User's Manual

JPEG Encoder

User's Manual

Renesas Micro Computer Middleware

Renesas Electronics
www.renesas.com

Rev.1.00 2013.09

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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

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

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

(2012.4)

Introduction

JPEG Encoder is a software library for compressing images as JPEG files file. It employs sequential DCT-based encoding.

This manual provides the information you will need to create application programs using JPEG Encoder.

1. Reference Documents

The JPEG Encoder software is based on the specifications contained in the standards listed below. Refer to these standards alongside this manual.

■ISO/IEC 10918-1 Information technology - Digital compression and coding of continuous-tone still images

■JIS X 4301-1995 Digital Compression and Coding Of Continuous-tone Still Images

2. Notational Conventions Observed in This Manual

Unless otherwise indicated, the terms and notational conventions described below are used in this manual.

Table 1. Symbols

Notation	Meaning
Numerals	Unless otherwise indicated, decimal notation is used in this manual.

Table 2. Terms

Notation	Meaning
Original image	The original image expressed as a combination of luminance (Y) and chrominance (Cb and Cr) signals
Component	A set of data expressing luminance (Y) or chrominance (Cb or Cr)
JPEG file	An image file in JPEG format
JPEG header	Data contained in an JPEG file, consisting of SOI marker, APPn substring, DQT substring, SOF0 substring, DHT substring, DRI substring, and SOS substring
Image data	The image data after the SOS in a JPEG file
Block	The unit of data that is subject to processing when compressing or decompressing images using the library functions of JPEG Encoder
Marker	A two-byte code marker, where the first byte is hexadecimal FF and the second byte is a value between 1 and hexadecimal FE
Substring	A marker followed by an argument
DCT	Discrete Cosine Transform
Inverse DCT	Inverse Discrete Cosine Transform

Notation	Meaning
DCT coefficients	The coefficients resulting from DCT (no quantization performed)
DCT quantization coefficients	The coefficients resulting from DCT and quantization
MCU	Minimum coded unit The smallest data unit group of coded data
Restart interval	The encoding reinitialization interval Within a scan, the number of MCUs processed as an independent sequence
Restart marker (RSTm)	A marker separating two restart intervals within a scan
Huffman encoding	An entropy encoding method in which a variable-length code is assigned to each input symbol
Huffman decoding	A method of restoring the original value of a Huffman encoded value
Huffman table	A set of variable-length codes required for Huffman encoding and Huffman decoding
Zig-zag sequence	A special sequencing method with DCT coefficients from the lowest to the highest spatial frequencies

3. Structure of This Manual

■Chapter 1, “Overview of JPEG Encoder”
This chapter provides an overview of JPEG Encoder.

■Chapter 2, “JPEG File Compress Library”
This chapter provides a simple explanation of how to use the JPEG File Compress Library to compress images as JPEG files.

■Chapter 3, “JPEG Encode Library”
This chapter describes JPEG Encode Library, which performs basic arithmetic operations required for JPEG image compression, such as quantization and DCT.

Supplementary Description

In addition to this manual, "introduction guides" covering the use of JPEG Encoder are available for each compatible microcontroller. These documents contain information specific to each microcontroller, including hints regarding program ROM and RAM size and processing performance. Refer to them alongside this manual.

Table of Contents

List of Figures

List of Tables

1. Overview of JPEG Encoder

This chapter provides an overview of the features of JPEG Encoder and the procedure for developing software using it.

1.1. JPEG Encoder Features

1.1.1. Algorithm

JPEG Encoder is a software library based on the ISO/IEC 10918-1 and JIS X 4301 standards. This library has the following features.

■Sequential DCT-based encoding algorithm

■Entropy encoding using Huffman code table

■Support for image compression of up to three components with 8-bit precision

1.1.2. Functionality

JPEG Encoder has two components: JPEG File Compress Library, which performs JPEG file compression, and JPEG Encode Library, which performs basic arithmetic operations.

■JPEG File Compress Library
The JPEG File Compress Library provides routines which compress bitmap images into JPEG files. It is used in combination with the JPEG Encode Library. Since the JPEG Encode Library is controlled from the JPEG File Compress Library, users can compress JPEG files using only the functions in the JPEG File Compress Library. See Chapter 2, “JPEG File Compress Library” for details on the JPEG File Compress Library.

■JPEG Encode Library
The JPEG Encode Library mainly provides routines for performing the calculations required when compressing to a JPEG file. These routines accept Y, Cr, and Cb component data and perform DCT transformation, quantization, and Huffman encoding, and output compressed image data (a JPEG file). This library can be used either in conjunction with the JPEG File Compress Library or can be used to implement the image data compression components in an application program. See Chapter 3, “JPEG Encode Library” for details on this library.

A basic diagram is shown in Figure 1.1, “ Compression and Decompression of Image Data ” .

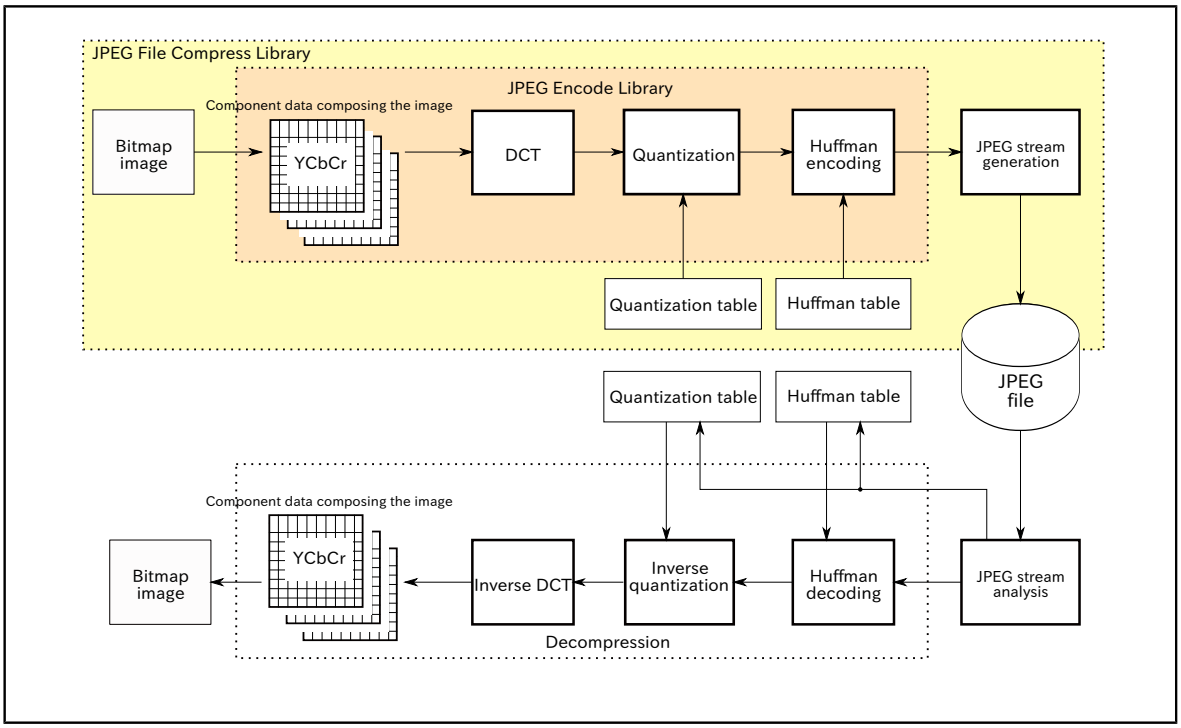

Figure 1.1. Compression and Decompression of Image Data

Shows Table 1.1, “JPEG File Compress Library Specifications” .

Table 1.1. JPEG File Compress Library Specifications

Item		Specification
Output JPEG file	Supported format	JFIF compliant
	Color components	3 components (YCbCr)
	Sampling ratio	4:2:2 (2x1,1x1,1x1) or 4:2:0 (2x2,1x1,1x1)
	Image quality	A value in the range 1 to 128 may be specified.
	Restart marker	Either none or an arbitrary interval may be set.
	Comments	None
	Exif	Not supported
	Progressive	Not supported
	Thumbnail	Not supported
	Output units	Output buffers with an arbitrary size may be provided and the data can be stored to various media in units of that size.
Input image	Image format	RGB565 (16bit color), RGB888 (24bit color), YCbCr 4:2:2
	Input units	Data is read as single unit. (Reading divided into smaller units is not possible.)

JPEG File Compress Library is supplied with source code, so the user can modify its specifications as necessary. To support aspects that are not already supported, add the required functions by modifying the JPEG File Compress Library source code.

For information on the supplied source code, see Section 2.6, “Source Code Information” in Chapter 2, “JPEG File Compress Library” .

1.2. Program Development Procedure

JPEG Encoder is supplied in the form of a library. To use it, link the library to the application program.

Shows Figure 1.2, “Application Program Development Sequence” .

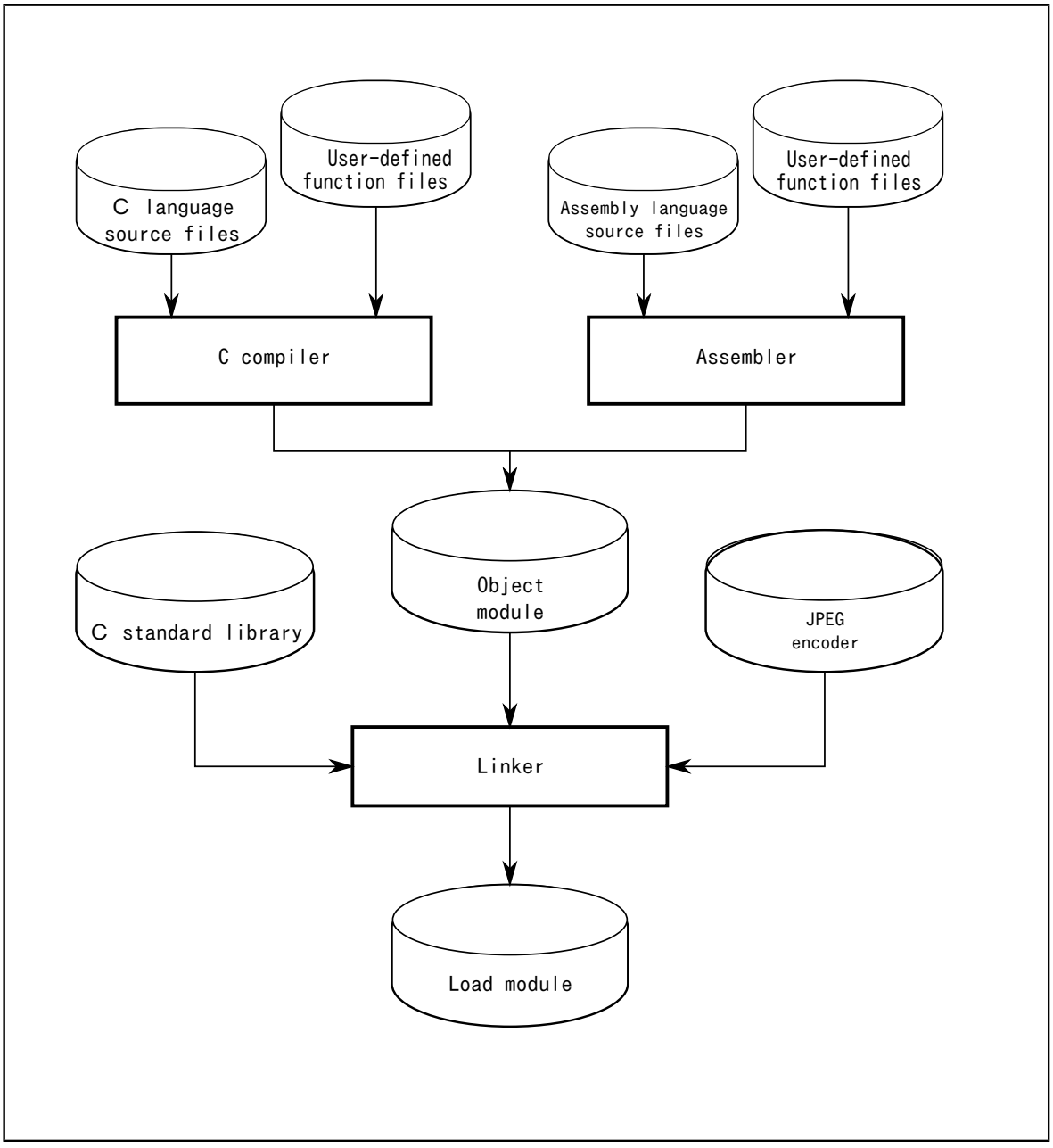

Figure 1.2. Application Program Development Sequence

2. JPEG File Compress Library

This chapter describes the JPEG File Compress Library.

2.1. Overview

The JPEG File Compress Library provides routines which compress bitmap images into JPEG files. It is used in combination with the JPEG Encode Library. Since the JPEG Encode Library is controlled from the JPEG File Compress Library, users can compress JPEG files using only the functions in the JPEG File Compress Library.

Table 2.1, “Library Functions” lists the library functions supported by the JPEG File Compress Library, and Table 2.2, “User-Defined Functions” lists the supported user-defined functions.

Table 2.1. Library Functions

Function	Functionality
R_compress_jpeg	JPEG image compression

Table 2.2. User-Defined Functions

User-Defined Function	Functionality
R_jpeg_write_out_buffer	Output of the output buffer data

2.2. Structure

This is shown in Figure 2.1, “JPEG File Compress Library Structure”

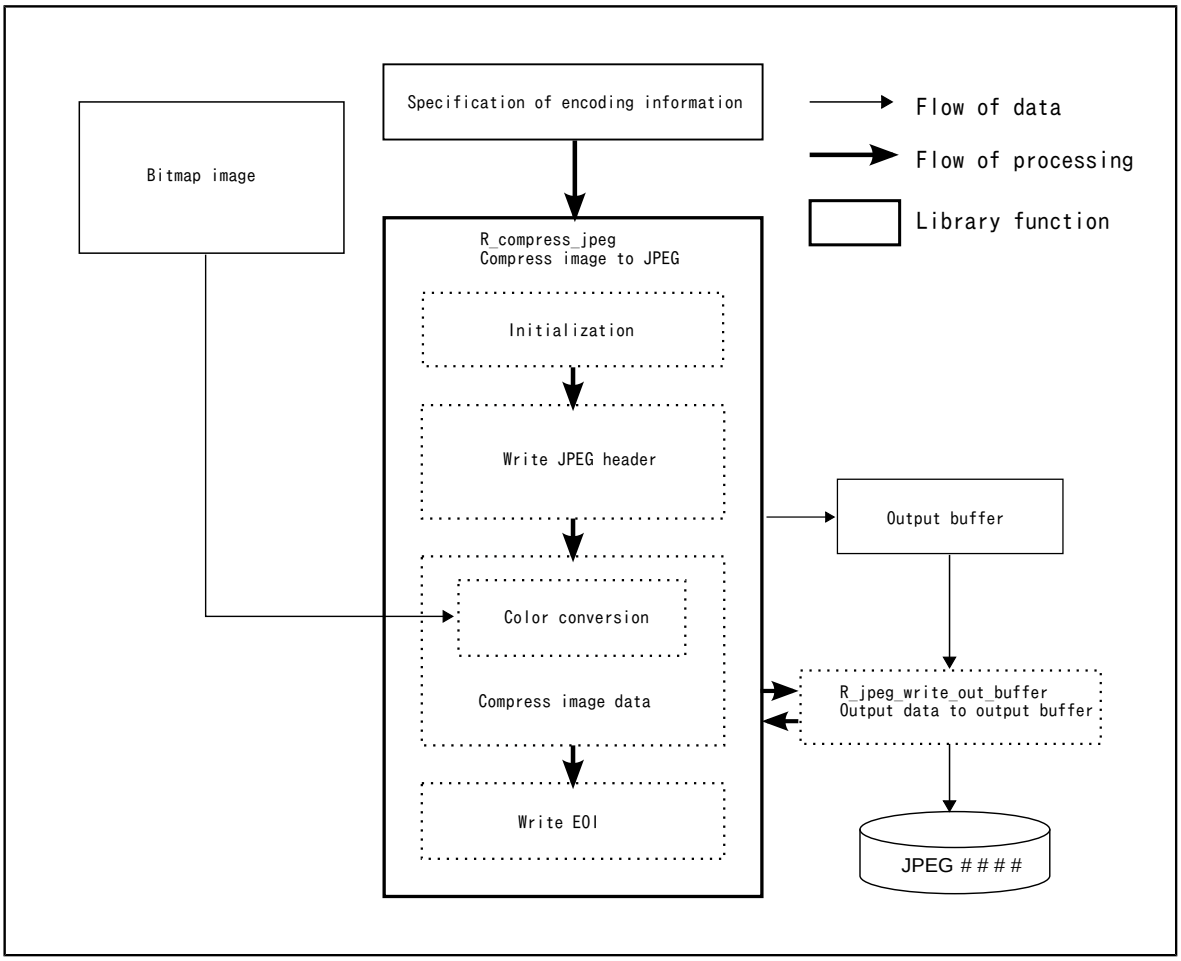

Figure 2.1. JPEG File Compress Library Structure

2.3. Data Configuration

The data configuration defined by the JPEG File Compress Library is described below.

2.3.1. Library Structures

The JPEG File Compress Library uses the library structures defined by the JPEG Encode Library. All variables necessary for JPEG file compression are maintained by the JPEG File Compress Library. See Section 2.6, “Source Code Information” for a list of the maintained variables.

2.3.1.1. JPEG Encoding Information Structure

This structure is used for the various settings required when using the JPEG File Compress Library to compress an image to a JPEG file.

```
struct r_jpeg_encode_t
{
  /* JPEG setting */
  uint16_t width;           /* original image's width */
  uint16_t height;          /* original image's height */
  int16_t quality;          /* image quality when encoding an image (default: 64) */
  int16_t restart_interval; /* RST marker value; when don't use RST marker, set 0 */

  /* input */
  const uint8_t *input_addr;
  uint16_t input_line_byte;
  enum r_jpege_in_format_t input_format;

  /* output */
  uint8_t *outbuff;
  int32_t outbuff_size;
  enum r_jpege_out_format_t output_format;
}
```

Figure 2.2. JPEG Encoding Information Structure r_jpeg_encode_t

The members of structure r_jpeg_encode_t are listed in Table 2.3, "Members of Structure r_jpeg_encode_t".

Table 2.3. Members of Structure r_jpeg_encode_t

Structure Member	Symbol	Functionality
width	X	Width of input image
height	Y	Height of input image
quality	-	Specification of quality as a value in the range 1 to 128
restart_interval	Ri	Encoding reinitialization interval, 0 or user-defined integer
input_addr	-	Start address of input image
input_line_byte	-	The number of bytes per line in the input image area
input_format	-	Format of input image
outbuff	-	Start address of output buffer
outbuff_size	-	Size of output buffer
output_format	-	Output JPEG file format

Table 2.4. Input Image Formats

Definition	Description
JPEGE_IN_RGB565	An RGB565 (16-bit) image is input.

Definition	Description
JPEGE_IN_RGB888	An RGB888 (24-bit) image is input.
JPEGE_IN_YCC422	A YCbCr 4:2:2 image is input. In this case only YCbCr 4:2:2 can be selected as the output.

Table 2.5. Output JPEG File Formats

Definition	Description
JPEGE_OUT_YCC422	The JPEG file is output in YCbCr 4:2:2 format.
JPEGE_OUT_YCC420	The JPEG file is output in YCbCr 4:2:0 format.

2.3.2. Data Input Method

Image data is read in from the address specified with the input_addr member of the r_jpeg_encode structure.

Image data must be provided in a memory space that the CPU can access. Also, the read address cannot be changed during compression processing.

2.3.3. Data Output Method

The JPEG image is output to the area (output buffer) specified with the outbuff member of the r_jpeg_encode structure.

If the output buffer becomes full, the user-defined function R_jpeg_write_out_buffer will be called. In the user-defined function R_jpeg_write_out_buffer , the buffer is cleared by writing the data to the recording media (such as USB memory or an SD card). Compression processing restarts when the user-defined function R_jpeg_write_out_buffer returns. The user-defined function R_jpeg_write_out_buffer is also called when compression completes.

The user-defined function R_jpeg_write_out_buffer will be called multiple times when the JPEG file is larger than outbuff. A single JGEG file is completed by appending the data to the recording media each time it is called.

2.3.4. Macro Definitions

The macro definitions contained in the header file r_compress_jpege.h are listed below.

Table 2.6. Error Code Definitions

Definition	Value	Description
COMPRESS_JPEGE_OK	0	Normal end
COMPRESS_JPEGE_ERROR_ARG	-1	Argument error
COMPRESS_JPEGE_ERROR_WRITE	-2	Write error

2.4. Library Functions

In this section the functions of the JPEG Encode Library are described in detail. The format of the detailed function descriptions is as follows:

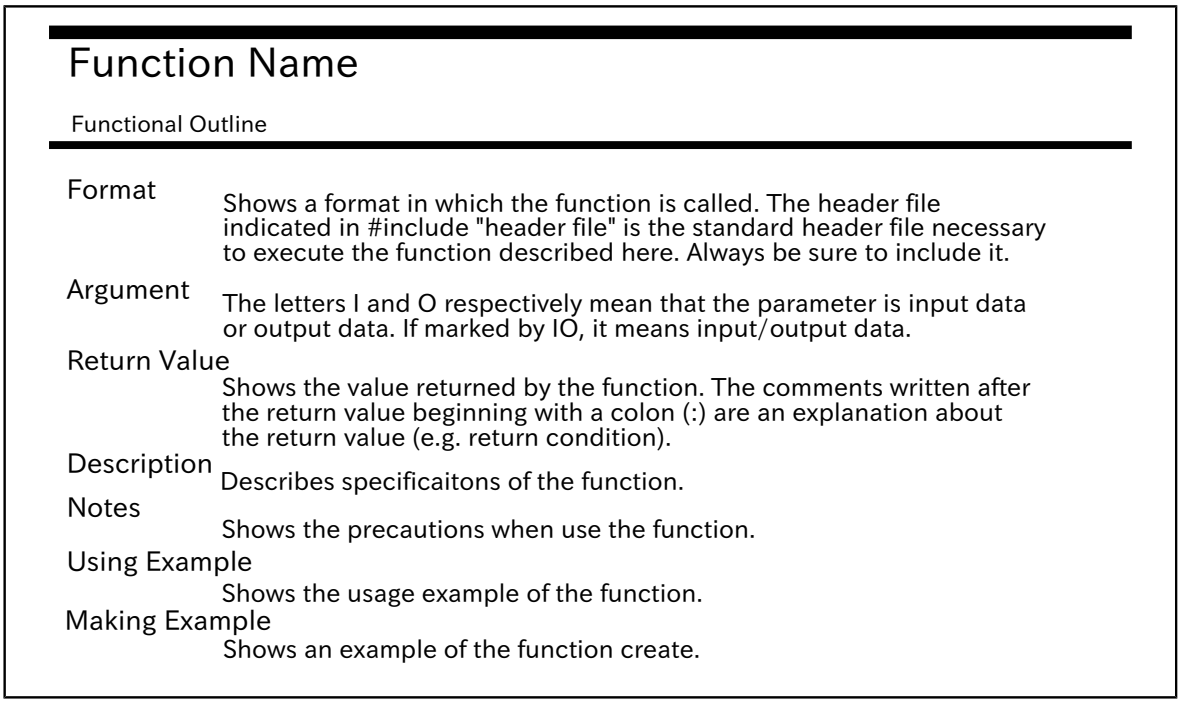

Figure 2.3. Format of Detailed Library Function Descriptions

R_compress_jpeg

Library function

— JPEG image compression

Format

```
#include "r_compress_jpege.h"

int16_t R_compress_jpeg (
                struct r_jpeg_encode_t *setting );
```

Argument

Argument	I/O	Description
setting	I/O	Pointer to JPEG encoding information structure

Return value

Return value	Description
0	Normal end
Other than 0	Error

Description

This function compresses image data into a JPEG file based on the information pointed to by the setting argument. See Table 2.3, “Members of Structure r_jpeg_encode_t” for details on the setting argument.

The width and height members of setting specify the size of the input image. (Only the members of the setting argument are discussed below.)

The quality member specifies the image quality. This is a value from 1 (lowest quality) to 128 (best quality).

The restart_interval member specifies the restart interval. Normally, 0 (no restart marker) is specified. To insert restart markers, specify the interval with which restart markers are inserted as a value of 1 or greater.

The input_addr member specifies the start address of the input image. The input_line_byte member specifies the number of bytes in one line of the memory space in which the input images is located. The input_format member specifies the format of the input image. See Table 2.4, “Input Image Formats” for the image formats that may be input.

For example, if the image has a 320 × 240 RGB888 format, the input_line_byte member will be 320 × 3 = 960 bytes. Also, to compress the center 240 × 160 pixels of a 320 × 240 pixel image, specify the image_addr member to be the address at which the pixel (40, 30) is stored, the width to be 240, the height to be 160, and the input_line_byte to be 960.

The generated JPEG file will output to the output buffer specified with outbuff. The output buffer size is specified with outbuff_size. The user-defined function R_jpeg_write_out_buffer will be called if the output buffer becomes full or JPEG file output completes. See the description of The user-defined function R_jpeg_write_out_buffer for details.

The output_format member specifies the format of the JPEG file. See Table 2.5, “Output JPEG File Formats” , for the values which may be specified.

The following figure shows the structure of the generated JPEG file.

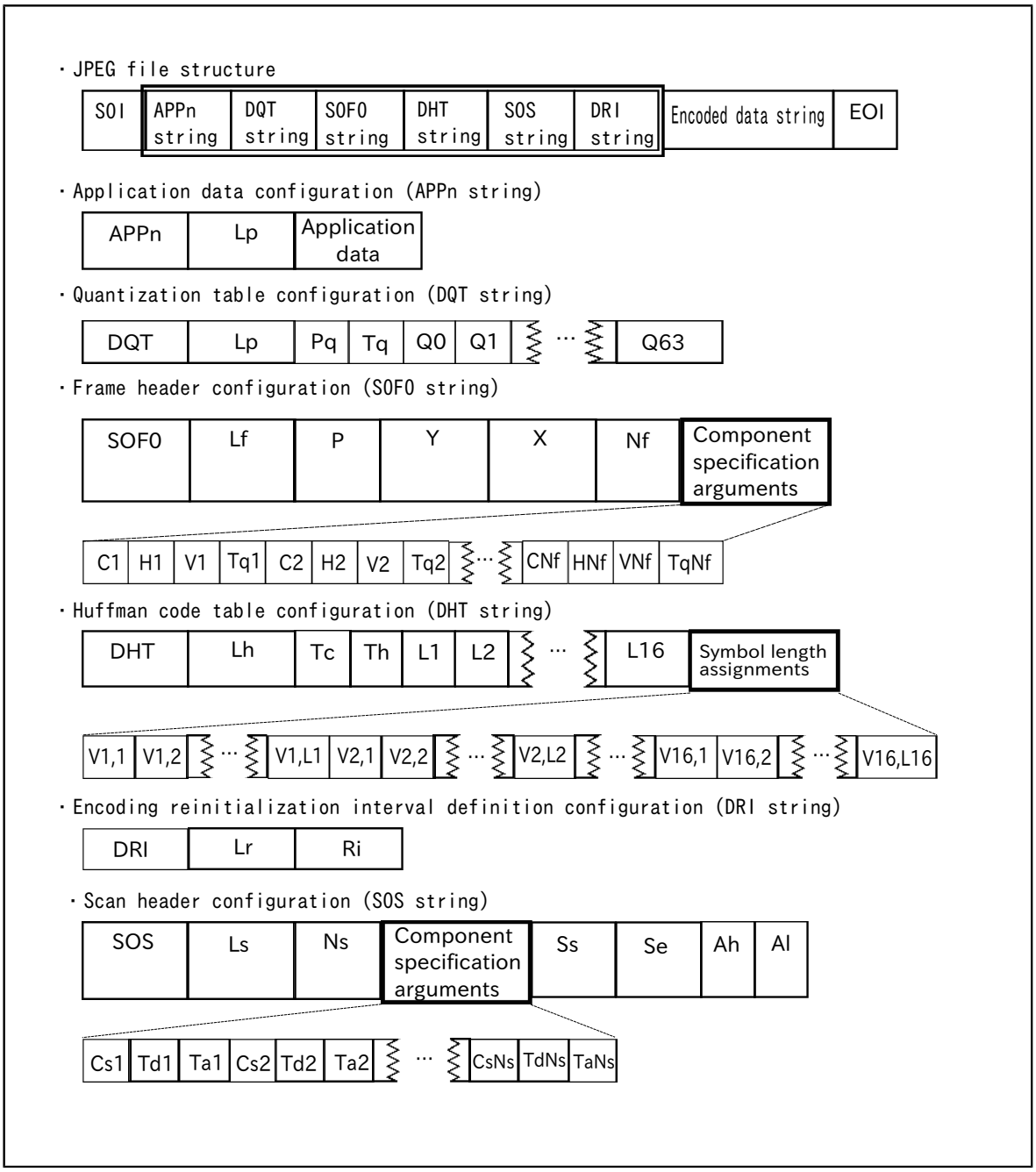

Figure 2.4. JPEG File Structure

2.5. User-Defined Functions

When using the JPEG File Compress Library, the user-defined function R_jpeg_write_out_buffer is required.

R_jpeg_write_out_buffer

User-defined function

— Output of the output buffer data

Format

```
#include "r_jpeg.h"

uint8_t * R_jpeg_write_out_buffer (
                int32_t *rest );
```

Argument

Argument	I/O	Description
rest	I/O	Remaining data count in output buffer

Return value

Returns the start address of the output buffer.

Description

This function is called when the output buffer is full or JPEG file output has completed. If the generated JPEG file is larger than the output buffer, it will be called multiple times.

The contents of the output buffer should be written to the storage media used by the application. The number of bytes written will be the size of the output buffer minus the value pointed to by the argument rest.

After the write, the output buffer is initialized. The start address of the output buffer should be returned as the return value from this function. Also, the value pointed to by the argument rest should be reinitialized to the size of the output buffer.

Using example

```
#include ”r_compress_jpege.h”

#define JPEG_BUFF_SIZE 10240
uint8_t output_jpeg_file[JPEG_BUFF_SIZE];

uint8_t * R_jpeg_write_out_buffer(int32_t *rest)
{
  int32_t size;
  size = JPEG_BUFF_SIZE - *rest;

  // R_tfat_f_write(&fp, (void*)output_jpeg_file, size, &file_rw_cnt);

  _jpeg_file_size += size;
  *rest = JPEG_BUFF_SIZE;

  return output_jpeg_file;
}
```

2.6. Source Code Information

Information about the source code of the JPEG File Compress Library is provided below.

Table 2.7. JPEG File Compress Library List of Files

File Name	Functionality
r_compress_jpege.c	Main compression processing routine
r_write_headers.c	Writes the JPEG header information
r_rgb2ycc.c	Converts the color space

Table 2.8. JPEG File Compress Library List of Functions

Function	Functionality
R_compress_jpege	Compression to JPEG file
_jpeg_init	Initial settings
encode_jpeg422	Main processing routine for compression in YCbCr 4:2:2 format
encode_jpeg420	Main processing routine for compression in YCbCr 4:2:0 format
R_jpeg_dump_buffer	User-defined function for JPEG Encode Library
readSamplingMCU_RGB888 _YCbCr422	Converts the input image (RGB888 format) to YCbCr 4:2:2 format and reads it
readSamplingMCU_RGB888 _YCbCr420	Converts the input image (RGB888 format) to YCbCr 4:2:0 format and reads it
readSamplingMCU_RGB565 _YCbCr422	Converts the input image (RGB565 format) to YCbCr 4:2:2 format and reads it
readSamplingMCU_RGB565 _YCbCr420	Converts the input image (RGB565 format) to YCbCr 4:2:0 format and reads it
readSamplingMCU_YCbCr422	Reads the input image (YCbCr 4:2:2 format) without conversion
_jpeg_write_header	Writes the JPEG header information
_jpeg_writeSOI	Writes the SOI
_jpeg_writeSOF0	Writes the SOF0
_jpeg_writeSOS	Writes the SOS
_jpeg_writeDQT	Writes the DQT
_jpeg_writeDHT	Writes the DHT
_jpeg_writeAPP	Writes the APP
r_rgb2ycc	Converts one pixel of data from RGB888 format to YCbCr format

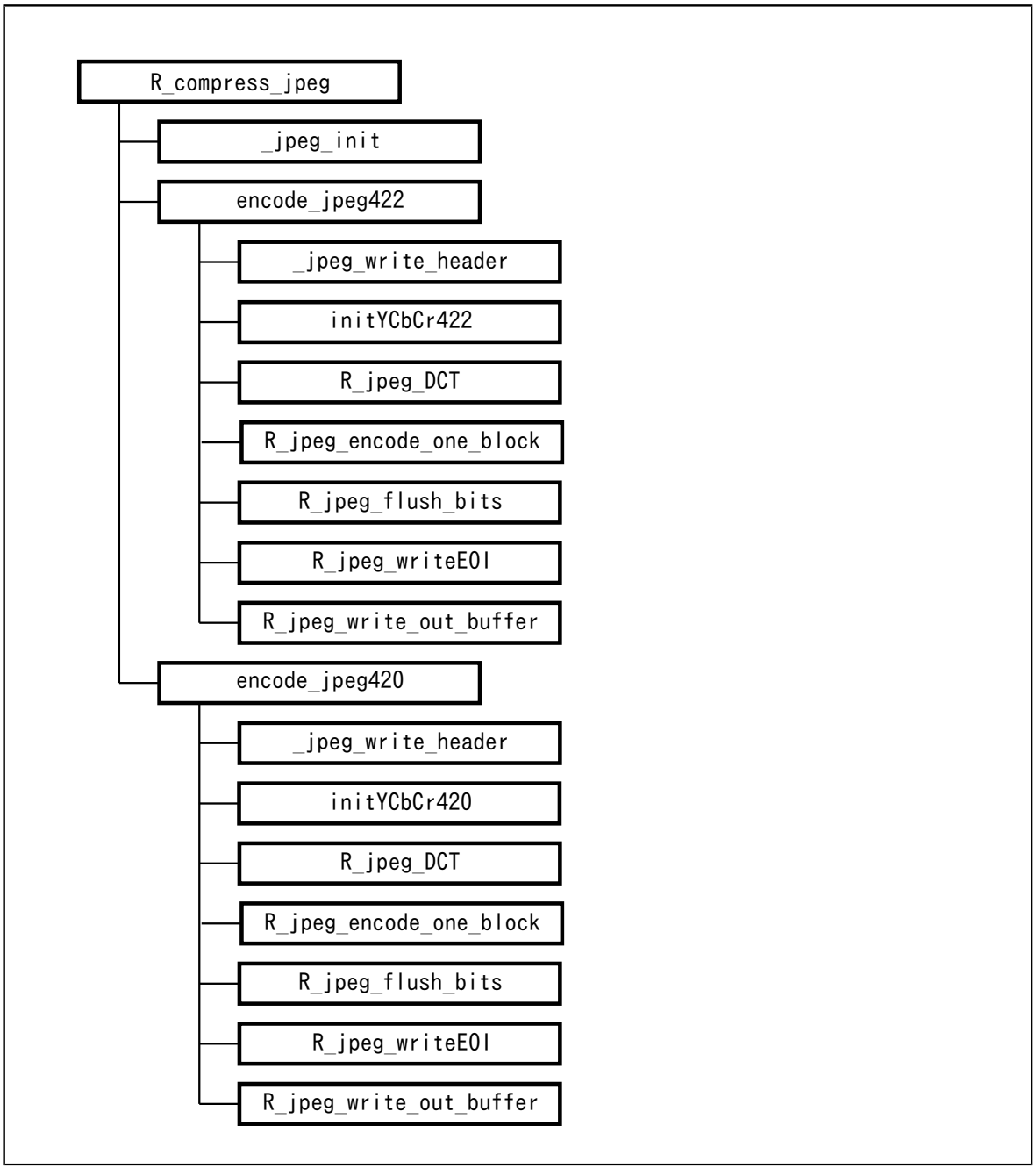

Figure 2.5. Tree Structure of Functions

Table 2.9. Maintained Variables

Variable	Application
working	JPEG Encode Library structure _jpeg_working
encFMB	JPEG Encode Library structure _jpeg_enc_FMB
encSMB	JPEG Encode Library structure _jpeg_enc_SMB
MCU_data	DTC work area (8 × 8 + 2 words)
Y_buf[]	Y component data ((8 × 8) × 4 bytes) for 1 MCU
Cb_buf[]	Cb component data (8 × 8 bytes) for 1 MCU

Variable	Application
Cr_buf[]	Cr component data (8 × 8 bytes) for 1 MCU (8x8 byte)

3. JPEG Encode Library

This chapter describes the JPEG Encode Library, which performs basic arithmetic operations.

3.1. Overview

The JPEG Encode Library performs Huffman encoding, quantization, and DCT required to compress JPEG files.

Table 3.1, “Library Functions” lists the library functions supported by the JPEG Encode Library, and Table 3.2, “User-Defined Functions” lists the supported user-defined functions.

Table 3.1. Library Functions

Function	Functionality
R_jpeg_add_quant_table	Registers quantization table
R_jpeg_DCT	Executes DCT and quantization
R_jpeg_encode_one_block	Executes Huffman encoding
R_jpeg_writeDRI	Writes DRI marker
R_jpeg_writeRST	Writes RSTm marker
R_jpeg_writeEOI	Writes EOI marker
R_jpeg_flush_bits	Forcibly writes Huffman encoded data

Table 3.2. User-Defined Functions

User-Defined Function*	Functionality
R_jpeg_write_out_buffer	Outputs the JPEG file

* The names of user-defined functions may be changed to any name you wish.

3.2. Structure

Figure 3.1, “JPEG Encode Library Structure” shows and example of generating a JPEG file using the JPEG Encode Library for an 8×8 pixel block of Y, Cb, and Cr components.

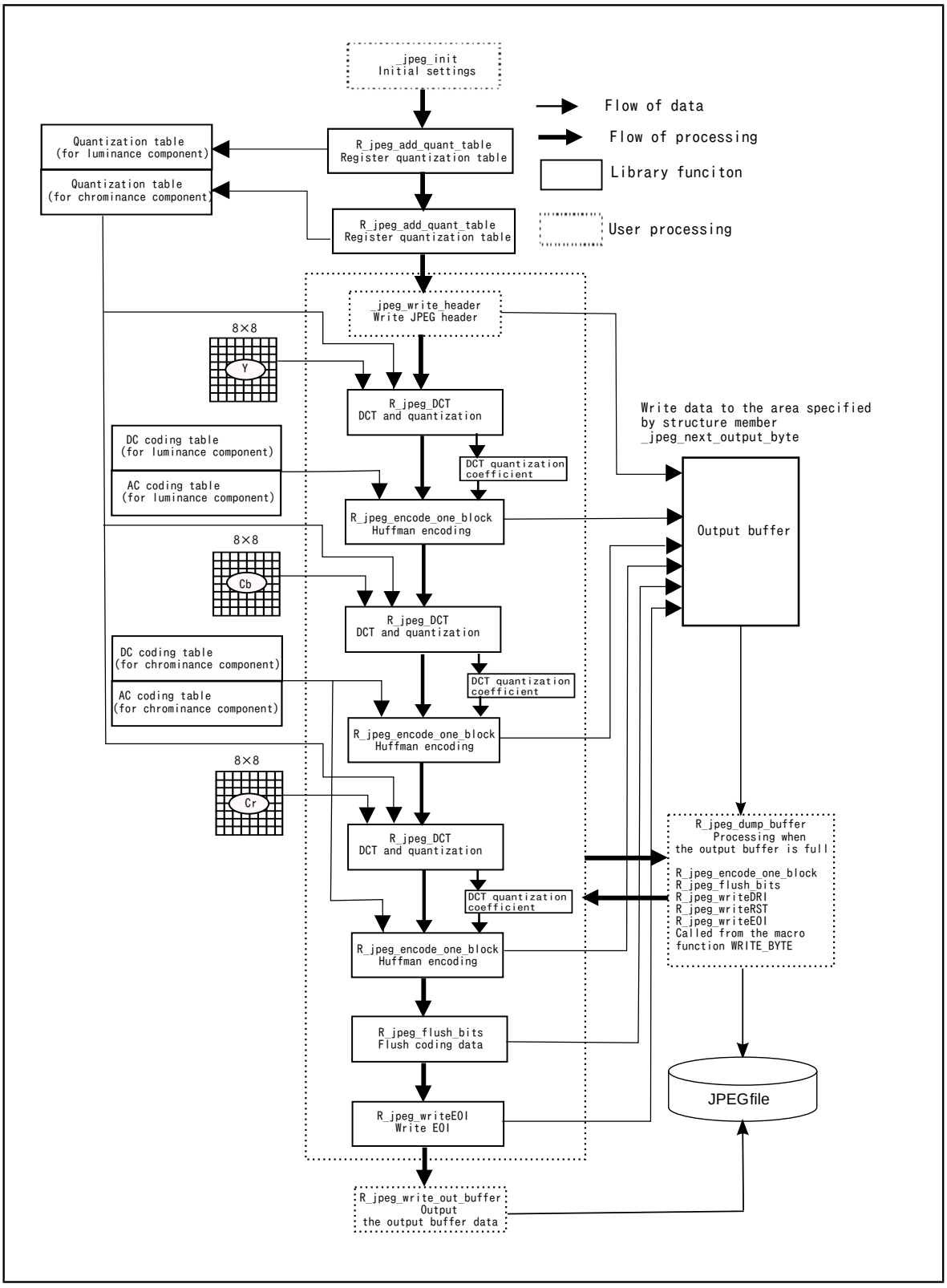

Figure 3.1. JPEG Encode Library Structure

3.3. Data Configuration

3.3.1. Library Structures

3.3.1.1. JPEG Software Library Environment Variables

To enable easy access to the data used internally by the JPEG Encode Library, a single structure is used in JPEG Encoder to manage all the data used internally by the library functions. The name "JPEG software library environment variable structure" is used to refer to this structure in this manual.

Figure 3.2, “JPEG Software Library Environment Variable Structure _jpeg_working” shows the contents of the structure.

```
struct _jpeg_working{
    /* encode */
    struct _jpeg_enc_FMB *encFMB;
    struct _jpeg_enc_FMC *encFMC;
    struct _jpeg_enc_SMB *encSMB;
    struct _jpeg_enc_SMC *encSMC;
    void (*enc_dump_func)(struct _jpeg_working *);

    /* decode */
    struct _jpeg_dec_FMB *decFMB;
    struct _jpeg_dec_FMC *decFMC;
    struct _jpeg_dec_SMB *decSMB;
    void (*dec_read_input)(struct _jpeg_working *);
};
```

Figure 3.2. JPEG Software Library Environment Variable Structure _jpeg_working

In the application program, make sure to set aside an area for the JPEG software library environment variable structure. During compression it is not necessary for the application program to access or set the members listed under "decode" in the above figure.

When the JPEG Encode Library is used in conjunction with the JPEG File Compress Library, the JPEG File Compress Library maintains, references, and makes settings to this area.

3.3.1.2. JPEG Encode Library Fast Memory Variables

The variables that are used most frequently by the JPEG Encode Library are defined by the structure _jpeg_enc_FMB. Figure 3.3, “JPEG Encode Library Fast Memory Variable Structure_jpeg_enc_FMB” lists the contents of the structure _jpeg_enc_FMB.

```
#define _JPEG_DCTSIZE2        64
#define _JPEG_COMPONENT_NUM   3   /* YCbCr */

struct _jpeg_enc_FMB {
  int32_t _jpeg_work[_JPEG_DCTSIZE2];
  int16_t _jpeg_DCTqtbl[_JPEG_DCTSIZE2*3];
  uint8_t *_jpeg_next_output_byte;
  int32_t _jpeg_free_in_buffer;
  uint32_t _jpeg_cur_put_buffer;
  uint32_t _jpeg_cur_put_bits;
  int32_t _jpeg_restart_interval;
  int32_t _jpeg_thinning_mode;
  int32_t _jpeg_dc_size[_JPEG_COMPONENT_NUM+1];
  int32_t _jpeg_ac_size[_JPEG_COMPONENT_NUM+1];
};
```

Figure 3.3. JPEG Encode Library Fast Memory Variable Structure_jpeg_enc_FMB

The members of structure _jpeg_enc_FMB are listed in Table 3.3, “Members of Structure _jpeg_enc_FMB” .

Table 3.3. Members of Structure _jpeg_enc_FMB

Structure Member	Symbol	Functionality
_jpeg_work[]	-	reserved
_jpeg_DCTqtbl[]	-	reserved
_jpeg_next_output_byte	-	Pointer containing the address of the next byte of encoded data to be read
_jpeg_free_in_buffer	-	The amount of capacity (in bytes) remaining in the output buffer
_jpeg_cur_put_buffer	-	reserved
_jpeg_cur_put_bits	-	reserved
_jpeg_restart_interval	Ri	Encoding reinitialization interval
_jpeg_thinning_mode	-	reserved
_jpeg_dc_size[]	-	reserved
_jpeg_ac_size[]	-	reserved

The symbols in the above table refer to parameter symbols defined in the reference documents.

3.3.1.3. JPEG Encode Library Fast Memory Constants

The constants that are used most frequently by the JPEG Encode Library are defined by the structure _jpeg_enc_FMC. Figure 3.4, “JPEG Encode Library Fast Memory Constant Structure _jpeg_enc_FMC” lists the contents of the structure _jpeg_enc_FMC.

```
struct _jpeg_enc_FMC
{
 uint32_t _jpeg_fmcData[634];
};
```

Figure 3.4. JPEG Encode Library Fast Memory Constant Structure _jpeg_enc_FMC

The constants in _jpeg_enc_FMC are defined in the JPEG Encode Library, and they can be accessed by the application program by means of the _top_of_jpeg_enc_FMC symbol. Figure 3.5, “Initialization of Structure _jpeg_enc_FMC” shows the method of registering items in the JPEG software library environment variable structure.

```
#include ”r_jpeg.h”

struct _jpeg_working working; // JPEG software library environment variable structure

void sample(void)
{
  working.encFMC = &_top_of_jpeg_enc_FMC;
        …
}
```

Figure 3.5. Initialization of Structure _jpeg_enc_FMC

The members of structure _jpeg_enc_FMC are listed in Table 3.4, “Members of Structure _jpeg_enc_FMC” .

Table 3.4. Members of Structure _jpeg_enc_FMC

Structure Member	Symbol	Functionality
_jpeg_fmcData[]	-	reserved

The symbols in the above table refer to parameter symbols defined in the reference documents.

3.3.1.4. JPEG Encode Library Slow Memory Variables

The variables that are used less frequently by the JPEG Encode Library are defined by the structure _jpeg_enc_SMB. Figure 3.6, “JPEG Encode Library Slow Memory Variable Structure _jpeg_enc_SMB” lists the contents of the structure _jpeg_enc_SMB.

```
#define _JPEG_DCTSIZE2        64
#define _JPEG_COMPONENT_NUM   3   /* YCbCr */

struct component_info
{
  uint8_t component_id[_JPEG_COMPONENT_NUM+1];    /* +1 for alignment */
  uint8_t hsample_ratio[_JPEG_COMPONENT_NUM+1];   /* +1 for alignment */
  uint8_t vsample_ratio[_JPEG_COMPONENT_NUM+1];   /* +1 for alignment */
  uint8_t quant_tbl_no[_JPEG_COMPONENT_NUM+1];    /* +1 for alignment */
};

struct frame_component_info
{
  uint8_t component_id[_JPEG_COMPONENT_NUM+1];    /* +1 for alignment */
  uint8_t dc_tbl_no[_JPEG_COMPONENT_NUM+1];       /* +1 for alignment */
  uint8_t ac_tbl_no[_JPEG_COMPONENT_NUM+1];       /* +1 for alignment */
};

struct _jpeg_enc_SMB {
  int32_t _jpeg_num_QTBL;
  uint16_t _jpeg_QTBL[_JPEG_DCTSIZE2*_JPEG_COMPONENT_NUM];
  uint8_t _jpeg_precision_QTBL[_JPEG_COMPONENT_NUM+1];
  uint16_t _jpeg_number_of_lines;
  uint16_t _jpeg_line_length;
  struct component_info component_info;
  struct frame_component_info frame_component_info;
  int32_t _jpeg_frame_num_of_components;
  uint8_t _jpeg_write_DQT;
  uint8_t _jpeg_write_DHT;
  uint8_t dummy1;
  uint8_t dummy2;
};
```

Figure 3.6. JPEG Encode Library Slow Memory Variable Structure _jpeg_enc_SMB

The members of structure _jpeg_enc_SMB are listed in Table 3.5, “Members of Structure _jpeg_enc_SMB” .

Table 3.5. Members of Structure _jpeg_enc_SMB

Structure Member	Symbol	Functionality
_jpeg_num_QTBL	-	Number of quantization tables
_jpeg_QTBL[]	-	Quantization table array Registered by the R_jpeg_add_quant_table function. Used as data when outputting the DQT block of the JPEG file.

Structure Member	Symbol	Functionality
_jpeg_precision_QTBL[]	-	reserved
_jpeg_number_of_lines	Y	Line count of output image
_jpeg_line_length	X	Number of pixels per line
struct component_info { ????uint8_t component_id[]; ????uint8_t hsample_ratio[]; ????uint8_t vsample_ratio[]; ????uint8_t quant_tbl_no[]; } component_info	-	Color component information Refer to the table below for a listing of the members of the structure.
struct frame_component_info { ????uint8_t component_id[]; ????uint8_t dc_tbl_no[]; ????uint8_t ac_tbl_no[]; } frame_component_info	-	Frame information Refer to the table below for a listing of the members of the structure.
_jpeg_frame_num_of_components	Ns	Number of components in scan
_jpeg_write_DQT	-	Is a quantization table created? 0: No 1: Yes
_jpeg_write_DHT	-	Is a Huffman code table created? 0: No 1: Yes
dummy1	-	reserved
dummy2	-	reserved

Table 3.6. Members of Structure component_info

Structure Member	Symbol	Functionality
component_id[]	Ci	Component identifier
hsample_ratio[]	Hi	Component horizontal sampling ratio
vsample_ratio[]	Vi	Component vertical sampling ratio
quant_tbl_no[]	Tqi	Quantization table number used by component

Table 3.7. Members of Structure frame_component_info

Structure Member	Symbol	Functionality
component_id[]	Csj	Component identifier
dc_tbl_no[]	Tdj	Array containing the Huffman table DC table number (0 or 1) used by the component
ac_tbl_no[]	Taj	Array containing the Huffman table AC table number (0 or 1) used by the component

The symbols in the above table refer to parameter symbols defined in the reference documents.

3.3.1.5. JPEG Encode Library Slow Memory Constants

The constants that are used less frequently by the JPEG Encode Library are defined by the structure _jpeg_enc_SMC. Figure 3.7, “JPEG Encode Library Slow Memory Constant Structure _jpeg_enc_SMC” lists the contents of the structure _jpeg_enc_SMC.

```
struct _jpeg_enc_SMC
{
 uint32_t _jpeg_smcData[174];
};
```

Figure 3.7. JPEG Encode Library Slow Memory Constant Structure _jpeg_enc_SMC

The constants in _jpeg_enc_SMC are defined in the JPEG Encode Library, and they can be accessed by the application program by means of the _top_of_jpeg_enc_SMC symbol. Figure 3.8, “Initialization of Structure _jpeg_enc_SMC” shows the method of registering items in the JPEG software library environment variable structure.

```
#include ”r_jpeg.h”

struct _jpeg_working working; // JPEG software library environment variable structure

void sample(void)
{
  working.encSMC = &_top_of_jpeg_enc_SMC;
        …
}
```

Figure 3.8. Initialization of Structure _jpeg_enc_SMC

The members of structure _jpeg_enc_SMC are listed in Table 3.8, “Members of Structure _jpeg_enc_SMC” .

Table 3.8. Members of Structure _jpeg_enc_SMC

Structure Member	Symbol	Functionality
_jpeg_smcData[]	-	reserved

The symbols in the above table refer to parameter symbols defined in the reference documents.

3.3.2. Data Input Method

Image data is input in MCU units.

Since it is necessary to allocate the data in the horizontal direction for each color component consecutively, the application should reorder the data to the required order. For example, if the data has the order Y,Cb,Cr,Y,Cb,Cr,Y,Cb,Cr,... it must be reordered to the Y,Y,Y,..., Cb,Cb,Cb,... and Cr,Cr,Cr,... order.

See the description of the R_jpeg_DCT function, which is the first function to process the image data.

3.3.3. Data Output Method

The data output by this library is stored in the RAM area set as the JPEG Encode Library output buffer. The JPEG file header section and part of, or all of, the encoded data is stored in the output buffer. If the output buffer becomes full, the output buffer is cleared by writing the contents of the output buffer to some form of storage media. This process is repeated until compression of all of the image data has completed.

■Output buffer management

Output buffers are managed with the two members of the _jpeg_enc_FMB structure.

```
uint8_t *_jpeg_next_output_byte;     // Next data byte write position
int32_t _jpeg_free_in_buffer;        // Remaining buffer capacity (in bytes)
```

When one byte of data is written to the output buffer, _jpeg_next_output_byte is incremented by 1 and _jpeg_free_in_buffer is decremented by 1.

The above two members should be initialized before using the JPEG Encode Library.

■Order of data storage to the output buffer

The content and order of the data stored in output buffers depends on the application. The following presents one example of the possible storage orders.

1. The JPEG file header section (the markers and substrings recorded recorder before the SOI, APPn, DQT, SOF0, DHT, DRI, SOS and other encoded data) is written.
2. The DCT quantization coefficients are generated from the Y, Cb, and Cr component data and then the encoded data is generated and the encoded data is written to the output buffer. If the encoding reinitialization interval is enabled, the RSTm marker is written by the R_jpeg_writeRST function.
3. Encoded data is written forcibly by the R_jpeg_flush_bits function.
4. EOI marker codes are written by the R_jpeg_writeEOI function.

■Processing when the output buffer becomes full

If the output buffer becomes full during execution of one of the R_jpeg_encode_one_block function, the R_jpeg_flush_bits function, the R_jpeg_writeDRI function, the R_jpeg_writeRST function or the R_jpeg_writeEOI function, the function registered in the enc_dump_func member of the JPEG software library environment variable structure is called.

The output data should be written to storage media of some sort by this function. After writing, the output buffer should be reinitialized. If it is not reinitialized, an error will occur in the function that performs the next write. See the description of the user-defined function R_jpeg_dump_buffer.

3.3.4. Macro Definitions

The macro definitions contained in the header file r_jpeg.h are shown below.

Table 3.9. Error Code Definitions

Definition	Value	Description
_JPEG_OK	0	Normal end
_JPEG_ERROR	-1	Error end

Table 3.10. Constant Definitions

Definition	Value	Description
_JPEG_DCTSIZE	8	DCT size
_JPEG_DCTSIZE2	64	Square of DCT size
_JPEG_COMPONENT_NUM	3	Number of components
_JPEG_HUFFVAL_SIZE	256	Number of Huffman codes
_JPEG_BITS_SIZE	17	Number of Huffman bits

Table 3.11. Macro Variable Definitions (Fast Memory Variables _jpeg_enc_FMB)

Definition	Description
_jpeg_DCTqtbl(base)	((base)->_jpeg_DCTqtbl)
_jpeg_next_output_byte(base)	((base)->_jpeg_next_output_byte)
_jpeg_free_in_buffer(base)	((base)->_jpeg_free_in_buffer)
_jpeg_cur_put_buffer(base)	((base)->_jpeg_cur_put_buffer)
_jpeg_cur_put_bits(base)	((base)->_jpeg_cur_put_bits)
_jpeg_restart_interval(base)	((base)->_jpeg_restart_interval)
_jpeg_dc_size(base)	((base)->_jpeg_dc_size)
_jpeg_ac_size(base)	((base)->_jpeg_ac_size)

Table 3.12. Macro Variable Definitions (Slow Memory Variables _jpeg_enc_SMB)

Definition	Description
_jpeg_num_QTBL(base)	((base)->_jpeg_num_QTBL)
_jpeg_QTBL(base)	((base)->_jpeg_QTBL)
_jpeg_precision_QTBL(base)	((base)->_jpeg_precision_QTBL)
_jpeg_number_of_lines(base)	((base)->_jpeg_number_of_lines)
_jpeg_line_length(base)	((base)->_jpeg_line_length)
component_info(base)	((base)->component_info)

Definition	Description
frame_component_info(base)	((base)->frame_component_info)
_jpeg_frame_num_of_components (base)	((base)->_jpeg_frame_num_of_components)
_jpeg_write_DQT(base)	((base)->_jpeg_write_DQT)
_jpeg_write_DHT(base)	((base)->_jpeg_write_DHT)

Table 3.13. Data Write Macro Definition

Definition	Description
WRITE_BYTE(c, env)	Writes 1 byte of data

3.4. Library Functions

In this section the functions of the JPEG Encode Library are described in detail. The format of the detailed function descriptions is as follows:

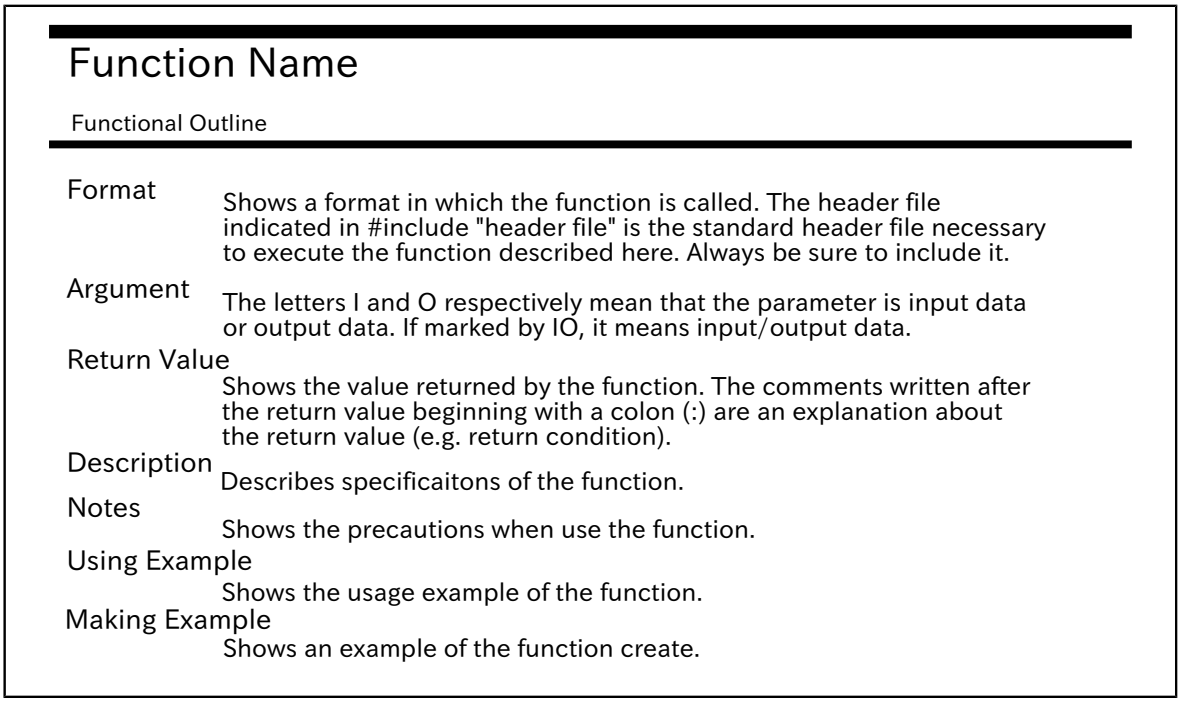

Figure 3.9. Format of Detailed Library Function Descriptions

R_jpeg_add_quant_table

Library Functions

— Registers quantization tables

Format

```
#include "r_jpeg.h"

void R_jpeg_add_quant_table (
                int32_t qtbl_no ,
                const uint16_t *basic_table ,
                int32_t quality ,
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
qtbl_no	I	Quantization table number
basic_table	I	Pointer to quantization table
quality	I	Precision coefficient of quantization table
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

None

Description

For the JPEG software library environment specified by wenv, this function references the quality values in the quantization table specified by basic_table and registers them in the quantization table specified by qtbl_no.

The number (0, 1, or 2) of the quantization table to be registered is specified by qtbl_no. Operation cannot be guaranteed if a value other than 0, 1, or 2 is specified for qtbl_no.

A pointer to the area (int16_t type 8×8 array) storing the contents of the quantization table to be registered by this function is specified by basic_table.

A value of 1 to 128 is specified by quality. This argument is used to change the image quality and image compression ratio. A higher value for quality corresponds to higher image quality but also a lower compression ratio. Using a low value for quality results in a higher compression ratio but also degrades the image quality. Operation cannot be guaranteed if a value less than 1 or greater than 128 is specified for quality.

Notes

The quantization table registration by this function should be performed before the user-defined function _jpeg_write_header.

Using Example

```
#include  ”r_jpeg.h”
```

```
struct _jpeg_working working; // JPEG software library environment variable structure

//   Quantization table for Y component
uint16_t _jpeg_std_luminance_quant_tbl[] = {
  16,  11,  10,  16,  24,  40,  51,  61,
  12,  12,  14,  19,  26,  58,  60,  55,
  14,  13,  16,  24,  40,  57,  69,  56,
  14,  17,  22,  29,  51,  87,  80,  62,
  18,  22,  37,  56,  68, 109, 103,  77,
  24,  35,  55,  64,  81, 104, 113,  92,
  49,  64,  78,  87, 103, 121, 120, 101,
  72,  92,  95,  98, 112, 100, 103,  99
};

//   Quantization table for Cb or Cr component
uint16_t _jpeg_std_chrominance_quant_tbl[] = {
  17,  18,  24,  47,  99,  99,  99,  99,
  18,  21,  26,  66,  99,  99,  99,  99,
  24,  26,  56,  99,  99,  99,  99,  99,
  47,  66,  99,  99,  99,  99,  99,  99,
  99,  99,  99,  99,  99,  99,  99,  99,
  99,  99,  99,  99,  99,  99,  99,  99,
  99,  99,  99,  99,  99,  99,  99,  99,
  99,  99,  99,  99,  99,  99,  99,  99
};

void sample (void)
{
          ...
  //    Quantization table registration
  R_jpeg_add_quant_table(0, _jpeg_std_luminance_quant_tbl, 64, &working);
  R_jpeg_add_quant_table(1, _jpeg_std_chrominance_quant_tbl, 64, &working);
          ...
}
```

R_jpeg_DCT

Library Functions

— Executes DCT and quantization

Format

```
#include "r_jpeg.h"

void R_jpeg_DCT (
                uint8_t *sample_data[] ,
                int32_t start_col ,
                int32_t qtbl_no ,
                int16_t *outptr ,
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
sample_data	I	Pointer to pointer array specifying line to be processed
start_col	I	Offset (in bytes) from beginning of line of processing target block
qtbl_no	I	Quantization table number
outptr	O	Pointer to storage area for quantized DCT coefficients
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

None

Description

For the JPEG software library environment specified by wenv, this function references the quantization table specified by qtbl_no and performs DCT and quantization on the component data in the area specified by sample_data and start_col. It then stores the DCT quantization coefficients in the area specified by outptr.

A pointer to a pointer array specifying the line to be processed is specified by sample_data. Thus, the lines specified by sample_data[0], sample_data[1], ⋯ constitute the image data processed by this function.

Make sure to specify addresses corresponding to 4-byte boundaries for sample_data[0], sample_data[1], ⋯. Operation cannot be guaranteed if addresses that do not correspond to 4-byte boundaries are specified.

The offset (in bytes) of the processing target block from the beginning of the line specified by sample_data[] is specified by start_col.

The numbers (0, 1, or 2) of the quantization table is specified by qtbl_no. Operation cannot be guaranteed if a value other than 0, 1, or 2 is specified for qtbl_no.

The DCT quantization coefficients obtained by this function are stored in the area (int16_t type 8×8 array) specified by outptr.

Figure 3.10, “ DCT Processing of Component Data by the R_jpeg_DCT Function ” shows an example of DCT processing and quantization of component data for a 32 × 32-pixel area. For example 1 the processing target is block 1, and block 6 is the processing target of example 2.

Notes

It is necessary to register the quantization table with the R_jpeg_add_quant_table function before executing this function.

Using Example

```
#include  ”r_jpeg.h”

struct _jpeg_working    working;      // JPEG library environment variable structure
static int16_t          MCU_data[64]; // MCU block data

void sample(void)
{
  uint8_t   *Y_sample_data[8];
  uint8_t   *Cb_sample_data[8];
  uint8_t   *Cr_sample_data[8];
  int       Y_last_dc_val=0;
  int       Cb_last_dc_val=0;
  int       Cr_last_dc_val=0;
  int       Y_start_col, Cb_start_col, Cr_start_col;

  // Y_sample_data[8], Y_start_col Initialization
          …
  // Cb_sample_data[8], Cb_start_col Initialization
          …
  // Cr_sample_data[8], Cr_start_col Initialization
          …
          //  Y0 component
          R_jpeg_DCT(Y_sample_data, Y_start_col, 0, MCU_data, &working);
          R_jpeg_encode_one_block(MCU_data, Y_last_dc_val, 0, 0, &working);
          Y_last_dc_val = MCU_data[0];
                    …
          //  Y1 component
          R_jpeg_DCT(Y_sample_data, Y_start_col+8, 0, MCU_data, &working);
          R_jpeg_encode_one_block(MCU_data, Y_last_dc_val, 0, 0, &working );
          Y_last_dc_val = MCU_data[0];
                    …
          //  Cb component
          R_jpeg_DCT(Cb_sample_data, Cb_start_col, 0, MCU_data, &working);
          R_jpeg_encode_one_block(MCU_data,  Cb_last_dc_val, 0, 0, &working );
          Cb_last_dc_val = MCU_data[0];
                    …
          //  Cr component
          R_jpeg_DCT(Cr_sample_data, Cr_start_col, 0, MCU_data, &working);
          R_jpeg_encode_one_block(MCU_data, Cr_last_dc_val, 0, 0, &working);
          Cr_last_dc_val = MCU_data[0];
                    …
}
```

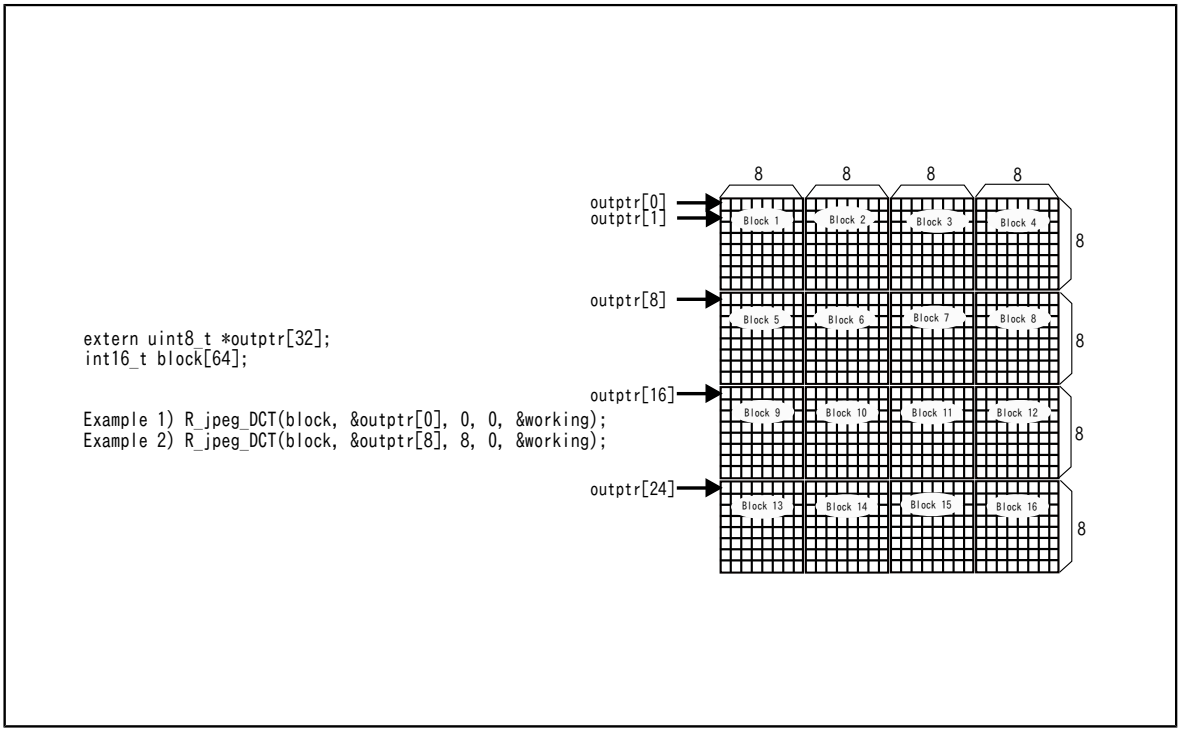

Figure 3.10. DCT Processing of Component Data by the R_jpeg_DCT Function

R_jpeg_encode_one_block

Library Functions

— Executes Huffman encoding

Format

```
#include "r_jpeg.h"

int32_t R_jpeg_encode_one_block (
                int16_t *block ,
                int32_t last_dc_val ,
                int32_t dc_tbl_no ,
                int32_t ac_tbl_no ,
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
block	I	Pointer to storage area of quantized DCT coefficients
last_dc_val	I	DC coefficients of previous block
dc_tbl_no	I	Table number of DC coefficient Huffman coding table
ac_tbl_no	I	Table number of AC coefficient Huffman coding table
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

Return value	Description
0	Normal end
Other than 0	Error

Description

For the JPEG software library environment specified by wenv, this function performs Huffman encoding of the DCT quantization coefficients stored in the area specified by block and then stores the encoded data in the output buffer.

A pointer to an area (int16_t type 8×8 array) storing the quantized DCT coefficients to be processed is specified by block.

DC coefficients of the previous block are specified by last_dc_val.

The DC coefficient Huffman coding table number (0 or 1) is specified by dc_tbl_no. Specify 0 when processing the Y component, and specify 1 when processing the Cb or Cr component. Operation cannot be guaranteed if a value other than 0 or 1 is specified for dc_tbl_no.

The AC coefficient Huffman coding table number (0 or 1) is specified by ac_tbl_no. Specify 0 when processing the Y component, and specify 1 when processing the Cb or Cr component. Operation cannot be guaranteed if a value other than 0 or 1 is specified for ac_tbl_no.

Notes

If the output buffer becomes full during execution of this function, the user-defined function R_jpeg_dump_buffer is called and the output buffer is initialized. After execution of the R_jpeg_dump_buffer function completes, execution of this function continues. See the description of the R_jpeg_write_out_buffer function for the specifications of the R_jpeg_dump_buffer function.

Using Example

Refer to the usage example for the R_jpeg_DCT function.

R_jpeg_writeDRI

Library Functions

— Writes DRI

Format

```
#include "r_jpeg.h"

int32_t R_jpeg_writeDRI (
                int32_t Ri ,
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
Ri	I	Encoding reinitialization interval
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

Return value	Description
0	Normal end
Other than 0	Error

Description

For the JPEG software library environment specified by wenv, this function writes to the output buffer the encoding reinitialization interval marker DRI and the encoding reinitialization interval specified by Ri.

The encoding reinitialization interval is specified by Ri.

Notes

Use this function in combination with the R_jpeg_writeRST function. It is not possible for the R_jpeg_writeRST function to write the encoding reinitialization marker RSTm when the encoding reinitialization interval is set to 0.

Using Example

```
#include  "r_jpeg.h"

struct _jpeg_working     working; // JPEG software library environment variable structure

void sample(void)
{
        ...
    int restart_interval;
        ...
```

```
    restart_interval = 8;                // Sets the encoding reinitialization interval to 8
    (working.encFMB)->_jpeg_restart_interval = restart_interval;
        ...

    if (restart_interval) {
      R_jpeg_writeDRI(restart_interval, wenv);
    }
        ...

}
```

R_jpeg_writeRST

Library Functions

— Writes RSTm

Format

```
#include "r_jpeg.h"

int32_t R_jpeg_writeRST (
                int32_t m ,
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
m	I	Encoding reinitialization interval number
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

Return value	Description
0	Normal end
Other than 0	Error

Description

For the JPEG software library environment specified by wenv, this function writes to the output buffer the encoding reinitialization marker RST and the encoding reinitialization interval number specified by m.

The encoding reinitialization interval m is specified as a modulo 8 counter that repeats values from 0 to 7.

Notes

Use this function in combination with the R_jpeg_writeDRI function.

This function is valid when the encoding reinitialization interval definition marker DRI has been written by the R_jpeg_writeDRI function and the value of the argument Ri is 1 or greater.

Using Example

```
#include  "r_jpeg.h"

struct _jpeg_working  working; // JPEG software library environment variable structure

void sample(void)
{
        ...
  int16_t restart_interval;
```

```
    extern int16_t MCU_count;             // MCU counter
    extern int16_t next_marker_num;       // Encoding reinitialization interval number, initial value 0
          …

    restart_interval = (working.encFMB)->_jpeg_restart_interval;
          …
    // Reads RSTm marker
    if(restart_interval) {
      if(MCU_count == 0) {
        R_jpeg_writeRST(next_marker_num,&working);  // Writes encoding reinitialization marker
        next_marker_num=(next_marker_num+1)&7;      // Updates encoding reinitialization interval number
        MCU_count = restart_interval - 1;           // Updates MCU counter
      }
      else {
        --MCU_count;
      }
    }
          …
  }
```

R_jpeg_writeEOI

Library Functions

— Writes EOI marker

Format

```
#include "r_jpeg.h"

int32_t R_jpeg_writeEOI (
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

Return value	Description
0	Normal end
Other than 0	Error

Description

For the JPEG software library environment specified by wenv, this function writes the end of image marker EOI to the output buffer.

Notes

Call this function after encoding of all components has finished and after forcibly writing the encoded data with the R_jpeg_flush_bits function.

Using Example

```
#include  ”r_jpeg.h”

struct _jpeg_working  working; // JPEG software library environment variable structure

void sample(void)
{

        …
  R_jpeg_flush_bits(&working);              // Forcibly writes encoded bits
  R_jpeg_writeEOI(&working);                // Writes the EOI marker
        …

}
```

R_jpeg_flush_bits

Library Functions

— Forcibly writes Huffman encoded data

Format

```
#include "r_jpeg.h"

int32_t R_jpeg_flush_bits (
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

Return value	Description
0	Normal end
Other than 0	Error

Description

For the JPEG software library environment specified by wenv, this function adds padding and writes to the output buffer encoded data of less than one byte retained by the R_jpeg_encode_one_block function, from among the encoded data generated by the R_jpeg_encode_one_block function.

Making Example

Refer to the usage example for the R_jpeg_writeEOI function.

3.5. User-Defined Functions

The user-defined function R_jpeg_dump_buffer is required to use the JPEG Encode Library.

R_jpeg_dump_buffer

User-defined function

— Outputs the JPEG file

Format

```
#include "r_jpeg.h"

void R_jpeg_dump_buffer (
                struct _jpeg_working *wenv );
```

Argument

Argument	I/O	Description
wenv	I/O	Pointer to JPEG software library environment variable structure

Return value

None

Description

This function transfers all of the data stored in the output buffer for JPEG software library environment specified by the argument wenv to the area specified by the application.

This function is a lower-level function used by the R_jpeg_encode_one_block function, the R_jpeg_flush_bits function, the R_jpeg_writeDRI function, the R_jpeg_writeRST function or the R_jpeg_writeEOI function. This function is called from one of the above functions when, during storage of encoded data, the output buffer becomes full.

This function should output the data in the output buffer starting from the start of the buffer area. The output destination depends on the target system.

作成例

The following two programs are coding examples for this function.

1. Registration of this function in the enc_dump_func member in the JPEG software library environment variable structure
2. Coding example for the main JPEG file output function that is called from the R_jpeg_dump_buffer function

1. Registration of this function in the enc_dump_func member in the JPEG software library environment variable structure

```
#include  ”r_jpeg.h”

extern void _jpeg_dump_buffer(void);
struct _jpeg_working working;

void sample()
{
        :
```

```
    working.enc_dump_func = _jpeg_dump_buffer;
        :
}
```

2. R_jpeg_dump_buffer implementation example

The following is a coding example for the R_jpeg_dump_buffer function for the case where the data stored in output buffer _jpeg_buffer[] is transferred to the transfer start address specified by _jpeg_file_topaddr.

```
#define OUTPUT_BUFFER_SIZE 2048   /* JPEG data output buffer size */
 :
uint8_t _jpeg_buffer[OUTPUT_BUFFER_SIZE]; /* JPEG data output buffer */
uint8_t *_jpeg_file_topaddr;      /* JPEG data file address (JPEG data storage destination) */
uint16_t _jpeg_file_size;         /* JPEG data file size */
 :
void R_jpeg_dump_buffer(void)
{
  struct _jpeg_enc_FMB *FMBbase;

  FMBbase = wenv->encFMB;
//---------------------------------------------------------
// Here, the processing for transfer of JPEG data stored in the
// output buffer to the storage media is performed.
//---------------------------------------------------------
  memcpy( _jpeg_file_topaddr+ _jpeg_file_size, _jpeg_buffer, OUTPUT_BUFFER_SIZE );

  /* Increment by an amount that is the size copied JPEG file size. */
  _jpeg_file_size += OUTPUT_BUFFER_SIZE;

//---------------------------------------------------------
// Library variable initialization
//---------------------------------------------------------
  /* Reset the free size for the output buffer (set the output buffer to empty) */
  _jpeg_free_in_buffer(FMBbase) = OUTPUT_BUFFER_SIZE;
  /* Return the write position for the output buffer to the start of the buffer. */
  _jpeg_next_output_byte(FMBbase) = _jpeg_buffer;
}
```

JPEG Encoder
User's Manual

Publication Date 30 Sep 2013 Rev.1.00
30 Sep 2013 Rev.1.00
Publisher Renesas Electronics Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki,
Kanagawa 211-8668, Japan

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

California Eastern Laboratories, Inc.
4590 Patrick Henry Drive, Santa Clara, California 95054-1817, U.S.A.
Tel: +1-408-919-2500, Fax: +1-408-988-0279

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

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

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

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

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

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

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

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

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

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

Colophon 4.0

JPEG Encoder
User's Manual

Renesas Electronics Corporation

R20UW0122EJ0100