User's Manual

JPEGエンコーダ

ユーザーズマニュアル

ルネサスマイクロコンピュータ ミドルウェア

ルネサス エレクトロニクス
www.renesas.com

Rev.1.00　2013.09

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

(2012.4)

はじめに

JPEGエンコーダは、画像をJPEGファイルに圧縮するソフトウェアライブラリです。 DCT利用型シーケンシャルを採用しています。

本マニュアルでは、JPEGエンコーダを使ってアプリケーションプログラムを作成するための情報を提供します。

1. 参考文献

JPEGエンコーダは次の規格書に示される仕様をもとに作成されたソフトウェアです。本マニュアルとあわせてご参照ください。

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

■JIS X 4301-1995 連続階調静止画像のディジタル圧縮及び符号処理

2. 本マニュアルの表記規則

本マニュアルでは、とくに説明のない限り以下の表記規則に示す用語を使用して説明しています。

表1.記号

表記	意味
数字	マニュアル上に指定がない限り10進数を表す

表2.用語

表記	意味
元画像	輝度（Y）や色差（Cb,Cr）を合成して表現される元の画像
成分	輝度（Y）や色差（Cb,Cr）を表すそれぞれのデータの集合
JPEGファイル	JPEG方式の画像ファイル
JPEGヘッダ	JPEGファイル中のSOIマーカ、APPn部分列、DQT部分列、SOF0部分列、DHT部分列、DRI部分列、SOS部分列のデータ
イメージデータ	JPEGファイル中のSOSに続く画像データ
ブロック	JPEGエンコーダのライブラリ関数によって画像を圧縮または伸張する場合の処理の対象となるデータの単位
マーカ	最初のバイトは16進数でFF、第2バイトが1から16進数のFEの値であるような2バイトコード
部分列	マーカとそれにつづく引数を表す
DCT	離散コサイン変換 (Discrete Cosine Transform)
逆DCT	逆離散コサイン変換 (Inverse Discrete Cosine Transform)
DCT係数	DCTによって得られた係数（量子化は行われていない）
量子化DCT係数	DCTと量子化によって得られた係数

表記	意味
MCU	最小符号化単位 符号化される最小のデータ単位グループ
リスタートインターバル	符号化再初期化間隔 1つのスキャン内で、独立のシーケンスとして処理されるMCUの数
リスタートマーカ (RSTm)	1つのスキャン内の2つのリスタートインターバルを分けるマーカ
ハフマン符号化	それぞれの入力シンボルに対して可変長コードを割り当てるエントロピ符号化手順
ハフマン復号化	ハフマン符号化された値を元に戻す手順
ハフマンテーブル	ハフマン符号化とハフマン復号化に必要な可変長コードのセット
ジグザグシーケンス	最低空間の空間周波数から最高の空間周波数までのDCT係数のある特定のシーケンシャル順序

3. 本マニュアルの構成

■1章JPEGエンコーダの概要
JPEGエンコーダの概要を説明しています。

■2章JPEGファイル圧縮ライブラリ
簡単に画像をJPEGファイルに圧縮するためのJPEGファイル圧縮ライブラリについて説明しています。

■3章JPEGエンコードライブラリ
JPEG画像の圧縮に必要な量子化とDCTなど基本演算を行うJPEGエンコードライブラリについて説明しています。

補足説明

JPEGエンコーダは、本マニュアルの他に、対応マイコン毎に「導入ガイド」を用意しています。 プログラムのROM/RAMサイズや処理性能、対応マイコン毎の注意事項等をまとめた資料です。 本マニュアルと合わせてご参照ください。

目次

図目次

表目次

1. JPEGエンコーダの概要

本章では、JPEGエンコーダの特徴および開発手順の概要について説明します。

1.1. JPEGエンコーダの特徴

1.1.1. アルゴリズム

JPEGエンコーダは、「ISO/IEC10918-1」の規格および「JIS X 4301」の規格をベースとしたソフトウェアライブラリです。本ライブラリの特徴を次に示します。

■DCT利用型シーケンシャルのアルゴリズム

■ハフマン符号表を用いたエントロピー符号化

■8ビット精度の最大3つの成分に対する画像圧縮が可能

1.1.2. 機能概要

JPEGエンコーダは、JPEGファイルの圧縮を行うJPEGファイル圧縮ライブラリと、基本演算を行うJPEGエンコードライブラリの２つで構成されています。

■JPEGファイル圧縮ライブラリ
JPEGファイル圧縮ライブラリは、ビットマップ画像をJPEGファイルに圧縮するためのライブラリです。 JPEGエンコードライブラリと組み合わせて使用します。 JPEGファイル圧縮ライブラリからJPEGエンコードライブラリを制御するため、ユーザはJPEGファイル圧縮ライブラリのライブラリ関数を使用するだけで、JPEGファイルに圧縮することができます。
JPEGファイル圧縮ライブラリの詳細については**2章JPEGファイル圧縮ライブラリ**をご参照ください。

■JPEGエンコードライブラリ
JPEGエンコードライブラリは、主にJPEGファイルへ圧縮する際に必要な演算を行うライブラリです。 Y、Cr、Cbの成分のデータを入力しDCT変換、量子化、ハフマン符号化を行い、圧縮画像データ（JPEGファイル）を出力します。 JPEGファイル圧縮ライブラリと組み合わせて使用するか、画像データの圧縮部分をアプリケーションプログラムで実装して使用してください。 ライブラリの詳細については**3章JPEGエンコードライブラリ**をご参照ください。

図1.1.「画像データの圧縮と伸張」　に基本ダイヤグラムを示します。

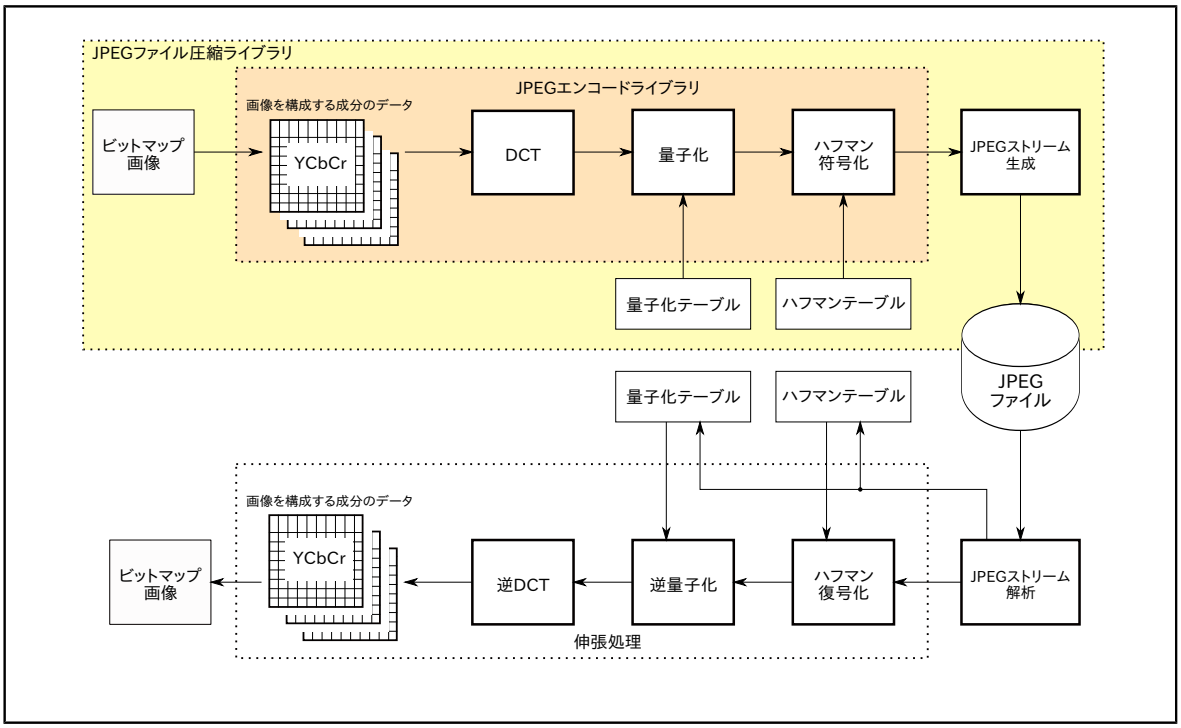

図1.1. 画像データの圧縮と伸張

表1.1.「JPEGファイル圧縮ライブラリの仕様」　を示します。

表1.1.JPEGファイル圧縮ライブラリの仕様

項目		仕様
出力するJPEGファイル	対応フォーマット	JFIF準拠
	色要素	3色(YCbCr)
	サンプル比	4:2:2 (2x1,1x1,1x1) または 4:2:0 (2x2,1x1,1x1)
	画質	1-128で指定可能
	リスタートマーカ	なし、または任意の間隔で設定可能
	コメント	なし
	Exif	非対応
	プログレッシブ	非対応
	サムネイル	非対応
	出力単位	任意のサイズの出力バッファを用意し、その単位で各種メディアに記録することが可能
入力画像	画像フォーマット	RGB565 (16bit color), RGB888 (24bit color), YCbCr 4:2:2
	入力単位	一括読み込み（分割読み込み不可）

JPEGファイル圧縮ライブラリはソースコードが付属するため、ユーザが仕様を変更することができます。 非対応部分に対応させるためには、JPEGファイル圧縮ライブラリのソースコードを編集して機能を追加してください。

付属のソースコードについては、2章JPEGファイル圧縮ライブラリの「ソースコード情報」を参照してください。

1.2. プログラム開発手順

JPEGエンコーダはライブラリ形式で提供いたします。アプリケーションプログラムにライブラリをリンクして使用してください。

図1.2.「アプリケーションプログラムの開発フロー」　を示します。

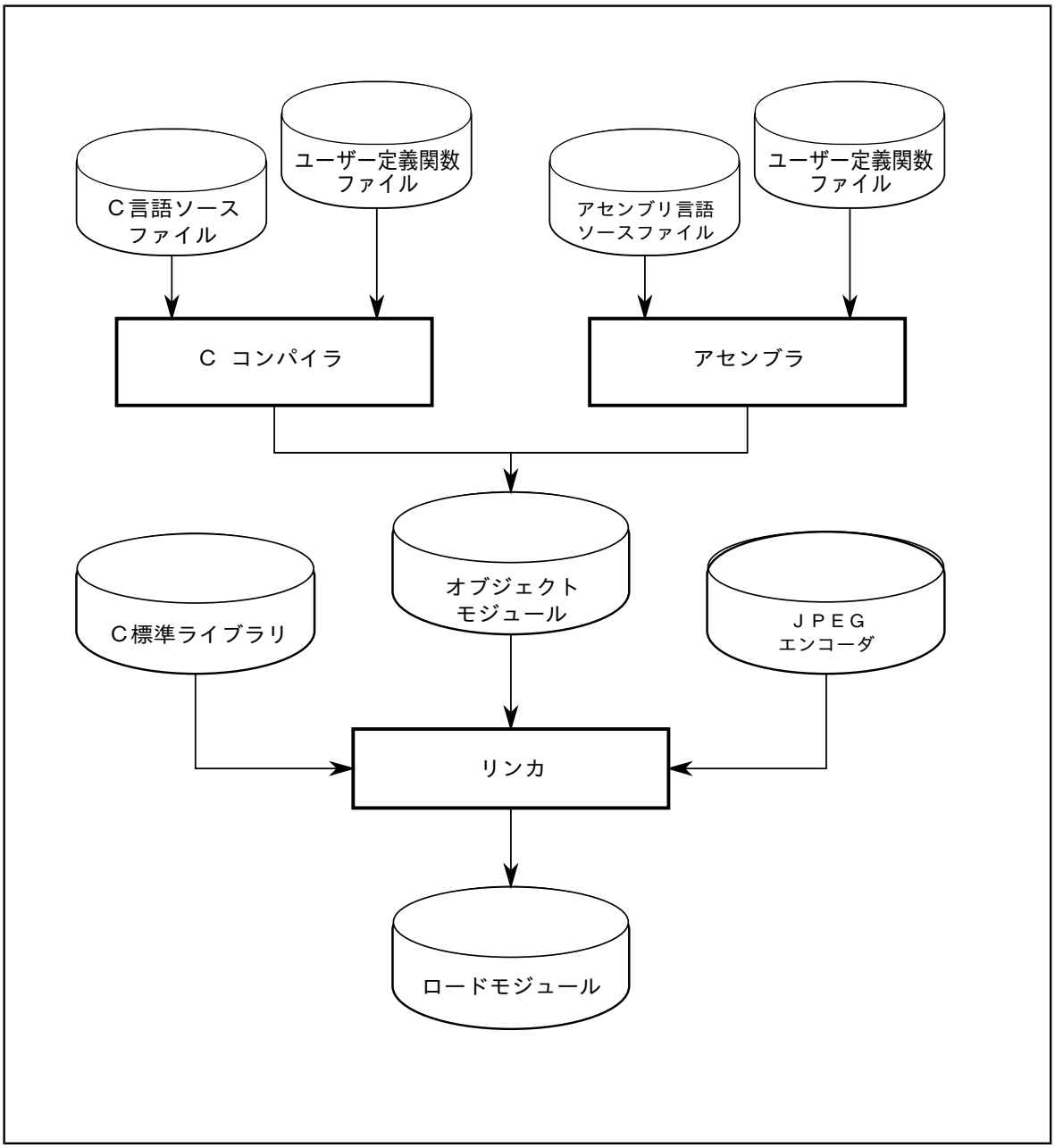

図1.2.アプリケーションプログラムの開発フロー

2. JPEGファイル圧縮ライブラリ

本章では、JPEGファイル圧縮ライブラリについて説明します。

2.1. 概要

JPEGファイル圧縮ライブラリは、ビットマップ画像をJPEGファイルに圧縮するためのライブラリです。 JPEGエンコードライブラリと組み合わせて使用します。 JPEGファイル圧縮ライブラリからJPEGエンコードライブラリを制御するため、ユーザはJPEGファイル圧縮ライブラリのライブラリ関数を使用するだけで、JPEGファイルに圧縮することができます。

JPEGファイル圧縮ライブラリでサポートするライブラリ関数を **表2.1.「ライブラリ関数一覧」** に、 ユーザ定義関数を **表2.2.「ユーザ定義関数一覧」** に示します。

表2.1.ライブラリ関数一覧

関数名	機能概要
R_compress_jpeg	画像のJPEG圧縮

表2.2.ユーザ定義関数一覧

ユーザ定義関数名	機能
R_jpeg_write_out_buffer	出力バッファのデータの出力

2.2. 構成

図2.1.「JPEGファイル圧縮ライブラリの構成」 を示します。

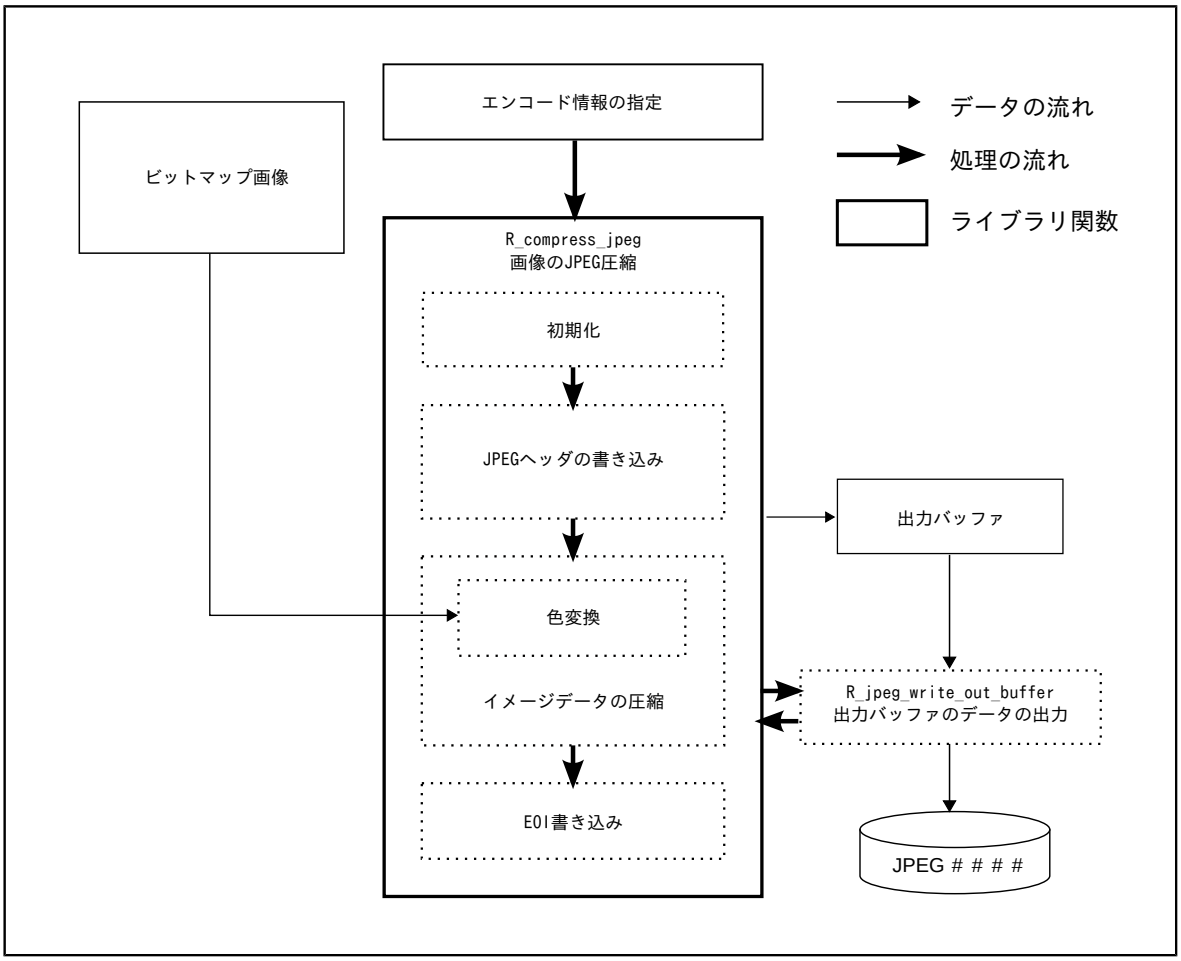

図2.1.JPEGファイル圧縮ライブラリの構成

2.3. データ構造

JPEGファイル圧縮ライブラリで定義するデータ構造について説明します。

2.3.1. ライブラリ構造体

JPEGファイル圧縮ライブラリは、JPEGエンコードライブラリで定義されたライブラリ構造体を使用します。 JPEGファイルの圧縮に必要な変数は全てJPEGファイル圧縮ライブラリが確保しています。 確保している変数については、**「ソースコード情報」**を参照してください。

2.3.1.1. JPEGエンコード情報構造体

JPEGファイル圧縮ライブラリで画像をJPEGファイルに圧縮する場合の各種設定を行う構造体です。

```
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;
}
```

図2.2.JPEGエンコード情報構造体 r_jpeg_encode_t

表2.3.「構造体r_jpeg_encode_tのメンバ」 に各メンバの説明を示します。

表2.3.構造体r_jpeg_encode_tのメンバ

構造体メンバ名	記号	機能概要
width	X	入力画像の横幅
height	Y	入力画像の高さ
quality	-	品質 1-128で指定
restart_interval	Ri	符号化再初期化間隔 0または任意の整数
input_addr	-	入力画像の先頭アドレス
input_line_byte	-	入力画像の領域の1行あたりのバイト数
input_format	-	入力画像の形式
outbuff	-	出力バッファの先頭アドレス
outbuff_size	-	出力バッファのサイズ
output_format	-	出力するJPEGファイルの形式

表2.4.入力画像の形式

定義	内容
JPEGE_IN_RGB565	RGB565(16bit)の画像を入力します

定義	内容
JPEGE_IN_RGB888	RGB888(24bit)の画像を入力します
JPEGE_IN_YCC422	YCbCr 4:2:2の画像を入力します この場合、出力はYCbCr 4:2:2のみ選択できます

表2.5.出力するJPEGファイルの形式

定義	内容
JPEGE_OUT_YCC422	YCbCr 4:2:2で出力します
JPEGE_OUT_YCC420	YCbCr 4:2:0で出力します

2.3.2. データ入力方法

構造体r_jpeg_encode_tのメンバinput_addrで指定したアドレスから、画像データを読み込みます。

画像データは、CPUがアクセスできるメモリ空間上に用意してください。 また圧縮処理の途中で、読み込むアドレスを変更することはできません。

2.3.3. データ出力方法

構造体r_jpeg_encode_tのメンバoutbuffで指定した領域(出力バッファ)へJPEG画像を出力します。

出力バッファが満杯になった場合、ユーザ定義関数R_jpeg_write_out_bufferが呼び出されます。 R_jpeg_write_out_bufferでは、記録メディア(USBメモリ、SDカードなど)にデータを書き込み出力バッファを空にします。 R_jpeg_write_out_bufferを抜けると圧縮処理を再開します。また圧縮が完了した時も同様にR_jpeg_write_out_bufferが呼び出されます。

outbuffのサイズよりJPEGファイルの方が大きい場合、R_jpeg_write_out_bufferが数回呼び出されることになります。 その都度記録メディアに追記することにより、1つのJPEGファイルが完成します。

2.3.4. マクロ定義

ここではヘッダファイルr_compress_jpege.hに記述しているマクロ定義について示しています。

表2.6.エラーコード定義

定義	値	内容
COMPRESS_JPEGE_OK	0	正常終了
COMPRESS_JPEGE_ERROR_ARG	-1	引数エラー
COMPRESS_JPEGE_ERROR_WRITE	-2	書き込みエラー

2.4. ライブラリ関数

本節ではJPEGエンコードライブラリの各関数の詳細を示します。各関数詳細の読み方は以下のとおりです。

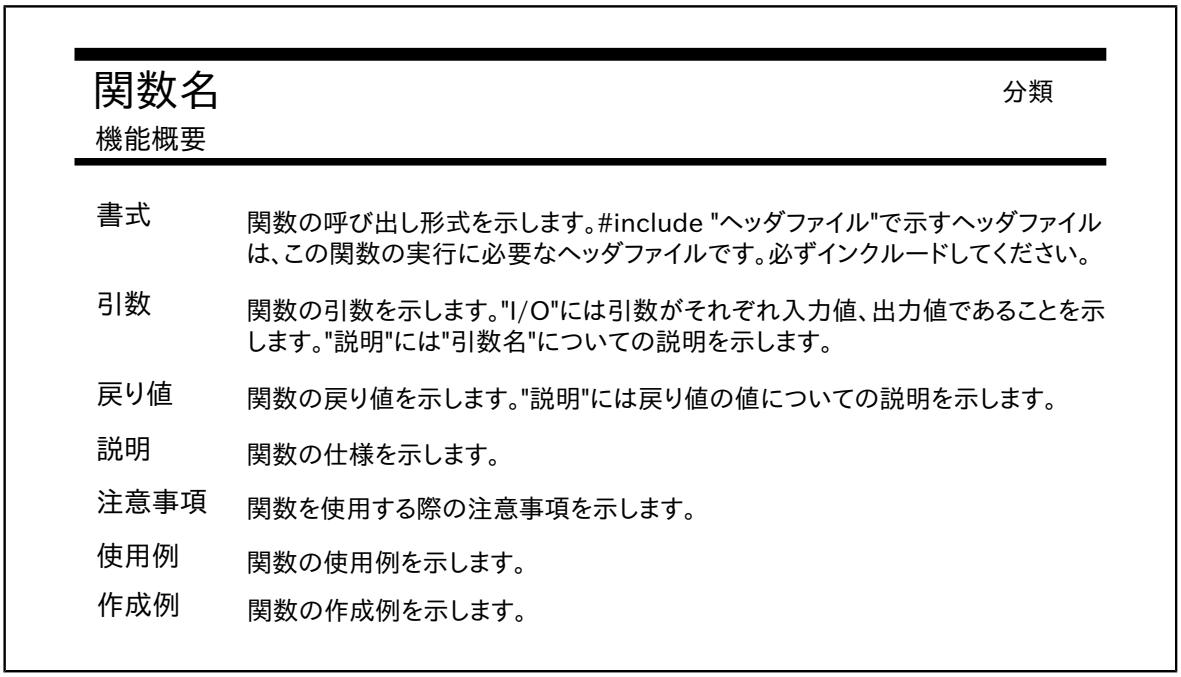

図2.3.ライブラリ関数詳細の見方

R_compress_jpeg

ライブラリ関数

— 画像のJPEG圧縮

書式

```
#include "r_compress_jpege.h"

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

引数

引数名	I/O	説明
setting	I/O	JPEGエンコード情報構造体へのポインタ

戻り値

戻り値	説明
0	正常終了
0以外	エラー

説明

本関数はsettingで指定された情報をもとに、画像データをJPEGファイルに圧縮します。settingの詳細は、 **表2.3.「構造体r_jpeg_encode_tのメンバ」** を参照ください。

settingのメンバwidth, heightで入力画像の大きさを指定します。 (以下settingのメンバ名のみ記載します)

qualityで品質を指定します。1(最低画質)～128(最高画質)になります。

restart_intervalでリスタートインターバルを指定します。 通常は0(リスタートマーカ無し)を指定します。 リスタートマーカを挿入したい場合は、1以上の値でリスタートマーカを挿入する間隔を指定してください。

input_addrで入力画像の先頭アドレスを指定します。 input_line_byteは、入力画像が置かれているメモリ空間の1ラインあたりのバイト数を指定します。 input_formatは、入力画像の形式を指定します。 入力可能な画像の形式は **表2.4.「入力画像の形式」** を参照ください。

たとえば、320x240のRGB888形式の画像の場合、input_line_byte = 320 x 3 = 960バイトになります。 また画像の一部320x240の画像の中央240x160部分をJPEGに圧縮する場合は、 input_addrは画素(40,30)が格納されているアドレス、width = 240、height = 160, input_line_byte = 960 と指定します。

生成するJPEGファイルはoutbuffで指定した出力バッファに出力します。 出力バッファのサイズをoutbuff_sizeで指定します。 出力バッファがフル、またはJPEGファイルの出力が完了した場合、 ユーザ定義関数R_jpeg_write_out_bufferがコールされます。 詳細はR_jpeg_write_out_bufferを参照ください。

output_formatでJPEGファイルの形式を指定します。 設定可能な値は **表2.5.「出力するJPEGファイルの形式」** を参照ください。

生成するJPEGファイルの構成は次の図を参照ください。

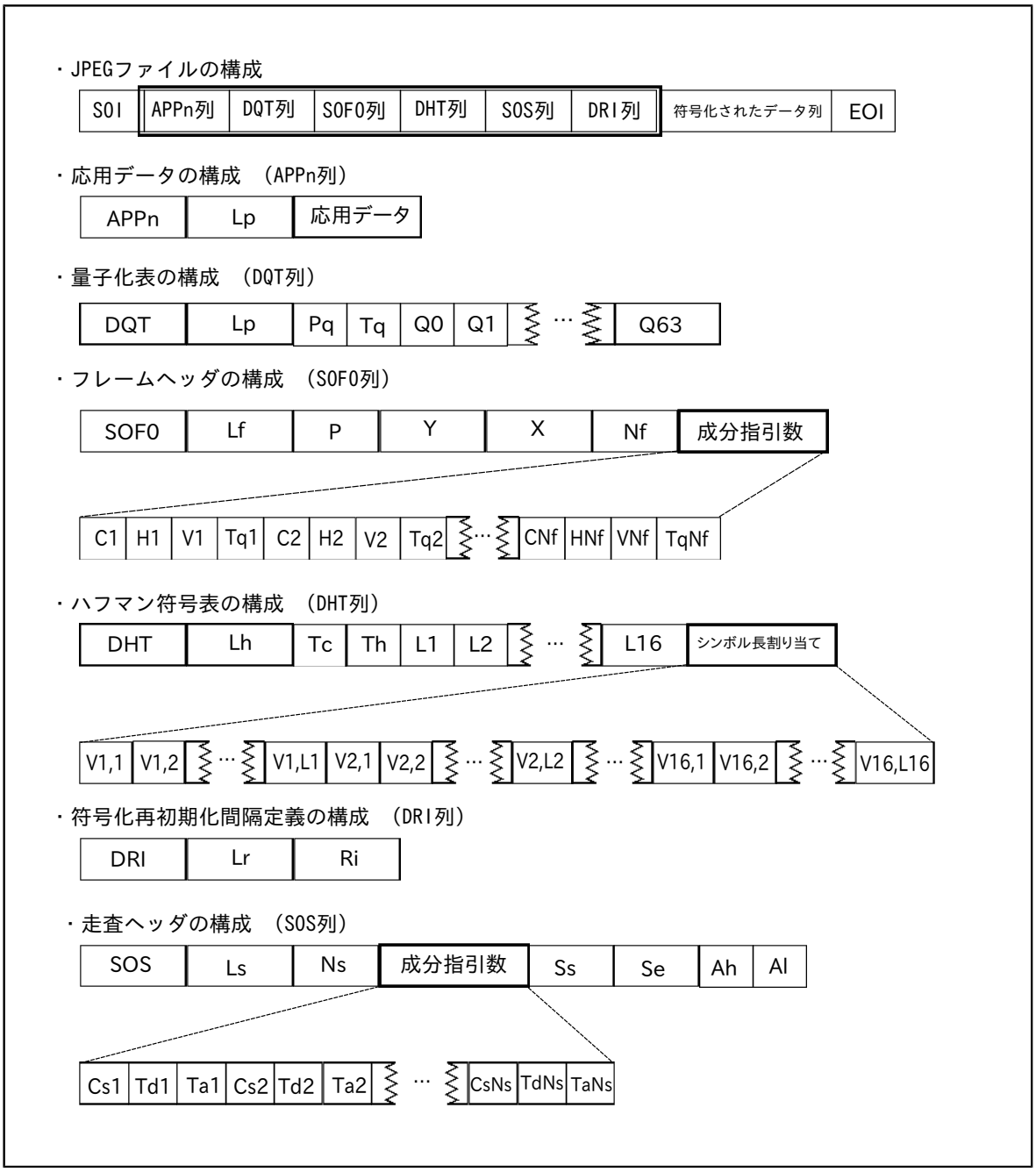

図2.4.JPEGファイルの構成

2.5. ユーザ定義関数

JPEGファイル圧縮ライブラリを使用する場合、ユーザ定義関数R_jpeg_write_out_bufferが必要になります。

R_jpeg_write_out_buffer

ユーザ定義関数

— 出力バッファのデータの出力

書式

```
#include "r_jpeg.h"

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

引数

引数名	I/O	説明
rest	I/O	出力バッファの残りデータ数

戻り値

出力バッファの先頭アドレスを返します

説明

出力バッファがフル、またはJPEGファイルの出力が完了した場合にコールされます。 生成されるJPEGファイルが出力バッファより大きい場合は、複数回に分けてコールされます。

本関数内で出力バッファの内容をメディア等に書き込んでください。 書き込みバイト数は、(出力バッファサイズ - rest)バイトになります。

書き込み後、出力バッファを初期化します。 本関数戻り値に出力バッファの先頭アドレスを返してください。 また引数restの値を出力バッファのサイズで再初期化してください。

作成例

```
#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. ソースコード情報

JPEGファイル圧縮ライブラリのソースコードの情報を示します。

表2.7.JPEGファイル圧縮ライブラリのファイル一覧

ファイル名	機能概要
r_compress_jpege.c	圧縮処理メイン
r_write_headers.c	JPEGヘッダ情報の書き込み
r_rgb2ycc.c	色空間の変換

表2.8.JPEGファイル圧縮ライブラリの関数一覧

関数名	機能概要
R_compress_jpege	JPEGファイルへの圧縮
_jpeg_init	初期設定
encode_jpeg422	YCbCr 4:2:2形式で圧縮するときのメイン処理
encode_jpeg420	YCbCr 4:2:0形式で圧縮するときのメイン処理
R_jpeg_dump_buffer	JPEGエンコードライブラリのためのユーザー定義関数
readSamplingMCU_RGB888 _YCbCr422	入力画像(RGB888形式)をYCbCr 4:2:2形式に変換して読み込む
readSamplingMCU_RGB888 _YCbCr420	入力画像(RGB888形式)をYCbCr 4:2:0形式に変換して読み込む
readSamplingMCU_RGB565 _YCbCr422	入力画像(RGB565形式)をYCbCr 4:2:2形式に変換して読み込む
readSamplingMCU_RGB565 _YCbCr420	入力画像(RGB565形式)をYCbCr 4:2:0形式に変換して読み込む
readSamplingMCU_YCbCr422	入力画像(YCbCr 4:2:2形式)を無変換で読み込む
_jpeg_write_header	JPEGヘッダ情報の書き込み
_jpeg_writeSOI	SOIの書き込み
_jpeg_writeSOF0	SOF0の書き込み
_jpeg_writeSOS	SOSの書き込み
_jpeg_writeDQT	DQTの書き込み
_jpeg_writeDHT	DHTの書き込み
_jpeg_writeAPP	APPの書き込み
r_rgb2ycc	1画素分のRGB888形式のデータをYCbCr形式に変換

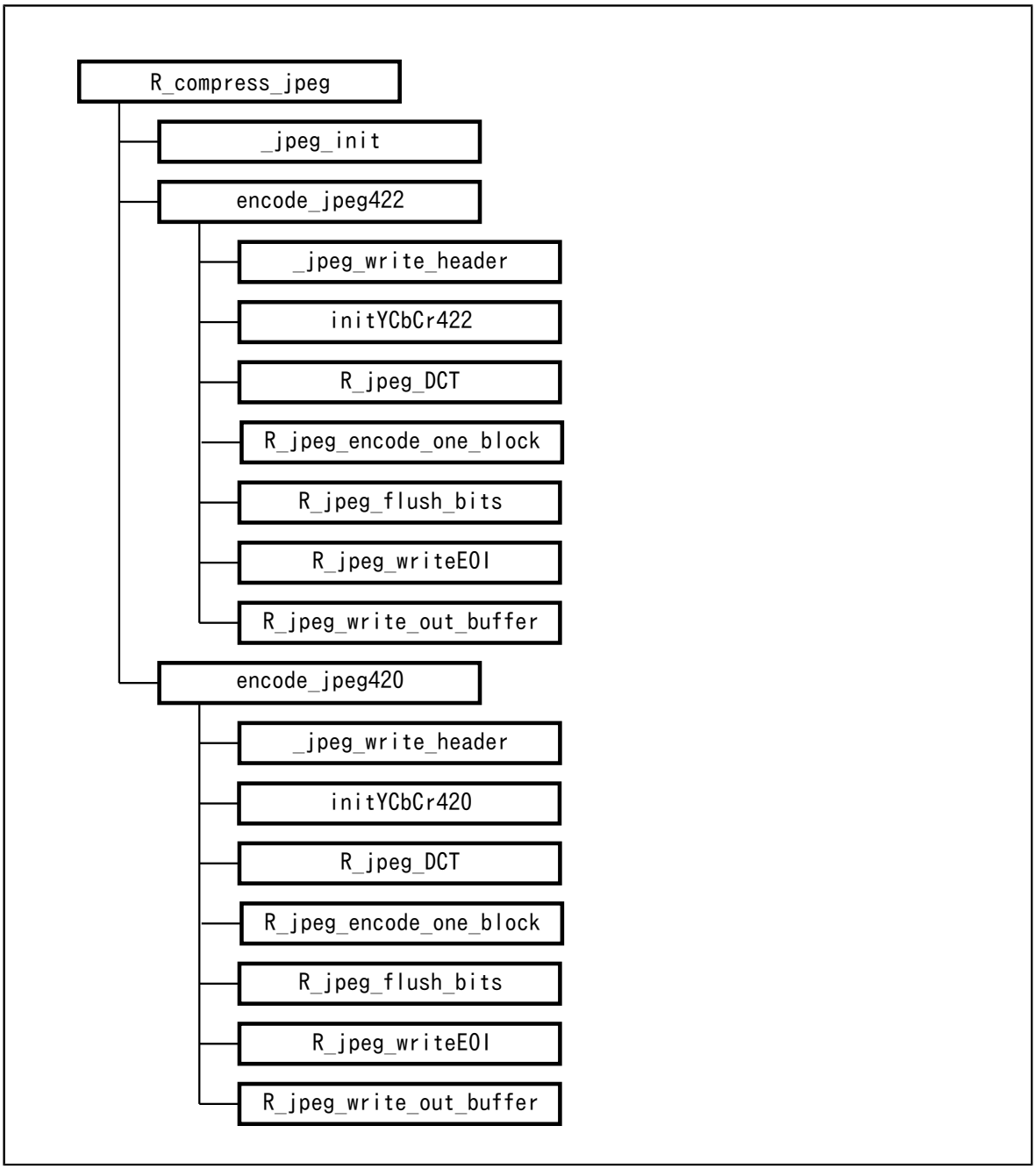

図2.5.関数のツリー構造

表2.9.確保している変数

変数名	用途
working	JPEGエンコードライブラリの構造体 _jpeg_working
encFMB	JPEGエンコードライブラリの構造体 _jpeg_enc_FMB
encSMB	JPEGエンコードライブラリの構造体 _jpeg_enc_SMB
MCU_data	DTC用ワークエリア (8x8+2 word)
Y_buf[]	1MCU分のY成分 ((8x8)x4 byte)
Cb_buf[]	1MCU分のCb成分 (8x8 byte)

変数名	用途
Cr_buf[]	1MCU分のCr成分 (8x8 byte)

3. JPEGエンコードライブラリ

本章では、基本演算を行うJPEGエンコードライブラリについて説明します。

3.1. 概要

JPEGエンコードライブラリは、JPEGファイルの圧縮に必要なハフマン符号化、量子化およびDCTを行うライブラリです。

JPEGエンコードライブラリでサポートするライブラリ関数を **表3.1.「ライブラリ関数一覧」** に、 ユーザ定義関数を **表3.2.「ユーザ定義関数一覧」** に示します。

表3.1.ライブラリ関数一覧

関数名	機能概要
R_jpeg_add_quant_table	量子化テーブルの登録
R_jpeg_DCT	DCTと量子化の実行
R_jpeg_encode_one_block	ハフマン符号化の実行
R_jpeg_writeDRI	DRIマーカの書き込み
R_jpeg_writeRST	RSTmマーカの書き込み
R_jpeg_writeEOI	EOIマーカの書き込み
R_jpeg_flush_bits	符号データの強制書込み

表3.2.ユーザ定義関数一覧

ユーザ定義関数名 (*)	機能
R_jpeg_write_out_buffer	JPEGファイルの出力

* 関数名は任意の名前に変更が可能です。

3.2. 構成

図3.1.「JPEGエンコードライブラリの構成」 では、元画像から抽出された8×8画素のY成分, Cb成分, Cr成分に対し JPEGエンコードライブラリを使用してJPEGファイルを生成する例を示しています。

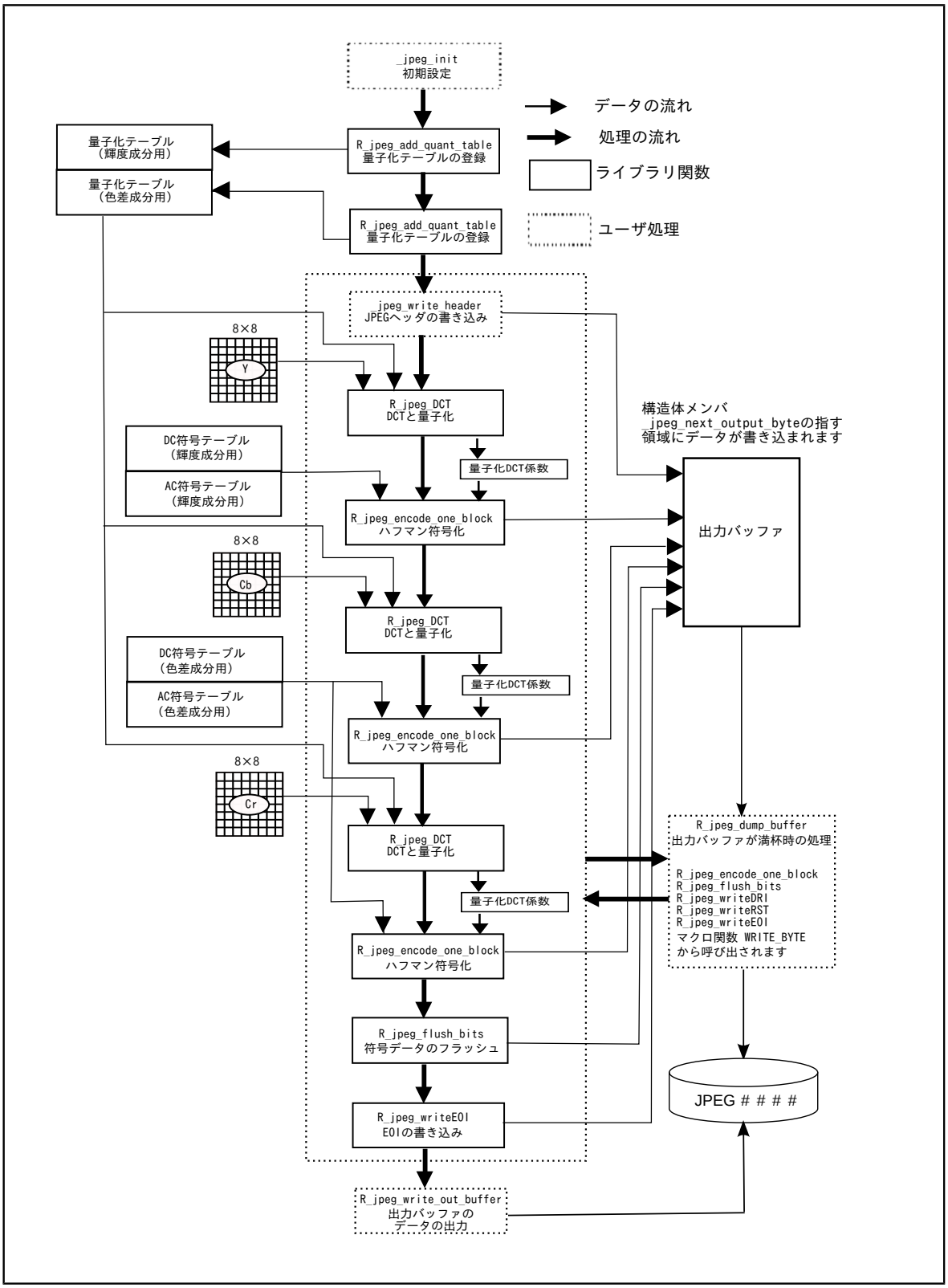

図3.1.JPEGエンコードライブラリの構成

3.3. データ構造

3.3.1. ライブラリ構造体

3.3.1.1. JPEGソフトウェアライブラリ環境変数

JPEGエンコーダでは、JPEGエンコードライブラリ内部で使用するデータのアクセスを容易にするためにライブラリ内部で使用するデータを一つの構造体により一括して管理しています。この構造体名を本マニュアルではJPEGソフトウェアライブラリ環境変数構造体と称します。

構造体の内容を 図3.2.「JPEGソフトウェアライブラリ環境変数構造体_jpeg_working」 に示します。

```
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 *);
};
```

図3.2.JPEGソフトウェアライブラリ環境変数構造体 _jpeg_working

JPEGソフトウェアライブラリ環境変数構造体の領域は、アプリケーションプログラムで確保してください。 圧縮時は"decode"のメンバについては、アプリケーションプログラムによる参照、設定の必要はありません。

JPEGファイル圧縮ライブラリを併用する場合は、この領域の確保、参照および設定はJPEGファイル圧縮ライブラリが行います。

3.3.1.2. JPEGエンコードライブラリ高速メモリ変数群

構造体_jpeg_enc_FMBには、JPEGエンコードライブラリの使用頻度の高い変数が定義されています。 構造体の内容を 図3.3.「JPEGエンコードライブラリ高速メモリ変数構造体 _jpeg_enc_FMB」 に示します。

```
#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];
};
```

図3.3.JPEGエンコードライブラリ高速メモリ変数構造体 _jpeg_enc_FMB

表3.3.「構造体_jpeg_enc_FMBのメンバ」 に各メンバの説明を示します。

表3.3.構造体_jpeg_enc_FMBのメンバ

構造体メンバ名	記号	機能概要
_jpeg_work[]	-	reserved
_jpeg_DCTqtbl[]	-	reserved
_jpeg_next_output_byte	-	符号化されたデータを次に書き込むアドレスを格納するポインタ
_jpeg_free_in_buffer	-	出力バッファに書き込める残量（バイト数）
_jpeg_cur_put_buffer	-	reserved
_jpeg_cur_put_bits	-	reserved
_jpeg_restart_interval	Ri	符号化再初期化間隔
_jpeg_thinning_mode	-	reserved
_jpeg_dc_size[]	-	reserved
_jpeg_ac_size[]	-	reserved

表中の記号は、参考文献で定義されたパラメータシンボルを意味します。

3.3.1.3. JPEGエンコードライブラリ高速メモリ定数群

構造体_jpeg_enc_FMCには、JPEGエンコードライブラリの使用頻度の高い定数が定義されています。 構造体の内容を 図3.4.「JPEGエンコードライブラリ高速メモリ定数構造体 _jpeg_enc_FMC」 に示します。

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

図3.4.JPEGエンコードライブラリ高速メモリ定数構造体 _jpeg_enc_FMC

_jpeg_enc_FMCの定数群は、JPEGエンコードライブラリの中で定義されており、 アプリケーションプログラムから_top_of_jpeg_enc_FMCのシンボルで参照することができます。 図3.5.「構造体_jpeg_enc_FMCの初期化」 にJPEGソフトウェアライブラリ環境変数構造体への登録方法を示します。

```
#include ”r_jpeg.h”

struct _jpeg_working working;        // JPEGライブリ環境変数構造体

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

図3.5.構造体_jpeg_enc_FMCの初期化

表3.4.「構造体_jpeg_enc_FMCのメンバ」 に各メンバの説明を示します。

表3.4.構造体_jpeg_enc_FMCのメンバ

構造体メンバ名	記号	機能概要
_jpeg_fmcData[]	-	reserved

表中の記号は、参考文献で定義されたパラメータシンボルを意味します。

3.3.1.4. JPEGエンコードライブラリ低速メモリ変数群

構造体_jpeg_enc_SMBには、JPEGエンコードライブラリの使用頻度の低い変数が定義されています。 構造体の内容を 図3.6.「JPEGエンコードライブラリ低速メモリ変数構造体 _jpeg_enc_SMB」 に示します。

```
#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;
};
```

図3.6.JPEGエンコードライブラリ低速メモリ変数構造体 _jpeg_enc_SMB

表3.5.「構造体_jpeg_enc_FMCのメンバ」 に各メンバの説明を示します。

表3.5.構造体_jpeg_enc_FMCのメンバ

構造体メンバ名	記号	機能概要
_jpeg_num_QTBL	-	量子化テーブルの数
_jpeg_QTBL[]	-	量子化テーブルの配列 R_jpeg_add_quant_table関数で登録。JPEGファイルのDQTブロックを出力する際のデータとして使用される。

構造体メンバ名	記号	機能概要
_jpeg_precision_QTBL[]	-	reserved
_jpeg_number_of_lines	Y	出力画像のライン数
_jpeg_line_length	X	1ライン当たりの画素数
struct component_info { uint8_t component_id[]; uint8_t hsample_ratio[]; uint8_t vsample_ratio[]; uint8_t quant_tbl_no[]; } component_info	-	色成分に関する情報 構造体の各メンバについて、別表に記述します。
struct frame_component_info { uint8_t component_id[]; uint8_t dc_tbl_no[]; uint8_t ac_tbl_no[]; } frame_component_info	-	フレームに関する情報 構造体の各メンバについて、別表に記述します。
_jpeg_frame_num_of_components	Ns	走査内の成分の数
_jpeg_write_DQT	-	量子化表の作成の有無 0:作成しない 1:作成する
_jpeg_write_DHT	-	ハフマン符号表の作成の有無 0:作成しない 1:作成する
dummy1	-	reserved
dummy2	-	reserved

表3.6.構造体component_infoのメンバ

構造体メンバ名	記号	機能概要
component_id[]	Ci	成分の識別子
hsample_ratio[]	Hi	成分の水平抽出比率
vsample_ratio[]	Vi	成分の垂直抽出比率
quant_tbl_no[]	Tqi	成分が用いる量子化テーブル番号

表3.7.構造体frame_component_infoのメンバ

構造体メンバ名	記号	機能概要
component_id[]	Csj	成分の識別子
dc_tbl_no[]	Tdj	成分が用いるハフマンテーブルのDCテーブル番号 (0,1) を格納する配列
ac_tbl_no[]	Taj	成分が用いるハフマンテーブルのACテーブル番号 (0,1) を格納する配列

表中の記号は、参考文献で定義されたパラメータシンボルを意味します。

3.3.1.5. JPEGエンコードライブラリ低速メモリ定数群

構造体_jpeg_enc_SMCには、JPEGエンコードライブラリの使用頻度の低い定数が定義されています。 構造体の内容を 図3.7.「JPEGエンコードライブラリ低速メモリ定数構造体_jpeg_enc_SMC」 に示します。

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

図3.7.JPEGエンコードライブラリ低速メモリ定数構造体_jpeg_enc_SMC

_jpeg_enc_SMCの定数群は、JPEGエンコードライブラリの中で定義されており、 アプリケーションプログラムから_top_of_jpeg_enc_SMCのシンボルで参照することができます。 図3.8.「構造体_jpeg_enc_SMCの初期化」 にJPEGソフトウェアライブラリ環境変数構造体への登録方法を示します。

```
#include ”r_jpeg.h”

struct _jpeg_working working;        // JPEGライブラリ環境変数構造体

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

図3.8.構造体_jpeg_enc_SMCの初期化

表3.8.「構造体_jpeg_enc_SMCのメンバ」 に各メンバの説明を示します。

表3.8.構造体_jpeg_enc_SMCのメンバ

構造体メンバ名	記号	機能概要
_jpeg_smcData[]	-	reserved

表中の記号は、参考文献で定義されたパラメータシンボルを意味します。

3.3.2. データ入力方法

画像データの入力はMCU単位で行います。

各色成分の横方向のデータは連続して配置されている必要があるため、必要に応じてデータを並べ替えてください。 たとえば、Y,Cb,Cr,Y,Cb,Cr,Y,Cb,Cr,... というデータの場合、Y,Y,Y,... Cb,Cb,Cb,... Cr,Cr,Cr,... と並べ替える必要があります。

画像データを最初に処理する関数 R_jpeg_DCTを参照してください。

3.3.3. データ出力方法

ライブラリが出力するデータはJPEGエンコードライブラリの出力バッファとして設定したRAM領域に格納します。 出力バッファには、JPEGファイルのヘッダ部分や符号データの一部または全部が格納されます。 出力バッファがフルになった場合、出力バッファの内容を記録メディアに書き込み、出力バッファをクリアします。 すべての画像データの圧縮が終わるまで繰り返します。

■出力バッファの管理方法

出力バッファは構造体_jpeg_enc_FMBの2つのメンバで管理されます。

```
uint8_t *_jpeg_next_output_byte;     // 次のデータの書込み位置
int32_t _jpeg_free_in_buffer;        // バッファの残量（バイト数）
```

出力バッファに1バイトのデータが書き込まれると、_jpeg_next_output_byteは1が加算され、_jpeg_free_in_bufferは1が減算されます。

上記2つのメンバは、JPEGエンコードライブラリを使用する前に初期化してください。

■出力バッファへのデータ格納順序

出力バッファに格納するデータの内容や順序はアプリケーションに依存します。 以下に格納順序の例を示します。

1. JPEGファイルのヘッダ部分（SOI、APPn、DQT、SOF0、DHT、DRI、SOSなどの符号データより前に記録されるマーカおよび部分列）の書き込みを行います。
2. Y,Cb,Crの成分のデータから量子化DCT係数を生成し、続いて符号データを生成し出力バッファに符号データの書き込みを行います。符号化再初期化間隔が有効な場合、R_jpeg_writeRST関数でRSTmマーカの書き込みを行います。
3. R_jpeg_flush_bits関数による符号データの強制書き込みを行います。
4. R_jpeg_writeEOI関数によるEOIマーカのコードの書き込みを行います。

■出力バッファがフルになった場合の処理

R_jpeg_encode_one_block、R_jpeg_flush_bits、R_jpeg_writeDRI、R_jpeg_writeRSTおよびR_jpeg_writeEOI 関数の実行中に出力バッファがフルになった場合、JPEGソフトウェアライブラリ環境変数構造体のメンバ enc_dump_funcに登録された関数が呼ばれます。

この関数にて、出力されたデータを記録メディア等へ書き込みしてください。 書き込み後、出力バッファの再初期化を行ってください。 再初期化を行わなかった場合、次に書き込みを行う関数でエラーが発生します。 ユーザ定義関数R_jpeg_dump_bufferをご参照ください。

3.3.4. マクロ定義

ここではヘッダファイルr_jpeg.hに記述しているマクロ定義について示しています。

表3.9.エラーコード定義

定義	値	内容
_JPEG_OK	0	正常終了

定義	値	内容
_JPEG_ERROR	-1	エラー終了

表3.10.定数定義

定義	値	内容
_JPEG_DCTSIZE	8	DCTサイズ
_JPEG_DCTSIZE2	64	DCTサイズの二乗
_JPEG_COMPONENT_NUM	3	成分数
_JPEG_HUFFVAL_SIZE	256	ハフマンコードの数
_JPEG_BITS_SIZE	17	ハフマンビットの数

表3.11.マクロ変数定義（高速メモリ変数群 _jpeg_enc_FMB）

定義	内容
_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)

表3.12.マクロ変数定義（低速メモリ変数群 _jpeg_enc_SMB）

定義	内容
_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)
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)

表3.13.データ書き込みマクロ定義

定義	内容
WRITE_BYTE(c, env)	1バイト長データの書き込み

3.4. ライブラリ関数

本節ではJPEGエンコードライブラリの各関数の詳細を示します。各関数詳細の読み方は以下のとおりです。

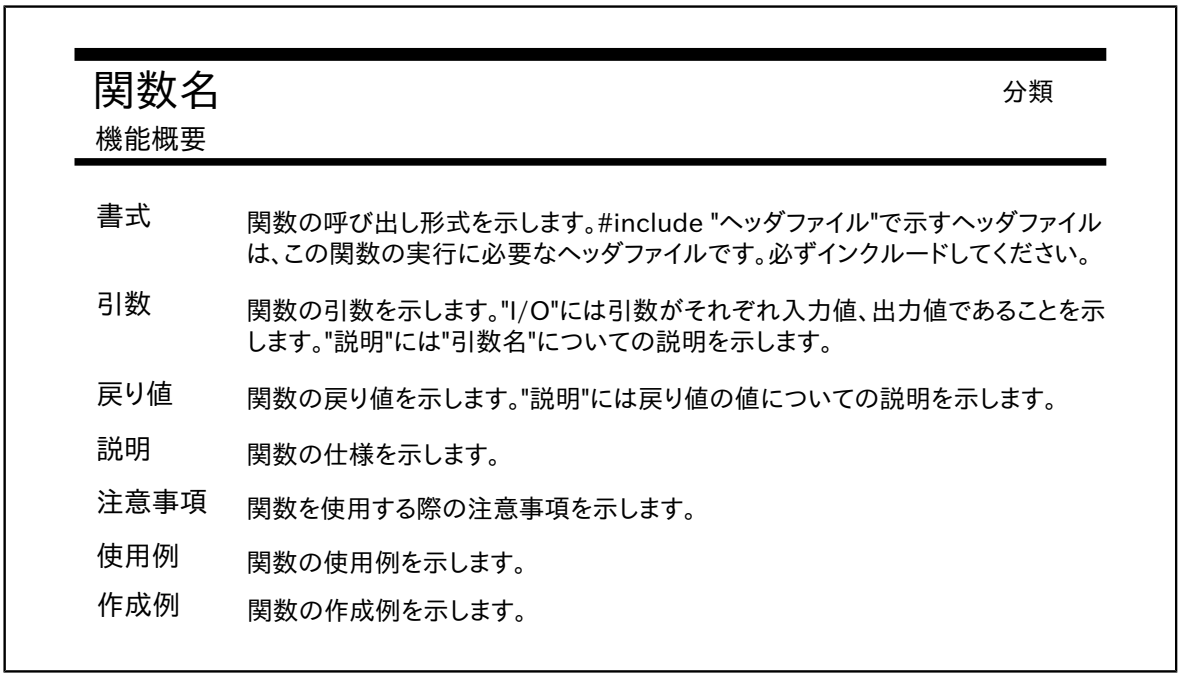

図3.9.ライブラリ関数詳細の見方

R_jpeg_add_quant_table

ライブラリ関数

— 量子化テーブルの登録

書式

```
#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 );
```

引数

引数名	I/O	説明
qtbl_no	I	量子化テーブル番号
basic_table	I	量子化テーブルへのポインタ
quality	I	量子化テーブルの精度の係数
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

なし

説明

本関数はwenvで指定されたJPEGソフトウェアライブラリ環境に対し、basic_tableで指定する量子化テーブルの内容を qualityの値を参照し、qtbl_noで指定する量子化テーブルに登録します。

qtbl_noには、登録する量子化テーブルの番号（0,1,2）を指定します。 qtbl_noに0,1,2以外の値が指定されたときの動作は不定です。

basic_tableには、本関数で登録する量子化テーブルの内容が格納された領域（int16_t型8×8の配列）へのポインタを指定します。

qualityは画質と画像圧縮率を変化させる引数で、1～128の範囲の値を指定します。 qualityの値が大きいほど高画質になりますが圧縮率は低下します。 qualityの値が小さいほど圧縮率は高くなりますが画質は劣化します。 quality < 1 または 128 < quality が指定された場合の本関数の動作は不定です。

注意

本関数による量子化テーブルの登録は、ユーザ定義関数_jpeg_write_headerの前に行ってください。

使用例

```
#include  ”r_jpeg.h”

struct _jpeg_working working;        // JPEGライブラリ環境変数構造体
```

```
//   Y成分用量子化テーブル
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
};

//   Cb,Cr成分用量子化テーブル
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)
{
          …
  //    量子化テーブルの登録
  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
ライブラリ関数

— DCTと量子化の実行

書式

```
#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 );
```

引数

引数名	I/O	説明
sample_data	I	処理の対象となるラインを指定するポインタ配列へのポインタ
start_col	I	ラインの先頭から処理対象ブロックへのオフセット（バイト数）
qtbl_no	I	量子化テーブルの番号
outptr	O	量子化されたDCT係数の格納領域へのポインタ
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

なし

説明

本関数はwenvで指定されたJPEGライブラリ環境に対し、 sample_dataとstart_colで指定する領域の成分のデータに対し、qtbl_noで指定する量子化テーブルを参照してDCTと量子化を行い、outptrで指定する領域に量子化DCT係数を格納します。

sample_dataには処理の対象となるラインを指定するポインタ配列へのポインタを指定します。すなわち、sample_data[0]、sample_data[1] ... それぞれが指定するラインが本関数の処理の対象の画像データとなります。

sample_data[0]、sample_data[1] ...では4バイト境界のアドレスを指定してください。 4バイト境界でないアドレスが指定された場合、本関数の動作は不定です。

start_colには、sample_data[]が指すラインの先頭から処理の対象となるブロックへのオフセット（バイト数）を指定します。

qtbl_noには、量子化テーブルの番号（0,1,2）を指定します。qtbl_noに0,1,2以外の値が指定されたときの動作は不定です。

outptrの指す領域（int16_t型8×8の配列）には、本関数によって得られる量子化DCT係数が格納されます。

図3.10.「R_jpeg_DCT関数による成分のデータのDCT処理」 に32×32画素の成分のデータを、DCTと量子化を実施する場合の例を示します。 例1ではブロック1が処理の対象となり、例2ではブロック6が処理の対象となります。

注意

本関数の実行の前に、R_jpeg_add_quant_table関数による量子化テーブルの登録が必要です。

使用例

```
#include  ”r_jpeg.h”

struct _jpeg_working    working;                // JPEGライブラリ環境変数構造体
static int16_t          MCU_data[64];           // MCUブロックデータ

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 初期化
          …
  // Cb_sample_data[8], Cb_start_col 初期化
          …
  // Cr_sample_data[8], Cr_start_col 初期化
          …
          //  Y0成分
          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成分
          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成分
          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成分
          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];
                  …
}
```

```
extern uint8_t *outptr[32];
int16_t block[64];


例1) R_jpeg_DCT(block, &outptr[0], 0, 0, &working);
例2) R_jpeg_DCT(block, &outptr[8], 8, 0, &working);
```

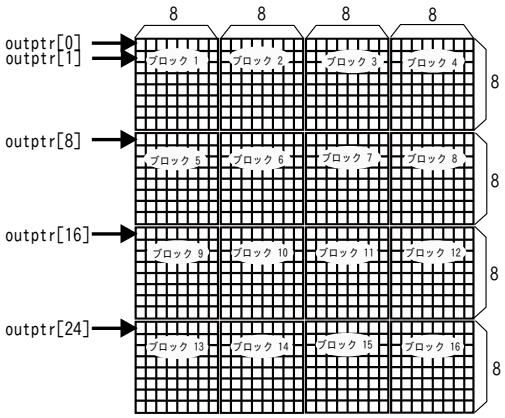

図3.10. R_jpeg_DCT関数による成分のデータのDCT処理

R_jpeg_encode_one_block

ライブラリ関数

— ハフマン符号化の実行

書式

```
#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 );
```

引数

引数名	I/O	説明
block	I	量子化されたDCT係数の格納領域へのポインタ
last_dc_val	I	前ブロックのDC係数
dc_tbl_no	I	DC係数ハフマン符号テーブルのテーブル番号
ac_tbl_no	I	AC係数ハフマン符号テーブルのテーブル番号
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

戻り値	説明
0	正常終了
0以外	エラー

説明

本関数は、wenvで指定されたJPEGソフトウェアライブラリ環境に対し、blockの指す領域に格納された量子化DCT係数をハフマン符号化し、符号化されたデータを出力バッファに格納します。

blockには、処理の対象となる量子化されたDCT係数が格納された領域（int16_t型8×8の配列）へのポインタを指定します。

last_dc_valには、前ブロックのDC係数を指定します。

dc_tbl_noには、DC係数のハフマン符号テーブルの番号（0,1）を指定します。 処理の対象がY成分の場合には0を指定してください。 処理の対象がCbまたはCr成分の場合には1を指定してください。 dc_tbl_noに0,1以外の値が指定されたときの動作は不定です。

ac_tbl_noには、AC係数のハフマン符号テーブルの番号（0,1）を指定します。 処理の対象がY成分の場合には0を指定してください。 処理の対象がCbまたはCr成分の場合には1を指定してください。 ac_tbl_noに0,1以外の値が指定されたときの動作は不定です。

注意

本関数の実行中に出力バッファがフルになった場合、ユーザ定義関数R_jpeg_dump_bufferが呼び出され出力バッファを初期化します。 R_jpeg_dump_buffer関数の実行終了後、本関数の実行

が継続されます。 R_jpeg_dump_buffer関数の仕様については、 R_jpeg_write_out_bufferをご参照ください。

使用例

R_jpeg_DCT関数の使用例をご参照ください。

R_jpeg_writeDRI

ライブラリ関数

— DRIマーカの書き込み

書式

```
#include "r_jpeg.h"

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

引数

引数名	I/O	説明
Ri	I	符号化再初期化間隔
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

戻り値	説明
0	正常終了
0以外	エラー

説明

本関数はwenvで指定されたJPEGソフトウェアライブラリ環境に対し、符号化再初期化間隔マーカDRIおよびRiで指定する符号化再初期化間隔を出力バッファに書き込みます。

Riには符号化再初期化間隔を指定します。

注意

本関数はR_jpeg_writeRST関数との組み合せでご使用ください。 符号化再初期化間隔に0を設定した場合、R_jpeg_writeRST関数による符号化再初期化マーカRSTmの書込みを行うことができません。

使用例

```
#include  ”r_jpeg.h”

struct _jpeg_working     working;          // JPEGライブラリ環境変数構造体

void sample(void)
{
        …
    int restart_interval;
        …

    restart_interval = 8;                  // 符号化再初期化間隔 8 を指定
```

```
    (working.encFMB)->_jpeg_restart_interval = restart_interval;
        …

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

}
```

R_jpeg_writeRST

ライブラリ関数

— RSTmマーカの書き込み

書式

```
#include "r_jpeg.h"

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

引数

引数名	I/O	説明
m	I	符号化再初期化間隔番号
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

戻り値	説明
0	正常終了
0以外	エラー

説明

本関数は、wenvで指定されたJPEGソフトウェアライブラリ環境に対し、符号化再初期化マーカRSTおよびmで指定する符号化再初期化間隔番号を出力バッファに書き込みます。

符号化再初期化間隔番号mには、0～7の値を繰り返すモジュロ8を指定します。

注意

本関数はR_jpeg_writeDRI関数との組み合わせでご使用ください。

本関数はR_jpeg_writeDRI関数により符号化再初期化間隔定義マーカDRIの書込みを行い、かつ引数Ri に1以上の値を設定した場合にのみ有効です。

使用例

```
#include  ”r_jpeg.h”

struct _jpeg_working  working;          // JPEGライブラリ環境変数構造体

void sample(void)
{
        …
  int16_t restart_interval;
  extern int16_t MCU_count;             // MCUカウンタ
  extern int16_t next_marker_num;       // 符号化再初期化間隔番号、初期値０
        …
```

```
    restart_interval = (working.encFMB)->_jpeg_restart_interval;
          …
    // RSTmマーカの書込み
    if(restart_interval) {
      if(MCU_count == 0) {
        R_jpeg_writeRST(next_marker_num,&working);  // 符号化再初期化マーカの書き込み
        next_marker_num=(next_marker_num+1)&7;      // 符号化再初期化間隔番号の更新
        MCU_count = restart_interval - 1;           // MCUカウンタを更新
      }
      else {
        --MCU_count;
      }
    }
          …
}
```

R_jpeg_writeEOI

ライブラリ関数

— EOIマーカの書き込み

書式

```
#include "r_jpeg.h"

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

引数

引数名	I/O	説明
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

戻り値	説明
0	正常終了
0以外	エラー

説明

本関数はwenvで指定されたJPEGソフトウェアライブラリ環境に対し、画像終了マーカEOIを出力バッファに書き込みます。

注意

本関数はすべての成分の符号化の完了後、R_jpeg_flush_bits関数による符号データの強制書込み後に呼び出してください。

使用例

```
#include  ”r_jpeg.h”

struct _jpeg_working  working;                  // JPEGライブリ環境変数構造体

void sample(void)
{

        …
  R_jpeg_flush_bits(&working);                  // 符号ビットの強制書込み
  R_jpeg_writeEOI(&working);                    // EOIマーカ書き込み
        …

}
```

R_jpeg_flush_bits

ライブラリ関数

— 符号データの強制書込み

書式

```
#include "r_jpeg.h"

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

引数

引数名	I/O	説明
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

戻り値	説明
0	正常終了
0以外	エラー

説明

本関数はwenvで指定されたJPEGソフトウェアライブラリ環境に対し、R_jpeg_encode_one_block関数が生成する符号データのうち、 R_jpeg_encode_one_block関数が保持した1バイトに満たない符号データをパディングして出力バッファに書き込みます。

使用例

R_jpeg_writeEOI関数の使用例をご参照ください。

3.5. ユーザ定義関数

JPEGエンコードライブラリを使用する場合、ユーザ定義関数R_jpeg_dump_bufferが必要になります。

R_jpeg_dump_buffer

ユーザ定義関数

— JPEGファイルの出力

書式

```
#include "r_jpeg.h"

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

引数

引数名	I/O	説明
wenv	I/O	JPEGソフトウェアライブラリ環境変数構造体へのポインタ

戻り値

なし

説明

本関数はwenvで指定されたJPEGソフトウェアライブラリ環境の環境に対し、出力バッファに格納された全データをアプリケーションで指定した領域に転送します。

本関数は、R_jpeg_encode_one_block, R_jpeg_flush_bits, R_jpeg_writeDRI, R_jpeg_writeRST, R_jpeg_writeEOI関数の下位関数です。 符号化データ格納時、出力バッファがフルになった時に上記の関数から呼び出されます。

本関数は、出力バッファのデータをバッファ領域の先頭から出力するようにします。出力先は、ターゲットシステムに応じます。

作成例

次の２つのプログラム記述例を示します。

1. 本関数のJPEGソフトウェアライブラリ環境変数構造体のメンバenc_dump_func への登録例
2. R_jpeg_dump_bufferから呼出すJEPGファイル出力のメイン関数作成例

1. 本関数のJPEGソフトウェアライブラリ環境変数構造体のメンバenc_dump_func への登録例

```
#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の作成例

出力バッファ_jpeg_buffer[ ]に格納されているデータを、_jpeg_file_topaddr で示される 先頭アドレスへ転送する場合のR_jpeg_dump_buffer 関数の作成例を示します。

```
#define OUTPUT_BUFFER_SIZE 2048   /* JPEG データ出力バッファサイズ */
 :
uint8_t _jpeg_buffer[OUTPUT_BUFFER_SIZE]; /* JPEG データ出力バッファ */
uint8_t *_jpeg_file_topaddr;      /* JPEG データファイルアドレス（JPEGデータ格納先）*/
uint16_t _jpeg_file_size;         /* JPEG データファイルサイズ */
 :
void R_jpeg_dump_buffer(void)
{
  struct _jpeg_enc_FMB *FMBbase;

  FMBbase = wenv->encFMB;
//-----------------------------------------------------
// ここで、出力バッファに格納されているJPEGデータを
// 記録メディアへの転送等の処理を行なう
//-----------------------------------------------------
  memcpy( _jpeg_file_topaddr+ _jpeg_file_size, _jpeg_buffer, OUTPUT_BUFFER_SIZE );

  /* JPEG ファイルサイズをコピーしたサイズ分インクリメントする */
  _jpeg_file_size += OUTPUT_BUFFER_SIZE;

//-----------------------------------------------------
// ライブラリ変数の初期化
//-----------------------------------------------------
  /* 出力バッファの空きサイズを再設定する(出力バッファEmpty) */
  _jpeg_free_in_buffer(FMBbase) = OUTPUT_BUFFER_SIZE;
  /* 出力バッファへの書出し位置をバッファ先頭に戻す */
  _jpeg_next_output_byte(FMBbase) = _jpeg_buffer;
}
```

JPEGエンコーダ
ユーザーズマニュアル

発行年月日	2013年 9月30日	Rev.1.00
	2013年 9月30日	Rev.1.00
発行	ルネサス エレクトロニクス株式会社	
	〒211-8668 神奈川県川崎市中原区下沼部 1753	

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口

http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社 〒100-0004 千代田区大手町2-6-2（日本ビル）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/

JPEGエンコーダ
ユーザーズマニュアル

ルネサス エレクトロニクス株式会社

R20UW0122JJ0100