	Sets decoding parameter.
Header	r_jcu.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.

4.10 R_JCU_GetImageInfo

Outline	Gets information on the JPEG data.
Header	r_jcu.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.

4.11 R_JCU_SetEncodeParam

Outline	Sets encoding parameter.	
Header	r_jcu.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.	

4.12 R_JCU_SetQuantizationTable

Outline	Sets the Quantization table.	
Header	r_jcu.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/A2M 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.	

4.13 R_JCU_SetHuffmanTable

Outline	Sets the Huffman table.	
Header	r_jcu.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/A2M 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.	

4.14 R_JCU_GetEncodedSize

Outline	Gets the size of data to be compressed.	
Header	r_jcu.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.	

4.15 R_JCU_OnInterrupting

Outline	Inerrupt service routine (ISR)	
Header	r_jcu.h	
Declaration	errnum_t R_JCU_OnInterrupting(void);	
Description	All JCU interrupt callback functions registered by R_JCU_OnInitialize function must call this ISR.	
Arguments	None.	
Return value	Error code.	

4.16 R_JCU_OnInitialize

Outline	Initializes the user defined process.	
Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_OnInitialize(void);	
Description	This user-defined function is callbacked from 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.	

4.17 R_JCU_OnFinalize

Outline	Finalizes the user defined process.	
Header	r_jcu_pl.h	
Declaration	errnum_t R_JCU_OnFinalize(void);	
Description	This user-defined function is callbacked from a finalizing process of the JCU driver. If necessary, execute the following processing. - Clock stops - Clear interrupt priority - Environment-depend process	
Arguments	None.	
Return value	Error code.	

4.18 R_JCU_EnableInterrupt

Outline	Callbacks on request of interrupt enabling.	
Header	r_jcu_pl.h	
Declaration	void R_JCU_EnableInterrupt(void);	
Description	This user-defined function is callbacked from the JCU driver.	
Arguments	None.	
Return value	None.	

4.19 R_JCU_DisableInterrupt

Outline	Callbacks on request of interrupt disabling.	
Header	r_jcu_pl.h	
Declaration	bool_t R_JCU_DisableInterrupt(void);	
Description	This user-defined function is callbacked from the JCU driver.	
Arguments	None.	
Return value	Whether all JCU interrupts have been enabled before calling this function.	

5. How to Import the Driver

5.1 e^2 studio

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

5.2 For Projects created outside e^2 studio

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

Generally, there are two steps in any IDE:

1) Copy the driver to the location in the source tree that you require for your project.
2) Add the link to where you copied your driver to the compiler.

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

6. Reference Documents

User's Manual: Hardware

RZ/A2M Group User's Manual: Hardware
The latest version can be downloaded from the Renesas Electronics website.

RTK7921053C00000BE (RZ/A2M CPU board) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

RTK79210XXB00000BE (RZ/A2M SUB board) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
The latest version can be downloaded from the ARM website.

ARM CortexTM-A9 Technical Reference Manual Revision: r4p1
The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification - Architecture version 2.0
The latest version can be downloaded from the ARM website.

ARM CoreLinkTM Level 2 Cache Controller L2C-310 Technical Reference Manual Revision: r3p3
The latest version can be downloaded from the ARM website.

Technical Update/Technical News

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

User's Manual: Development Tools

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

Revision History

Rev.	Date	Description	
		Page	**Summary**
1.10	May. 17, 2019	p4	Table 6.1 Operation Confirmation Conditions(1/2) Remove compiler option "-mthumb-interwork"
		p16	Added “5.How to Import the Driver”
1.03	Apr. 15, 2019	p4	Updated the confirmed version of integrated development environment
1.02	Dec. 28, 2018	-	Modify standby control. Modify to checking STBACK register in R_JCU_OnInitialize function and R_JCU_OnFinalize function.
1.00	Sep. 14, 2018	-	First edition issued

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

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

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Corporate Headquarters

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

Contact information

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

Trademarks