User's Manual

JPEGデコーダ

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.1.03　2015.03

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デコーダは、JPEGファイルを伸張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デ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デ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デ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	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つのリスターと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デコーダの概要
JPEGデコーダの概要を説明しています。

■2章JPEGファイル伸張ライブラリ
JPEGファイル伸張ライブラリについて説明しています。

■3章JPEGデコードライブラリ
JPEG画像の伸張に必要な逆量子化と逆DCTなど基本演算を行うJPEGデコードライブラリについて説明しています。

補足説明

JPEGデコーダは、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デコーダの概要

本章では、JPEGデ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デコーダの特徴

1.1.1. アルゴリズム

JPEGデコーダは、「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3つの成分への伸張が可能

1.1.2. 機能概要

JPEGデコーダは、JPEGファイルの伸長を行うJPEGファイル伸張ライブラリと、基本演算を行うJPEGデコードライブラリの２つで構成されています。

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

■JPEGデコードライブラリ
JPEGデコードライブラリは、圧縮画像データに対してハフマン復号化、逆量子化、逆DCTを行います。 JPEGファイル伸張ライブラリと組み合わせて使用するか、JPEGファイルの伸張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。 JPEGデコードライブラリの詳細については**3章JPEGデコードライブラリ**をご参照ください。

図1.1.「 画像データの圧縮と伸張 」 にデータフローを示します。

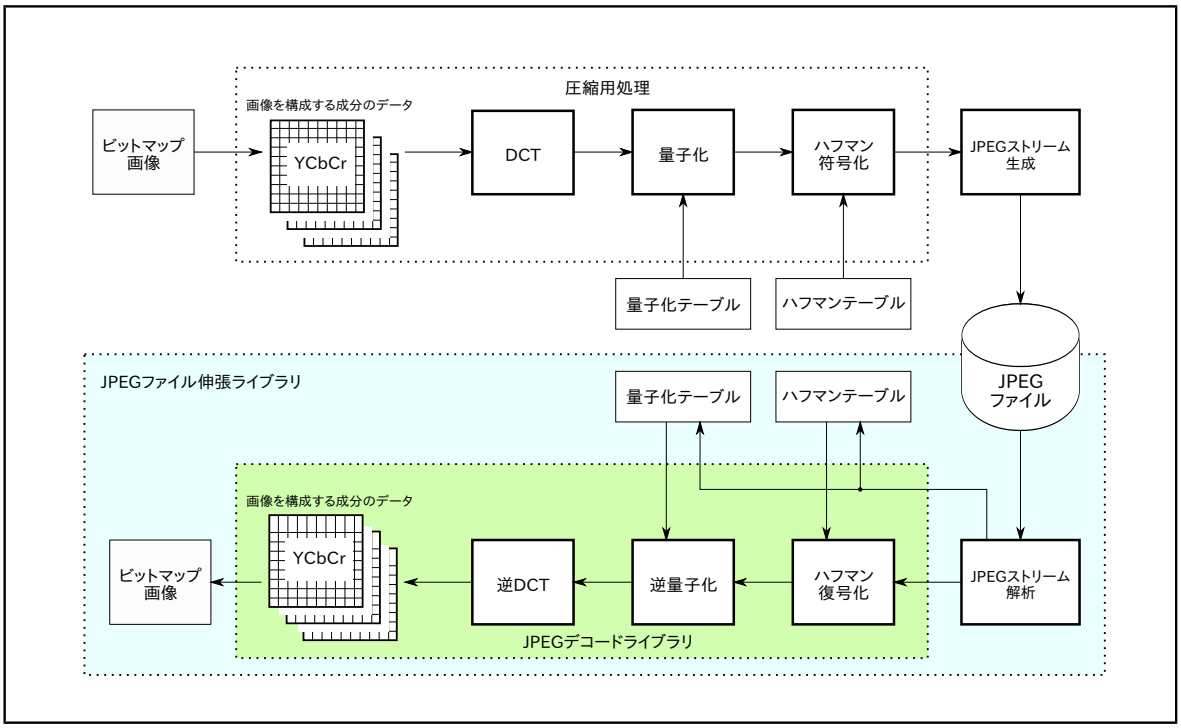

図1.1. 画像データの圧縮と伸張

表1.1.「JPEGファイル伸張ライブラリの仕様」 を示します。

表1.1.JPEGファイル伸張ライブラリの仕様

項目	仕様
対応フォーマット	JFIF Ver.1.1 準拠
色要素	Y,Cb,Cr
サンプル比	4:4:4 (1x1,1x1,1x1) 4:2:2 (2x1,1x1,1x1) 4:2:2垂直 (1x2,1x1,1x1) 4:2:0 (2x2,1x1,1x1)
プログレッシブ	非対応（伸張できません）
Exif	非対応（無視します）
出力フォーマット	RGB565 (16bit color)
クリッピング	非対応（すべて伸張します）

JPEGファイル伸張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ファイル伸張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ファイル伸張ライブラリの「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デ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。

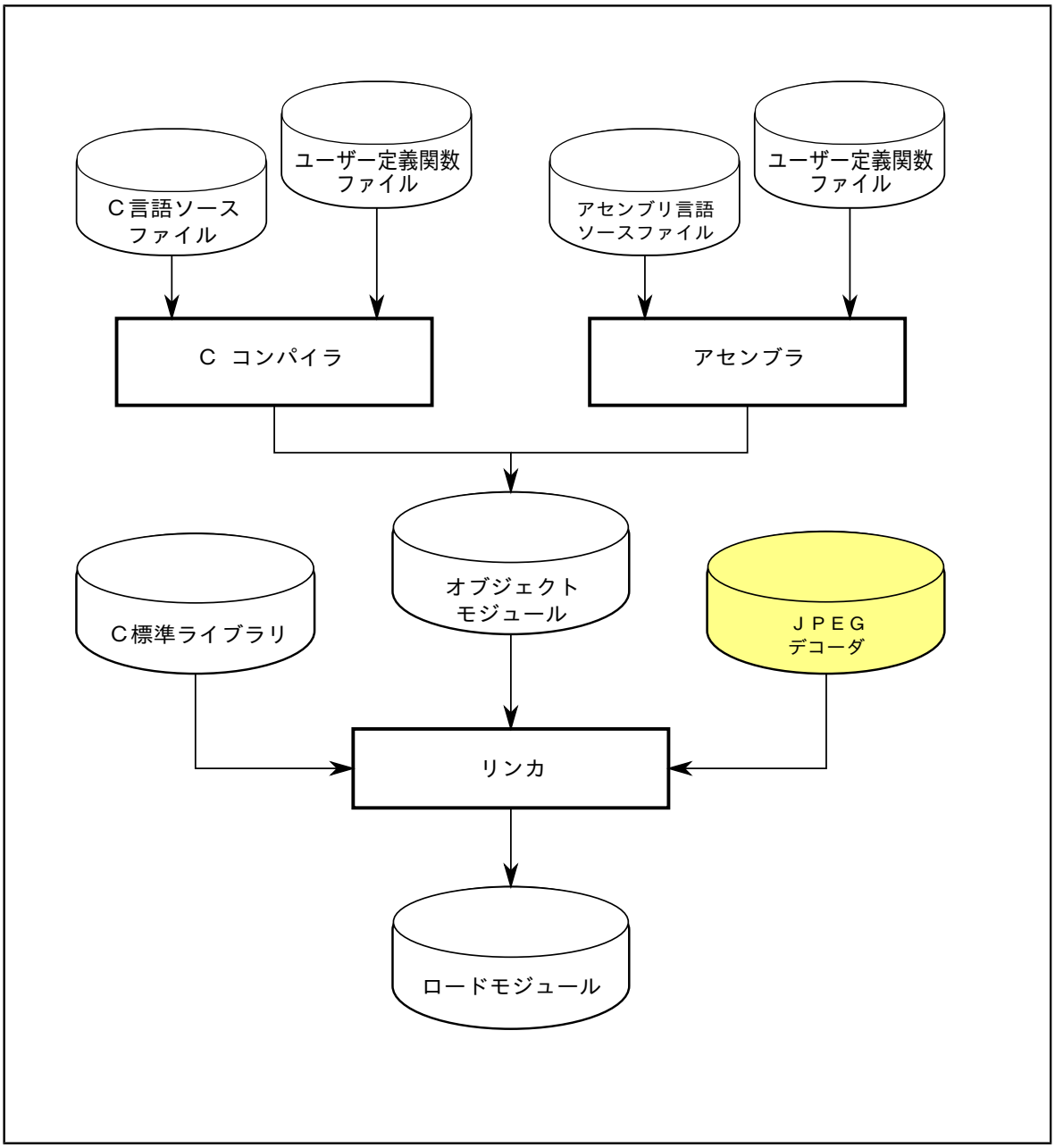

図1.2.アプリケーションプログラムの開発フロー

2. JPEGファイル伸張ライブラリ

本章では、JPEGファイル伸張ライブラリについて説明します。

2.1. 概要

JPEGファイル伸張ライブラリは、JPEGファイルをビットマップ画像に伸張するためのライブラリです。

JPEGファイル伸張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」に示します。

表2.1.ライブラリ関数一覧

関数名	機能概要
R_init_jpeg	JPEGデコーダの初期設定
R_expand_jpeg	JPEGファイルの伸張
R_get_info_jpeg	JPEGファイルの画像サイズの取得

2.2. 構成

図2.1.「JPEGファイル伸張ライブラリの構成」　を示します。

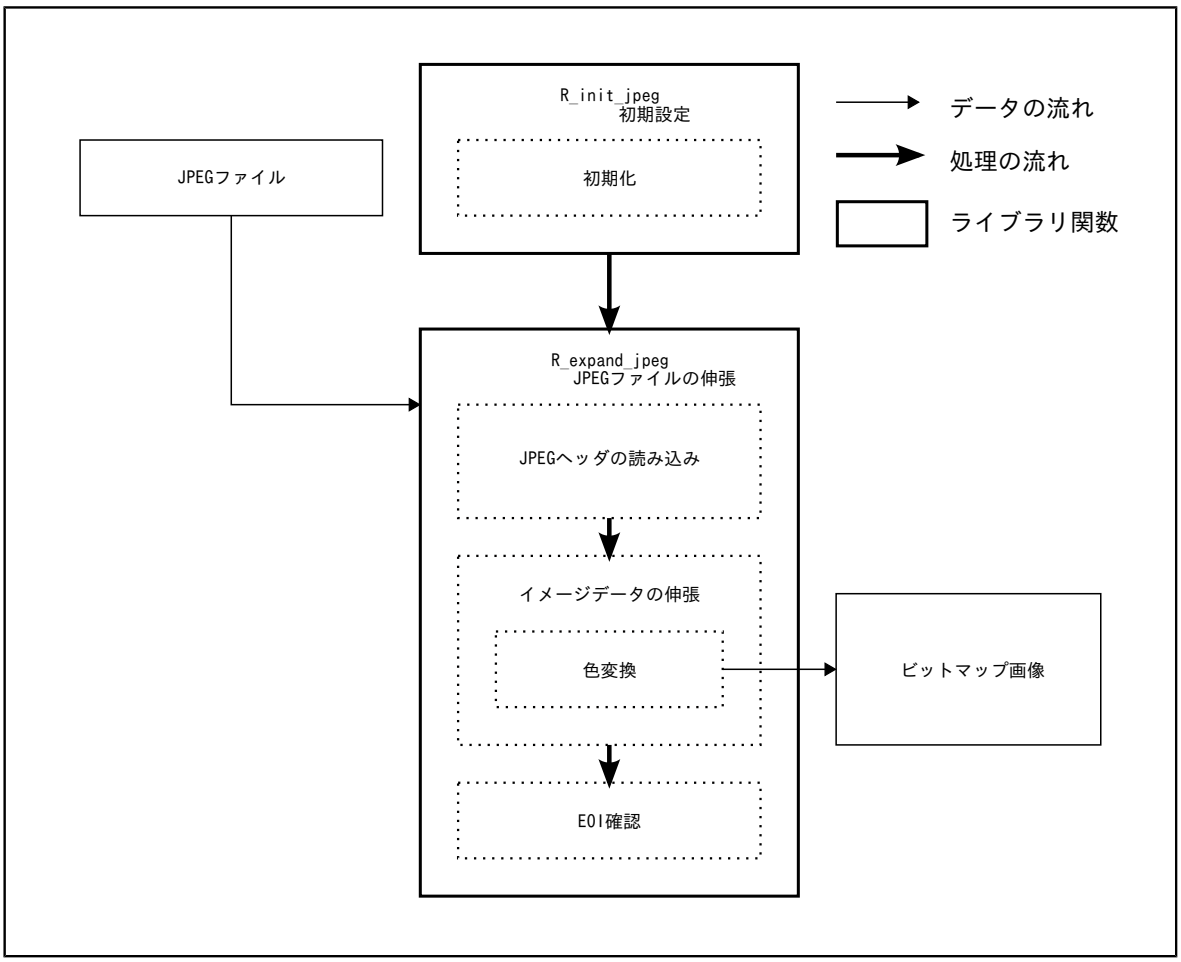

図2.1.JPEGファイル伸張ライブラリの構成

2.3. データ構造

JPEGファイル伸張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ファイル伸張ライブラリは、JPEGデ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ファイルのデコードに必要な変数は全てJPEGファイル伸張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。

2.3.2. データ入力方法

入力するデータは、JPEGファイル伸張ライブラリの関数R_expand_jpegの引数で指定します。JPEGデータを格納している領域の先頭アドレスをinputに、JPEGデータのサイズをfsizeで指定します。

これらの情報はJPEGデコードライブラリの入力バッファの初期値として設定し管理されます。入力バッファについては、**3章JPEGデコードライブラリ**の**「データ入力方法」**を参照してください。

2.3.3. マクロ定義

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

表2.2.エラーコード定義

定義	値	内容
EXPAND_JPEGD_OK	0	正常終了
EXPAND_JPEGD_ERROR_HEADER	-1	ヘッダ解析エラー
EXPAND_JPEGD_ERROR_DECODE	-2	伸張エラー
EXPAND_JPEGD_NOT_SUPPORT	-3	サポート外
EXPAND_JPEGD_ERROR_RST	-4	RST検出エラー
EXPAND_JPEGD_ERROR_SOI	-5	SOI検出エラー
EXPAND_JPEGD_ERROR_EOI	-6	EOI検出エラー

2.4. ライブラリ関数

本節ではJPEGファイル伸張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。

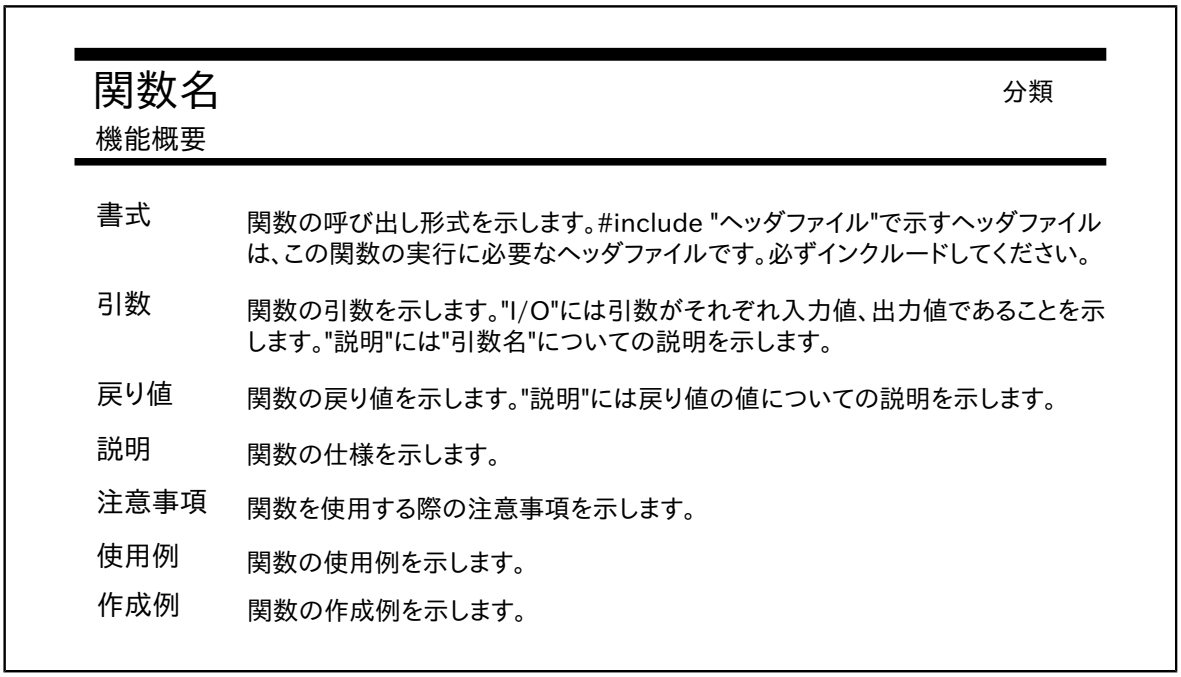

図2.2.ライブラリ関数詳細の見方

R_init_jpeg

ライブラリ関数

— JPEGファイル伸張ライブラリの初期設定

書式

```
#include "r_expand_jpegd.h"

void R_init_jpeg (
                void  );
```

引数

なし

戻り値

なし

説明

JPEGファイル伸張ライブラリを初期化します。

R_expand_jpeg

ライブラリ関数

— JPEGファイルの伸張

書式

```
#include "r_expand_jpegd.h"

int16_t R_expand_jpeg (
                uint8_t *input ,
                uint32_t fsize ,
                uint16_t *outptr ,
                uint32_t offset );
```

引数

引数名	I/O	説明
input	I	入力データの先頭アドレス
fsize	I	入力データのサイズ
outptr	O	出力先のアドレス
offset	I	出力先の1ライン辺りの画素数

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、inputで指定されたJPEGファイルを伸張し、RGB565形式(1ピクセル辺り2バイト)でoutputへ出力します。

入力するデータのサイズはfsizeで指定します。入力するJPEGデータは連続してメモリに配置されている必要があります。

offsetには出力先の1ライン辺りの画素数を指定します。 たとえば、320×240(横×縦)ピクセルのフレームバッファに出力する場合は320を指定します。 図2.3.「ビットマップの出力イメージ」 に出力イメージを示します。

JPEG以外の画像形式やフォーマットに問題があった場合、処理を中断しエラーを返します。

注意事項

出力先には伸張後の画像サイズ以上の領域を確保してください。 R_get_info_jpegを用いることで伸張後の画像サイズを取得することが出来ます。

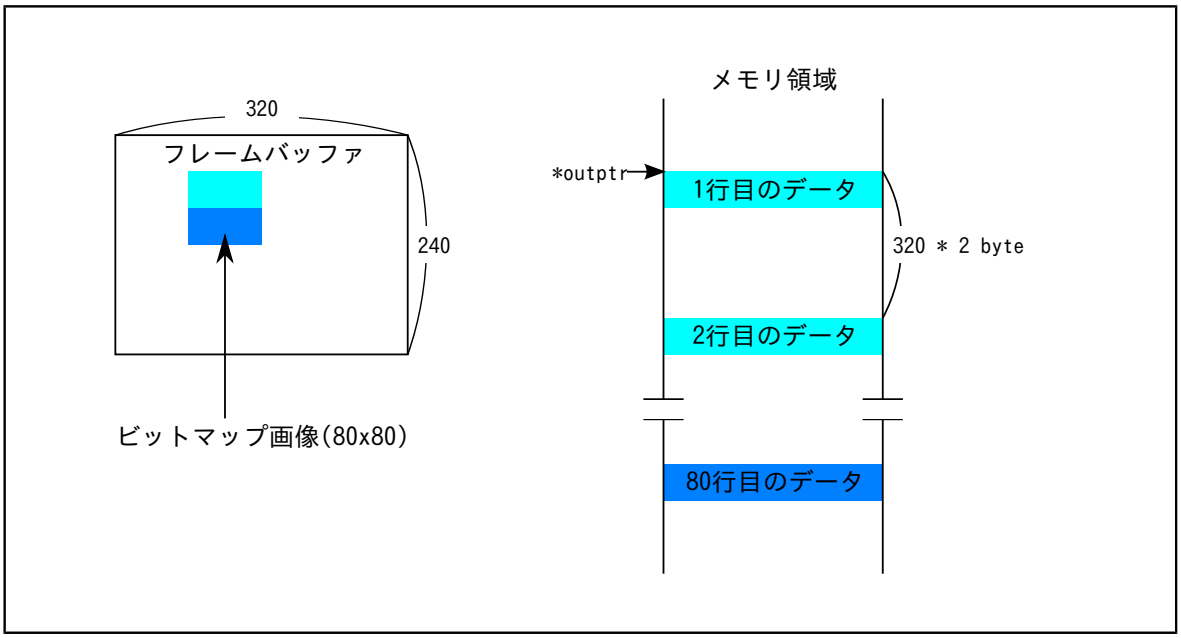

図2.3.ビットマップの出力イメージ

R_get_info_jpeg

ライブラリ関数

— JPEGファイルの画像サイズの取得

書式

```
#include "r_expand_jpegd.h"

int16_t R_get_info_jpeg (
                uint8_t *input ,
                uint32_t fsize ,
                uint16_t *w ,
                uint16_t *h );
```

引数

引数名	I/O	説明
input	I	入力データの先頭アドレス
fsize	I	入力データのサイズ
w	O	横方向の画素数の格納先
h	O	縦方向の画素数の格納先

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、inputで指定された画像ファイルがJPEG画像であるか判断し、JPEG画像であれば画像のサイズをw, hに出力します。

JPEG以外の画像形式やフォーマットに問題があった場合、処理を中断しエラーを返します。 伸張するJPEGファイルの画像サイズが事前に分かっている場合は、本関数を使用する必要はありません。

2.5. ユーザ定義関数

JPEGファイル伸張ライブラリにユーザ定義関数はありません。

2.6. ソースコード情報

JPEGファイル伸張ライブラリのソースコードの情報を示します。

表2.3.JPEGファイル伸張ライブラリのファイル一覧

ファイル名	機能概要
r_C_read_headers.c	ヘッダ解析
r_expand.c	JPEGファイルの伸張処理
r_expand_jpegd_version.c	バージョン情報
r_jpeg_read_input.c	JPEGファイルの入力（ダミー関数）
r_ycc2rgb.c	YCC->RGB変換
r_expand_jpegd_version.c	バージョン情報
r_jpegd.h	JPEGデコードライブラリのヘッダファイル
r_jpeg_maker.h	マーカ定義
r_rgb2short.h	RGB565変換のマクロ定義
r_expand_jpegd.h	JPEGファイル伸張ライブラリのヘッダファイル
r_stdint.h	型定義ヘッダファイル
r_mw_version.h	バージョン情報ヘッダファイル

表2.4.JPEGファイル伸張ライブラリの関数一覧

関数名	機能概要
R_init_jpeg	ライブラリの初期化
R_expand_jpeg	JPEG伸張のメイン関数
R_get_info_jpeg	画像情報の取得
decode444	YCbCr 4:4:4 のデコード処理
decode422	YCbCr 4:2:2 のデコード処理
decode422v	YCbCr 4:2:2垂直 のデコード処理
decode420	YCbCr 4:2:0 のデコード処理
init_ycc444_outptr	YCbCr 4:4:4 の各成分の出力ポインタの初期化
init_ycc4xx_outptr	YCbCr 4:4:4以外 の各成分の出力ポインタの初期化
init_last_outptr	最終画像の出力ポインタの初期化
_restart_marker	リスタートマーカの判定
_jpeg_open	JPEGファイルの情報登録
_jpeg_read_header	ヘッダの解析
_jpeg_readMarkers	マーカを検出し各処理を呼び出す
_jpeg_readSOF0	SOF0処理

関数名	機能概要
_jpeg_readSOS	SOS処理
_jpeg_readAPP0	APP0処理
_jpeg_readAPP14	APP14処理（実処理なし）
_jpeg_readDHT	DHT処理
_jpeg_readDQT	DQT処理
_jpeg_skipMarkerSegment	マーカ内データの読み飛ばし ヘッダ解析は継続
_jpeg_noSupportMarkers	サポート外マーカ検出 ヘッダ解析はエラー終了
_jpeg_readDRI	DRI処理
_jpeg_checkSOI	SOIチェック
_jpeg_checkEOI	EOIチェック
_jpeg_checkTableConsistency	テーブルの整合性チェック
ycc444_422v_rgb565	YCbCr(4:4:4, 4:2:2垂直)の横1ラインをRGB565で出力
ycc422_420_rgb565	YCbCr(4:2:2, 4:2:0)の横1ラインをRGB565で出力
ycc2rgb	1ピクセルのYCbCrデータをRGB(8bit x 3)に変換する
R_jpeg_read_input	JPEGファイルの入力（ダミー関数）

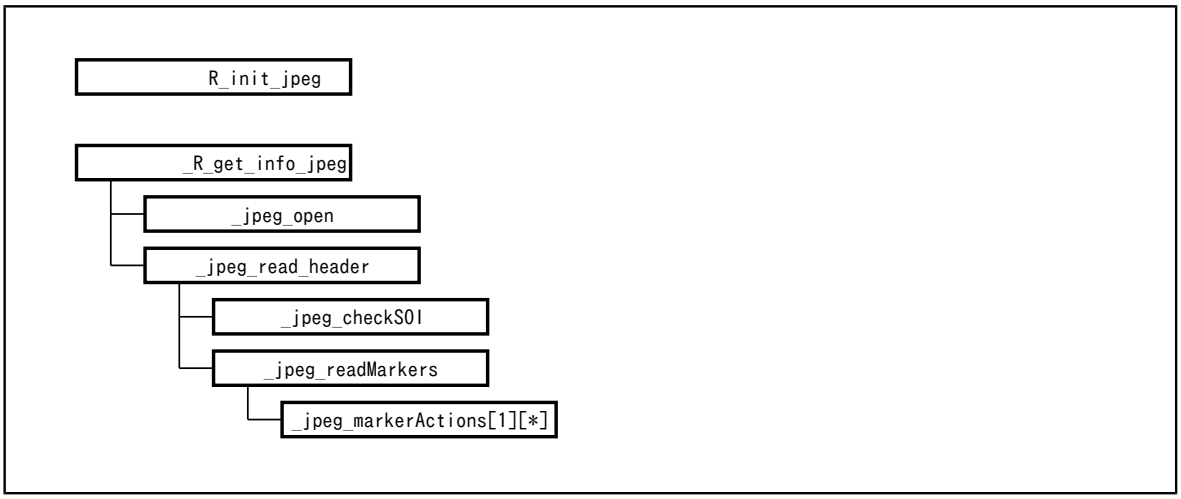

図2.4.関数のツリー構造 (1/2)

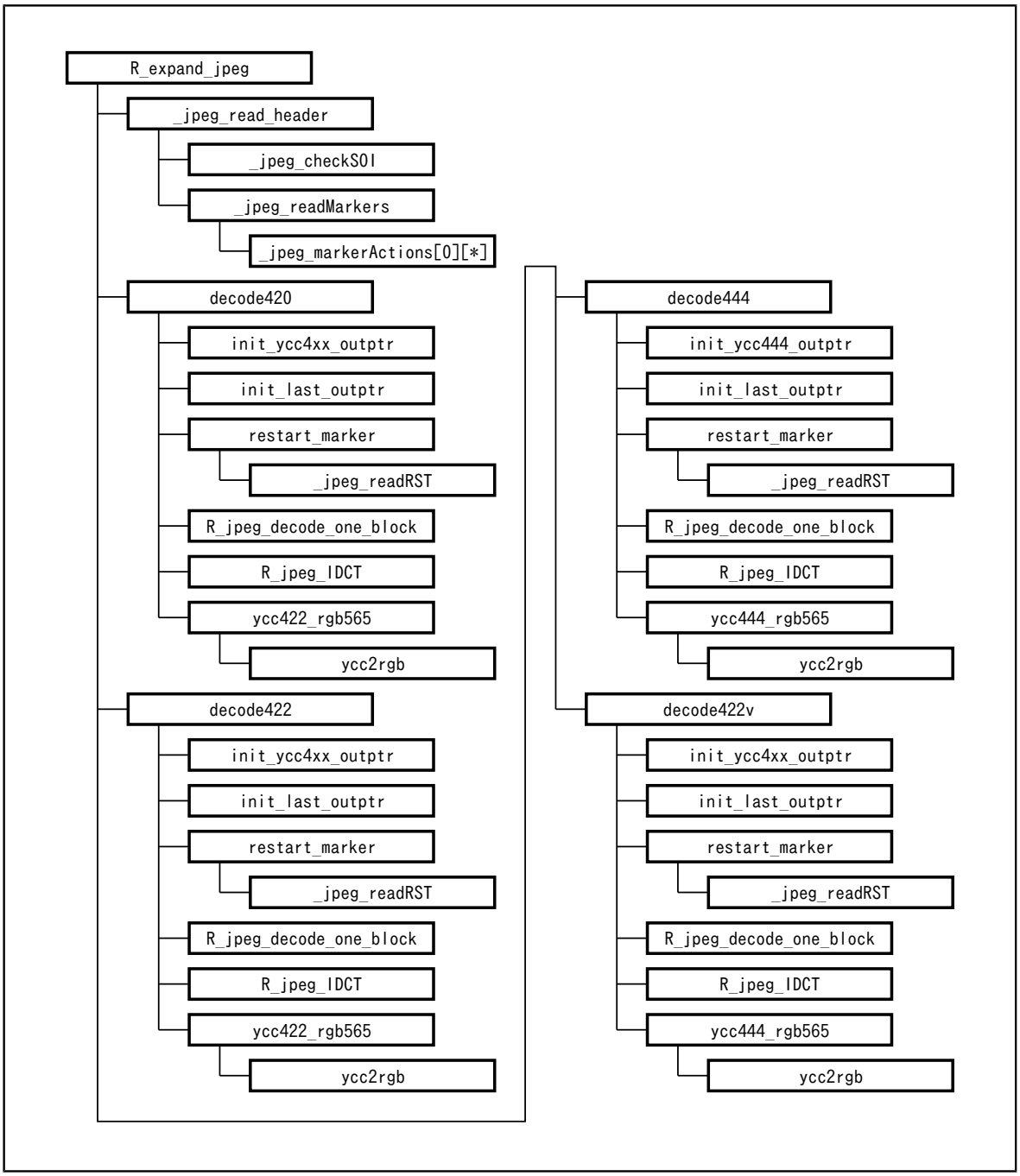

図2.5.関数のツリー構造 (2/2)

表2.5.確保している変数

変数名	用途
working	JPEGデコードライブラリの構造体 _jpeg_working
FMB	JPEGデコードライブラリの構造体 _jpeg_dec_FMB
SMB	JPEGデコードライブラリの構造体 _jpeg_dec_SMB

変数名	用途
align.mem.dtc_work[] (DTC_WORK[])	IDTC用ワークエリア (8x8+2 word)
align.mem.Y[]	Y成分の復号結果 ((8x8)x4 byte)
align.mem.Cb[]	Cb成分の復号結果 (8x8 byte)
align.mem.Cr[]	Cr成分の復号結果 (8x8 byte)
Y_last_dc_val	Y成分の最後のDC値
Cb_last_dc_val	Cb成分の最後のDC値
Cr_last_dc_val	Cr成分の最後のDC値
*Y_outptr[]	Y成分の出力先管理 (16 line)
*Cb_outptr[]	Cb成分の出力先管理 (8 line)
*Cr_outptr[]	Cr成分の出力先管理 (8 line)
*RGB_outptr[]	ビットマップの出力先管理 (16+1 line)
MCU_count	RST処理用のMCUカウント
next_restart_num	次に検出されるべきRSTの番号(0-7)

3. JPEGデコードライブラリ

本章では、基本演算を行うJPEGデコードライブラリについて説明します。

3.1. 概要

JPEGデコードライブラリは、JPEGファイルの伸張に必要なハフマン復号化、逆量子化および逆DCTを行うライブラリです。

JPEGデ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_make_huff_table	ハフマンテーブルの登録
R_jpeg_add_iquant_table	量子化テーブルの登録
R_jpeg_decode_one_block	ハフマン復号化
R_jpeg_IDCT	逆量子化と逆DCT
R_jpeg_readRST	リスタートマーカの検出

表3.2.ユーザ定義関数一覧

ユーザ定義関数名 (*)	機能
R_jpeg_read_input	JPEGファイルの入力

* 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デコードライブラリの構成」 では、JPEGファイルからJPEGデコードライブラリを使用して8×8画素のY成分, Cb成分, Cr成分を繰り返し復元する例を示しています。

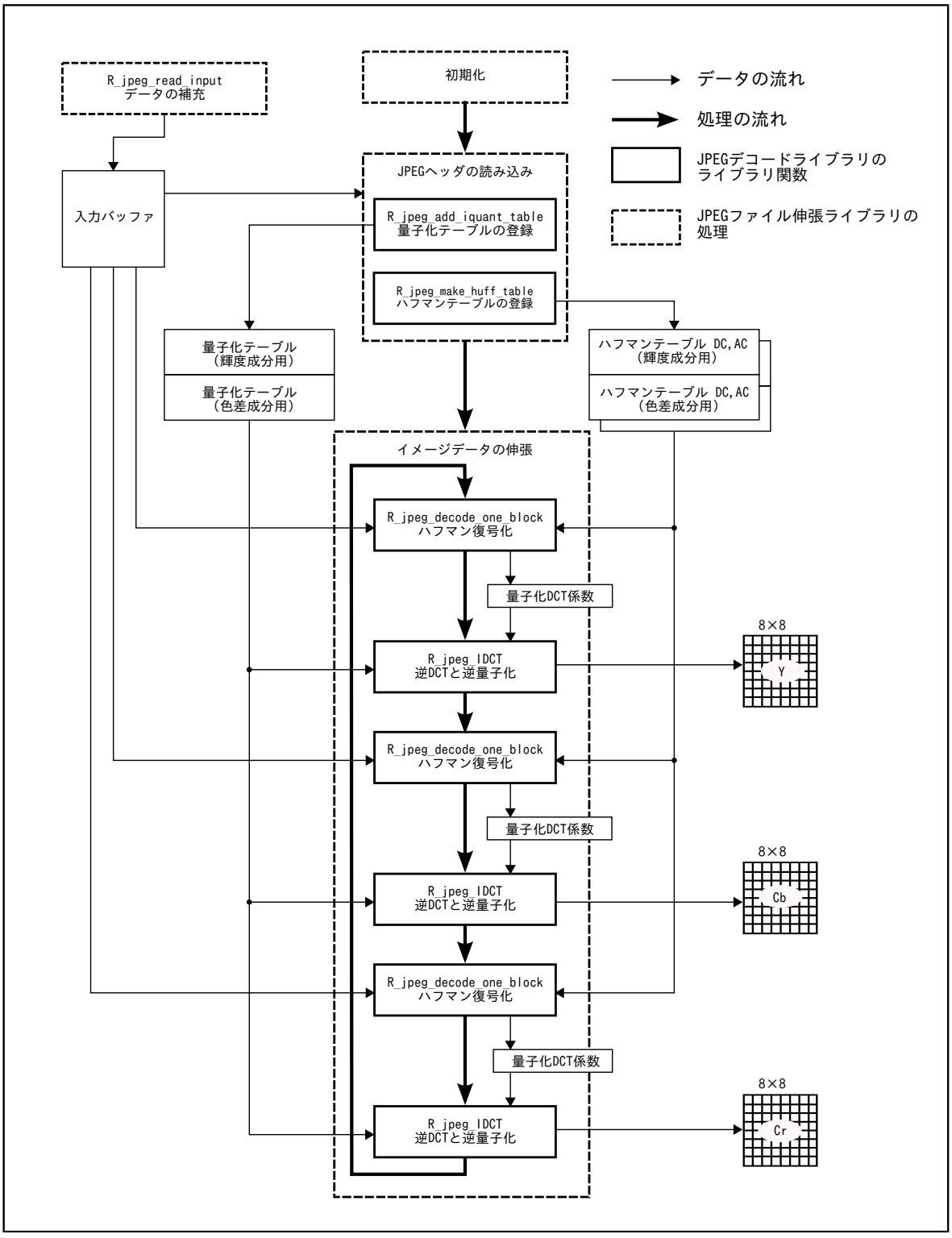

図3.1.JPEGデコードライブラリの構成

3.3. データ構造

JPEGデコードライブラリで定義するデータ構造について説明します。 変数の実体はJPEGデコードライブラリでは確保しません。

3.3.1. ライブラリ構造体

3.3.1.1. JPEGデコードライブラリ環境変数

JPEGデコーダでは、JPEGデ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デコード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デコードライブラリ環境変数構造体_jpeg_working」 に示します。

```
struct  _jpeg_working{
    /* reserved */
    void *encFMB;
    void *encFMC;
    void *encSMB;
    void *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デコードライブラリ環境変数構造体_jpeg_working

JPEGデコード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。 "reserved"記載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デコード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ファイル伸張ライブラリが行います。

3.3.1.2. JPEGデコードライブラリ高速メモリ変数群

構造体_jpeg_dec_FMBには、JPEGデ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デコードライブラリ高速メモリ変数構造体_jpeg_dec_FMB」 に示します。

```
#define _JPEG_HUFFVAL_SIZE              256

struct _jpeg_dec_FMB
{
    uint8_t fmb1[512];                  /* reserved */

    uint16_t _jpeg_work[_JPEG_HUFFVAL_SIZE+1];    /* reserved */
    int16_t _jpeg_restart_interval;
    uint8_t *_jpeg_next_read_byte;
    int32_t _jpeg_d_free_in_buffer;
    uint32_t _jpeg_cur_read_buffer;
    uint32_t _jpeg_cur_bits_offset;

    uint8_t fmb2[3632];                 /* reserved */
};
```

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

表3.3.「構造体_jpeg_dec_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_dec_FMBのメンバ

構造体メンバ名	記号	機能概要
fmb1[]	-	reserved
_jpeg_work[]	-	reserved
_jpeg_restart_interval	Ri	リスタートインターバル
*_jpeg_next_read_byte	-	次に読み込む、符号化されたデータを指すポインタ
_jpeg_d_free_in_buffer	-	入力バッファに読み込んだ符号化されたデータ（バイト数）
_jpeg_cur_read_buffer	-	reserved
_jpeg_cur_bits_offset	-	reserved
fmb2[]	-	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デコードライブラリ高速メモリ定数群

構造体_jpeg_dec_FMCには、JPEGデコードライブラリの参照頻度の高い定数が定義されています。構造体内容を 図3.4.「JPEGデコードライブラリ高速メモリ定数構造体_jpeg_dec_FMC」 に示します。

```
/* for RX */
struct _jpeg_dec_FMC
{
    uint8_t fmc[1284];                  /* reverved */
};


/* for SH */
struct _jpeg_dec_FMC
{
    uint8_t fmc[1240];                  /* reverved */
};
```

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

_jpeg_dec_FMCの定数群は、JPEGデ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_dec_FMCのシンボルで参照することができます。 JPEGデコードライブラリ環境変数構造体への登録方法を 図3.5.「構造体_jpeg_dec_FMCの初期化」に示します。

```
#include ”r_jpegd.h”

sample(void)
{
                    …
    working.decFMC = (struct _jpeg_dec_FMC *)_top_of_jpeg_dec_FMC;
                    …
}
```

図3.5.構造体_jpeg_dec_FMCの初期化

表3.4.「構造体_jpeg_dec_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_dec_FMCのメンバ

構造体メンバ名	記号	機能概要
fmc[]	-	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デコードライブラリ低速メモリ変数群

構造体_jpeg_dec_SMBには、JPEGデ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デコードライブラリ低速メモリ変数構造体_jpeg_dec_SMB」 に示します。

```
#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_dec_SMB
{
    int32_t _jpeg_d_num_QTBL;
    int32_t _jpeg_thinning_mode;
    uint16_t _jpeg_d_number_of_lines;
    uint16_t _jpeg_d_line_length;

    int16_t _jpeg_X_density;
    int16_t _jpeg_Y_density;
    int32_t _jpeg_d_frame_num_of_components;
    struct frame_component_info frame_component_info;
    int32_t _jpeg_error_stat;
    uint8_t _jpeg_d_precision;
    int32_t _jpeg_d_num_of_components;
    struct component_info component_info;
    int32_t flagStreamHeader[3];
};
```

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

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

表3.5.構造体_jpeg_dec_SMBのメンバ説明

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_thinning_mode	-	reserved
_jpeg_d_number_of_lines	Y	元画像のライン数
_jpeg_d_line_length	X	1ライン当たりの画素数
_jpeg_X_density	-	reserved
_jpeg_Y_density	-	reserved

構造体メンバ名	記号	機能概要
_jpeg_d_frame_num_of_components	-	走査内の成分の数
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、以下に記述します
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
_jpeg_error_stat	-	エラーコードを格納する変数
_jpeg_d_precision	-	reserved
_jpeg_d_num_of_components	Ns	走査内の成分の数を格納する配列
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、以下に記述します
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
flagStreamHeader[]	-	JPEGヘッダ確認用フラグ群

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

入力するデータはJPEGデコードライブラリの入力バッファとして設定したRAM領域に格納するか、 入力するデータが格納されているROM領域をJPEGデコードライブラリの入力バッファとして設定する必要があります。 入力バッファとは、JPEGファイルのデータの一部または全部が格納されるアドレス空間上の領域です。

■入力バッファの管理方法

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

```
uint8_t *_jpeg_next_read_byte;     //次のデータの読み出し位置
int32_t _jpeg_d_free_in_buffer;    //有効なデータ数(バイト数)
```

入力バッファから1バイトのデータが読み出されるごとに_jpeg_next_read_byteは1が加算され、_jpeg_d_free_in_bufferは1が減算されます。

■入力バッファの確保
入力バッファ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。 入力バッファの先頭から4バイトはJPEGデコードライブラリの予約領域となっています。 この4バイトは、0xFFで初期化する必要があります。 5バイト目からJPEGファイルのデータを格納してください。 5バイト目以降が有効なデータサイズとなります。

■入力バッファに有効なデータが無くなった場合の処理（バッファリング機能）
データ読み出し時に入力バッファに有効なデータがない場合、ユーザ定義関数が呼ばれます。この関数内で入力バッファに続きのデータを補充することにより、JPEGファイルを分割入力することができます。 ユーザ定義関数は、R_init_jpegにてワーク領域管理毎に任意の関数を登録することができます。

JPEGファイルのデータが全てROMなどメモリ空間上のアドレスに連続配置されている場合、入力バッファを使用せずJPEGファイルのある先頭アドレスを入力バッファとみなして一括入力することができます。 このとき先頭4バイトの0xFF領域は必要ありません。

補足説明

JPEGファイル伸張ライブラリでは、JPEGデコードライブラリのバッファリング機能を使用しておりません。

3.3.3. マクロ定義

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

表3.6.エラーコード定義

定義	値	内容
_JPEGD_OK	0	正常終了
_JPEGD_ERROR	-1	エラー終了

表3.7.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3.8.マクロ変数定義（高速メモリ変数群 _jpeg_dec_FMB）

定義	内容
_jpeg_work(base)	((base)->_jpeg_work)
_jpeg_next_read_byte(base)	((base)->_jpeg_next_read_byte)
_jpeg_d_free_in_buffer(base)	((base)->_jpeg_d_free_in_buffer)
_jpeg_cur_read_buffer(base)	((base)->_jpeg_cur_read_buffer)
_jpeg_cur_bits_offset(base)	((base)->_jpeg_cur_bits_offset)
_jpeg_restart_interval(base)	((base)->_jpeg_restart_interval)

表3.9.マクロ変数定義（低速メモリ変数群 _jpeg_dec_SMB）

定義	内容
_jpeg_d_num_QTBL(base)	((base)->_jpeg_d_num_QTBL)
_jpeg_d_number_of_lines(base)	((base)->_jpeg_d_number_of_lines)
_jpeg_d_line_length(base)	((base)->_jpeg_d_line_length)
_jpeg_d_precision(base)	((base)->_jpeg_d_precision)
_jpeg_X_density(base)	((base)->_jpeg_X_density)
_jpeg_Y_density(base)	((base)->_jpeg_Y_density)
_jpeg_d_num_of_components(base)	((base)->_jpeg_d_num_of_components)
_jpeg_d_frame_num_of _components(base)	((base)->_jpeg_d_frame_num_of_components)
_jpeg_error_stat(base)	((base)->_jpeg_error_stat)
component_info(base)	((base)->component_info)
frame_component_info(base)	((base)->frame_component_info)

表3.10.データ読み込みマクロ関数

定義	内容
CHECK_BUFF(env, fmb)	入力バッファの空きチェック
INPUT_BYTE(var, env, fmb)	1バイト長データの読み込み
INPUT_2BYTES(var, env, fmb)	2バイト長データの読み込み
READ_NBYTE(p, n, env, fmb)	nバイト読み込み
SKIP_BYTES(n, env, fmb)	nバイト読み飛ばし
READ_LENGTH(len, env, fmb)	レングス長の読み込み

入力バッファからデータを読み込む場合は、必ずこれらマクロ関数を使用してください。

表3.11.ストリームチェック用マクロ定義

定義	内容
STREAM_HEADER_FLAG_CLEAR (env)	ストリームチェック用のフラグクリア
STREAM_HEADER_FLAG_SET(env, n)	ストリームチェック用のフラグセット
STREAM_HEADER_DQT_SET(env, n)	ストリームチェック用のDQTフラグセット
DQT_BITS	4:DQT用の確保ビット数
DHT_INDEX2N(n)	DHT(n)のindexをフラグのビット情報へ変換
STREAM_HEADER_DHT_SET(env, n)	ストリームチェック用のDHT(n)フラグセット
STREAM_HEADER_CHECK_ QUNAT_TABLE(env, n)	量子化テーブル(n)の整合性チェック
STREAM_HEADER_CHECK_ DC_TABLE(env, n)	DCハフマンテーブル(n)の整合性チェック
STREAM_HEADER_CHECK_ AC_TABLE(env, n)	ACハフマンテーブル(n)の整合性チェック

定義	内容
STREAM_HEADER_MUST_SET0	必須ヘッダの定義1
STREAM_HEADER_MUST_SET1	必須ヘッダの定義2
STREAM_HEADER_CHECK(f,v)	ヘッダ(v)の有無チェック
STREAM_HEADER_CHECK_SOF0 (flag)	SOF0の有無チェック
STREAM_HEADER_CHECK_SOS (flag)	SOSの有無チェック
STREAM_HEADER_FLAG_CHECK (env)	必須ヘッダの有無チェック

3.4. ライブラリ関数

本節ではJPEGデ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。

関数名 分類

機能概要

書式	関数の呼び出し形式を示します。#include "ヘッダファイル"で示すヘッダファイルは、この関数の実行に必要なヘッダファイルです。必ずインクルードしてください。
引数	関数の引数を示します。"I/O"には引数がそれぞれ入力値、出力値であることを示します。"説明"には"引数名"についての説明を示します。
戻り値	関数の戻り値を示します。"説明"には戻り値の値についての説明を示します。
説明	関数の仕様を示します。
注意事項	関数を使用する際の注意事項を示します。
使用例	関数の使用例を示します。
作成例	関数の作成例を示します。

図3.7.ライブラリ関数詳細の見方

R_jpeg_make_huff_table

ライブラリ関数

— ハフマンテーブルの登録

書式

```
#include "r_jpegd.h"

int16_t R_jpeg_make_huff_table (
                uint8_t index ,
                uint8_t *huffval ,
                uint8_t *bits ,
                int16_t count ,
                struct _jpeg_working *wenv );
```

引数

引数名	I/O	説明
index	I	ハフマンテーブル番号(Tc/Th)
huffval	I	ハフマンコードに関連する値(Vij)を格納した領域へのポインタ
bits	I	ハフマンコードの長さ(Li)を格納した領域へのポインタ
count	I	登録するデータ数
wenv	I/O	JPEGデコード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デコードライブラリ環境に対し、 JPEGファイル中のハフマンテーブル定義(DHT)に記録されたデータを登録します。

indexには上位4ビットでテーブルクラスを、下位4ビットでハフマンテーブル識別子を指定します。 テーブルクラスは、0（DCテーブル）、1（ACテーブル）を指定します。 ハフマンテーブル識別子は、ハフマンテーブルのテーブル番号（0,1）を指定します。 indexに上記以外の値が指定されたときの動作は不定です。

huffvalには、ハフマンコードに関連する値(Vij)を格納した領域へのポインタを指定します。

bitsには、ハフマンコードの長さ(Li)を格納した領域へのポインタを指定します。

countには、登録するデータ数を指定します。

R_jpeg_add_iquant_table

ライブラリ関数

— 量子化テーブルの登録

書式

#include "r_jpegd.h"

```
int16_t R_jpeg_add_iquant_table (
                int16_t qtbl_no ,
                uint16_t *qtbl ,
                struct _jpeg_working *wenv );
```

引数

引数名	I/O	説明
qtbl_no	I	量子化テーブル識別子(Tq)
qtbl	I	量子化テーブル要素(Qk)を格納した領域へのポインタ
wenv	I/O	JPEGデコード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デコードライブラリ環境に対し、 JPEGファイル中の量子化テーブル定義(DQT)に記録されたデータを登録します。

qtbl_noには、量子化テーブル識別子(Tq）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。

qtblには、本関数で登録する量子化テーブル要素(Qk)が格納された領域(uint16_t型8×8の配列)へのポインタを指定します。 量子化テーブルは、ジグザグシーケンスの順のまま、本関数に渡してください。

制限事項 (SH-2A向け)

qtblのアドレスは4の倍数となるよう配置してください。

R_jpeg_decode_one_block

ライブラリ関数

— ハフマン復号化

書式

```
#include "r_jpegd.h"

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

引数

引数名	I/O	説明
last_dc_val	I	前ブロックのDC係数
dc_tbl_no	I	ハフマンテーブルのDCテーブル番号
ac_tbl_no	I	ハフマンテーブルのACテーブル番号
block	O	ハフマン復号化された量子化DCT係数の格納領域へのポインタ
wenv	I/O	JPEGデコード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デコードライブラリ環境に対し、入力バッファからJPEGファイルを読み出し、 blockで指定する領域にハフマン復号データ（量子化DCT係数）を格納します。

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、ハフマンテーブルのDCテーブル番号（0,1）を指定します。 処理の対象がY(輝度)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、ハフマンテーブルのACテーブル番号（0,1）を指定します。 処理の対象がY(輝度)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。

blockには本関数により出力されるハフマン復号データ（量子化DCT係数）を格納する領域（64+2個）を指定します。 blockの指す領域は本関数実行前に0クリアしてください。

注意

本関数の実行中に入力バッファがデータが無くなった場合、ユーザ定義関数R_jpeg_read_inputが呼び出され入力バッファを初期化します。 R_jpeg_read_inputの実行終了後、本関数の実行が

継続されます。 R_jpeg_read_inputの仕様については、R_jpeg_read_input をご参照ください。
本関数の実行の前に、R_jpeg_make_huff_table関数によるハフマン符号テーブルの登録が必要です。

制限事項 (SH-2A向け)

blockのアドレスは4の倍数となるよう配置してください。

使用例

```
#include "r_jpegd.h"

#define DCT_WORK    align.mem.dct_work

static union {
  uint32_t dummy;
  struct {
    int16_t dct_work[_JPEG_DCTSIZE2 + 2];
    uint8_t  Y[_JPEG_DCTSIZE2 * 4];
    uint8_t Cb[_JPEG_DCTSIZE2];
    uint8_t Cr[_JPEG_DCTSIZE2];
  } mem;
} align;
static uint8_t *Y_outptr[_JPEG_DCTSIZE*2];
static uint8_t *Cb_outptr[_JPEG_DCTSIZE];
static uint8_t *Cr_outptr[_JPEG_DCTSIZE];
static int16_t Y_last_dc_val, Cb_last_dc_val, Cr_last_dc_val;

void
sample(void)
{
      ...
  for (i = 0; i < lines; ++i)
  {
    for (j = 0; j < width; ++j)
    {
      //  Y0
      ret = R_jpeg_decode_one_block(Y_last_dc_val, Y_dc_tbl_no, Y_ac_tbl_no, DCT_WORK, wenv);
      if(ret != _JPEGD_OK)
      {
        return EXPAND_JPEGD_ERROR_DECODE;
      }
      Y_last_dc_val = DCT_WORK[0];
      R_jpeg_IDCT(DCT_WORK, Y_outptr, 0, Y_q_tbl_no, wenv);
      //  Y1
      ret = R_jpeg_decode_one_block(Y_last_dc_val, Y_dc_tbl_no, Y_ac_tbl_no, DCT_WORK, wenv);
      if(ret != _JPEGD_OK)
      {
        return EXPAND_JPEGD_ERROR_DECODE;
      }
      Y_last_dc_val = DCT_WORK[0];
      R_jpeg_IDCT(DCT_WORK, Y_outptr, _JPEG_DCTSIZE, Y_q_tbl_no, wenv);
      //  Cb
      ret = R_jpeg_decode_one_block(Cb_last_dc_val, Cb_dc_tbl_no, Cb_ac_tbl_no, DCT_WORK, wenv);
      if(ret != _JPEGD_OK)
```

```
        {
          return EXPAND_JPEGD_ERROR_DECODE;
        }
        Cb_last_dc_val = DCT_WORK[0];
        R_jpeg_IDCT(DCT_WORK, Cb_outptr, 0, Cb_q_tbl_no, wenv);
        //  Cr
        ret = R_jpeg_decode_one_block(Cr_last_dc_val, Cr_dc_tbl_no, Cr_ac_tbl_no, DCT_WORK, wenv);
        if(ret != _JPEGD_OK)
        {
          return EXPAND_JPEGD_ERROR_DECODE;
        }
        Cr_last_dc_val = DCT_WORK[0];
        R_jpeg_IDCT(DCT_WORK, Cr_outptr, 0, Cr_q_tbl_no, wenv);
                    …
      }
    }
  }
```

R_jpeg_IDCT

ライブラリ関数

— 逆量子化と逆DCT

書式

```
#include "r_jpegd.h"

void R_jpeg_IDCT (
                int16_t *block ,
                uint8_t *outptr[] ,
                int16_t start_col ,
                int16_t qtbl_no ,
                struct _jpeg_working *wenv );
```

引数

引数名	I/O	説明
block	I	ハフマン復号化された量子化DCT係数の格納領域へのポインタ
outptr	I	処理結果を格納するラインの先頭アドレスを格納した配列
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値
qtbl_no	I	量子化テーブルの番号
wenv	I/O	JPEGデコード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デコードライブラリ環境に対し、blockの指す領域に格納されたハフマン復号データ（量子化DCT係数）を入力して逆量子化と逆DCTを行い、outptrとstart_colで指定する領域にY,Cb,Crの成分のデータを格納します。

blockには、処理の対象のハフマン復号データ（量子化DCT係数）が格納された領域へのポインタを指定します。 outptrには、成分のデータを格納するラインへのポインタ配列へのポインタを指定します。すなわち、outptr[0]、outptr[1] ... がそれぞれ指すラインに成分のデータが格納されます。配列outptrの必要な要素数は8になります。

start_colには、outptrで指定するラインの先頭から出力データを書き込む位置までのオフセットを指定します。

qtbl_noには、量子化テーブルの番号（0,1）を指定します。qtbl_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

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

制限事項 (SH-2A向け)

blockのアドレスは4の倍数となるよう配置してください。

outptrで指定する処理結果を格納するラインは、先頭アドレスが4の倍数となるよう配置してください。 またstart_colの値は4の倍数で指定してください。

使用例

図3.8.「R_jpeg_IDCT関数の引数outptrとstart_colの関係」 に逆量子化と逆DCTを行う場合の例を示します。

例1のように記述した場合、ブロック1に伸張された成分が格納されます。 例2のように記述した場合、ブロック6に伸張された成分が格納されます。

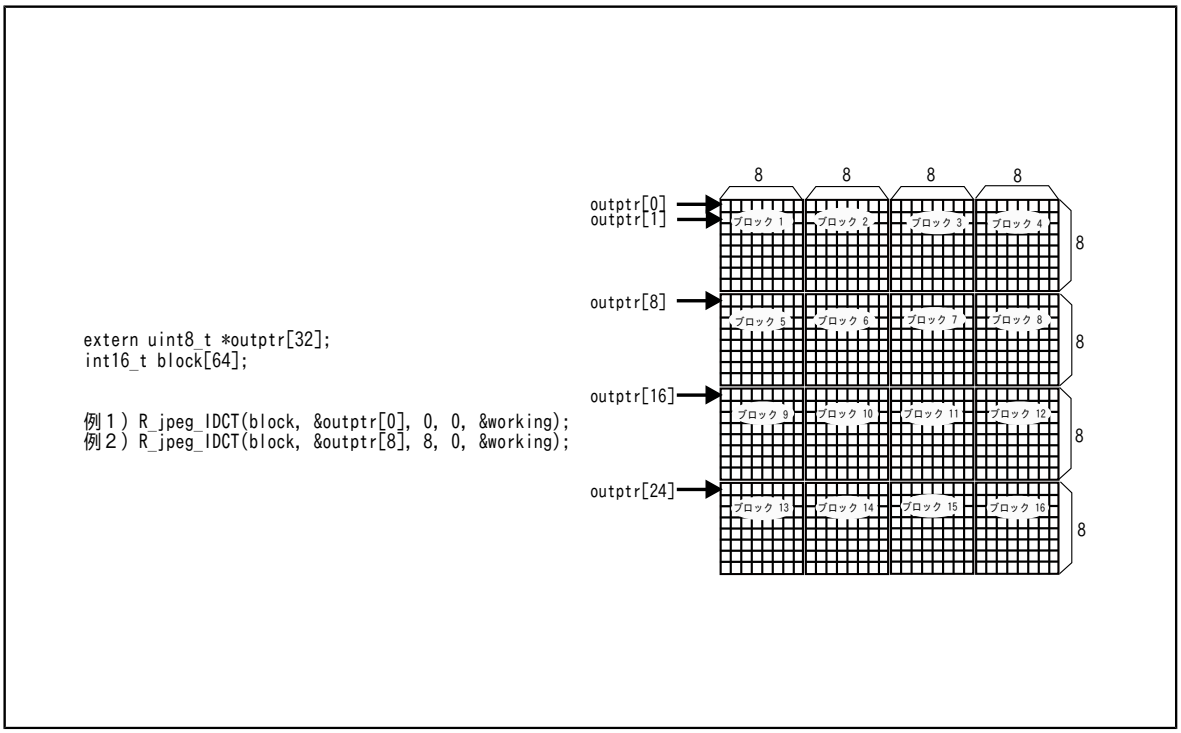

図3.8.R_jpeg_IDCT関数の引数outptrとstart_colの関係

R_jpeg_readRST

ライブラリ関数

— リスタートマーカの検出

書式

```
#include "r_jpegd.h"

int16_t R_jpeg_readRST (
                struct _jpeg_working *wenv );
```

引数

引数名	I/O	説明
*wenv	I/O	JPEGデコード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
-1	エラー
0～7	正常終了

説明

本関数はwenvで指定されたJPEGデコードライブラリ環境に対し、リスタートマーカ(RSTm)を読み出し、 リスタートマーカ番号(m)を戻り値として返します。

戻り値は、0～7まで順に繰り返すモジュロ8の値が返されます。 戻り値と期待値を比較することで、データの誤りを検出することができます。

3.5. ユーザ定義関数

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

R_jpeg_read_input

ユーザ定義関数

— JPEGファイルの入力

書式

```
#include "r_jpegd.h"

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

引数

引数名	I/O	説明
wenv	I/O	JPEGデコードライブラリ環境変数構造体を指すアドレス

戻り値

なし

説明

本関数は入力バッファの有効なデータが無くなった場合に呼出され、アプリケーションにより入力バッファにJPEGファイルのデータを補充します。

入力バッファにデータを補充し、マクロ関数_jpeg_d_free_in_buffer()と_jpeg_next_read_byte()を用いて入力バッファの情報を更新してください。 JPEGファイルを一括入力している場合は、本関数で行う処理はありません。

注意

本関数を使用する場合、入力バッファの先頭4バイトはライブラリが使用します。 この4バイトは0xFFで初期化する必要があります。 入力するデータは5バイト目から格納してください。

作成例

```
void _jpeg_read_input(struct _jpeg_working *wenv)
{
    /* 入力バッファに読み込むデータ（バイト）数を設定 */
    _jpeg_d_free_in_buffer(wenv->decFMB) = JPG_BUFSIZE;

    /* 入力バッファから次に読み出すデータの位置を設定(先頭４バイトはライブラリで使用）*/
    _jpeg_next_read_byte(wenv->decFMB)   = _input_buf + 4;

    /* 入力バッファに新たなデータを読み込む */
    transmit_data(src_address, (int32_t)_input_buf+4, JPG_BUFSIZE);

    /* 転送元アドレスを更新 */
    src_address += JPG_BUFSIZE;
}
```

JPEGデコーダ
ユーザーズマニュアル

発行年月日	2015年 3月17日	Rev.1.03
	2015年 3月17日	Rev.1.03
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デコーダ
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

R20UW0075JJ0103