SH7268/7269 グループ

JPEG コーデックユニットドライバー ユーザーズマニュアル

R01AN2338JJ0100
Rev.1.00
2014.9.30

要旨

本アプリケーションノートは、SH7268/SH7269 の JPEG コーデックユニット(以下 JCU)ドライバの仕様について説明するものです。

動作確認デバイス

SH7268/SH7269

目次

1. 概要

1.1 環境

本ドライバの開発および動作確認環境を以下に示します。

CPU

SH7269

開発環境

HEW (SuperH RISC engine microcomputer software integrated development environment) Version 4.09.01

Renesas SuperH RISC engine Standard Toolchain Version 9.4.1.0

- SH C/C++ Compiler Version 9.04.00
- SH Assembler Version 7.01.02
- SH C/C++ Standard Library Generator Version 3.00.03
- Optimizing Linkage Editor Version 10.00.01

評価ボード

SH7269 CPU board (Part number: R0K572690C000BR)

SH7269 VDC4 board (Part number: R0K572690B000BR)

1.2　機能

本ドライバのサポートする機能を以下に示します。

Table 1 JCU driver 機能

共通項目	対応 JPEG 規格	JPEG ベースライン - 2 成分を持つスキャンに未対応 - 複数成分のノンインタリーブスキャンに未対応
	演算精度	JPEG Part2, ISO-IEC10918-2 準拠
	画像入出力方式	ブロックインターリーブ方式
	画像データレート	最大 133.34MB/s (66.67MHz 動作時)
	分割処理	指定したライン数、データ数転送ごとに入力データの転送を一時的に止めるモードをサポート
	処理単位	アドレス境界 8 バイト単位で設定可能
	処理可能画像サイズ	Minimum Coded Unit(MCU)単位で割り切れるサイズ YCbCr4:2:2: 16 ピクセル×8 ライン YCbCr4:2:0: 16 ピクセル×16 ライン 0 ライン、0 ピクセルの画像は、処理禁止とする。
圧縮処理	対応入力フォーマット	YCbCr4:2:2
	出力フォーマット	JPEG ベースライン(YCbCr4:2:2 形式として保存)
	量子化テーブル	4 テーブル内蔵
	ハフマンテーブル	4 テーブル内蔵 - AC 係数 2 テーブル - DC 係数 2 テーブル
	生成可能マーカ	SOI/SOF0/SOS/DQT/DHT/DRI/RSTm/EOI
伸長処理	対応入力フォーマット	JPEG ベースライン フォーマットは YCbCr4:2:2 および YCbCr4:2:0 に対応。これら以外の画像はデコード禁止とする。
	対応出力フォーマット	YCbCr4:2:2, ARGB8888 または RGB565

1.3　ファイル構成

本ドライバのファイル構成を以下に示します。なお、OS 移植層本体のファイルは除きます。

Table 2　ファイル構成

File Name	概要
jcu_api.c	Source file for JCU driver functions.(main)
jcu_para.c	Source file checking arguments.
jcu_reg.c	Source file controlling registers.
jcu_pl.c	Source file for JCU driver functions.(interrupt handlers, and porting layer)
jcu_misc.c	Source file for JCU driver functions.(other)
r_jcu_api.h	Header file including the prototype declarations for the JCU driver calls and definitions of constants.
r_jcu_local.h	Header file including local definitions.
r_jcu_pl.h	Header file including interrupt hanlers and porting layer)
jcu_namecnv.h	Header file for old naming rule.
r_jcu_user.h	Header file for compilation option.

また、本ドライバを使用する際には下表に示す外部ヘッダファイルが必要となります。

Table 3　外部ファイルの依存関係

File Name	概要
typedefine.h, r_typedefs.h	Header file including the typedef declarations for the basic types.
iodefine.h	Header file including IO definitions.
r_ospl.h	Header file including OS porting layer.

1.4　プログラムサイズとセクション

JCU driver で使用するプログラムサイズとセクションを Table 4 に記載します 。

Table 4　プログラムサイズとセクション

"Renesas SuperH RISC engine Standard Toolchain 9.3.2.0"
"Speed & size optimization enabled"

Type	セクション	サイズ[byte]	概要
ROM	P_JCU	5464	Program area
	C_JCU	1161	Constant area
	D_JCU	0	Initialized data area
RAM	B_JCU	104	Uninitialized data area

Note:プログラムサイズは、デコードやエンコードの結果を保存する RAM 領域は含みません。。
パラメータのチェックが定義されている場合、括弧内のサイズになります。
サイズは、最適化オプションによって変わります。

1.5　基本動作

本ドライバは JCU の持つ機能である圧縮/伸長処理をサポートしています。
これらの処理は、排他的に使用します。

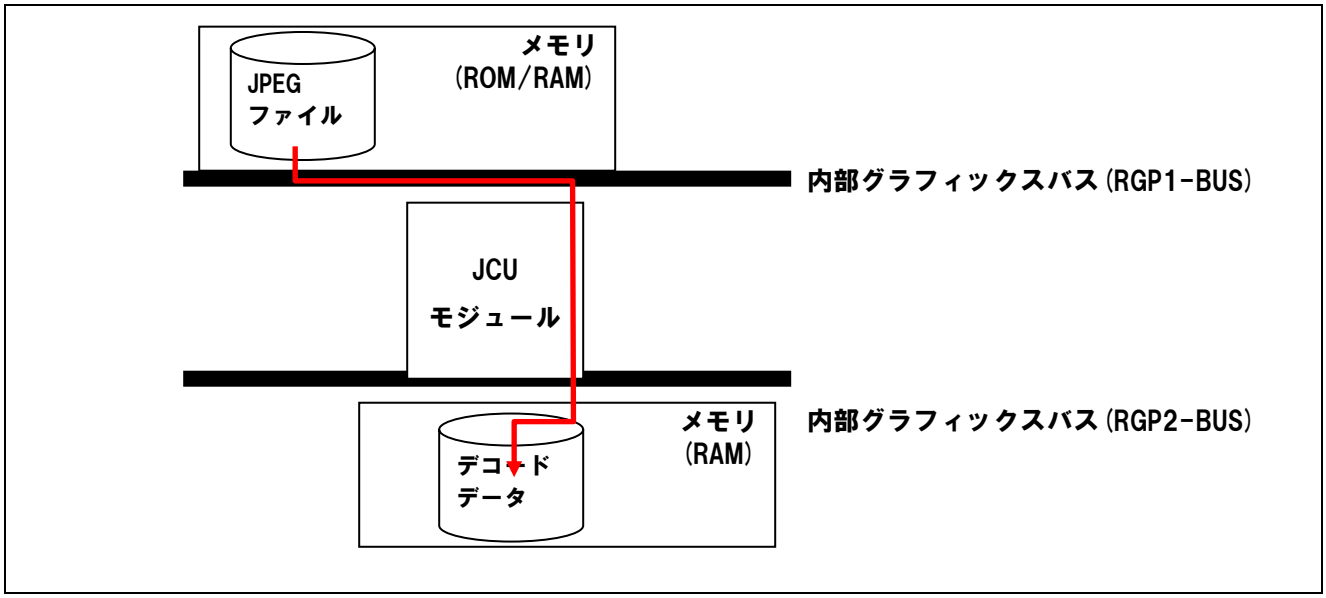


Figure 1　基本動作処理

1.6　状態遷移

本ドライバは動作状態を管理し、状態に応じて処理の可否を判定します。
本状態は API を使用することで変化します。API の使用と状態遷移の関係を以下に示します。

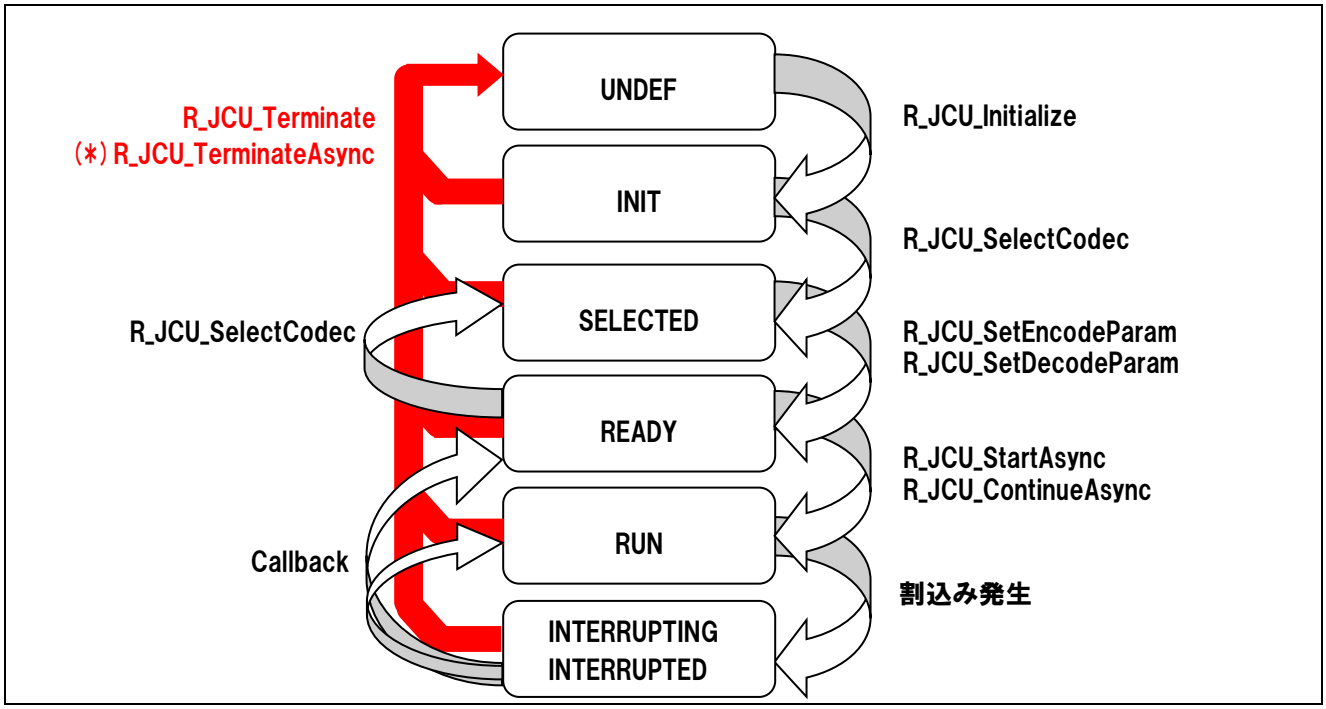


Figure 2　JCU driver 状態遷移

Table 5 は、各ドライバが呼び出し可能な状態遷移を示します。

Table 5 状態表

API	呼び出し可能な状態						実行後の状態
	UNDEF	INIT	SELECTED	READY	RUN	INTERRUPTING INTERRUPTED	
共通 API							
R_JCU_Initialize	OK	NG	NG	NG	NG	NG	INIT
R_JCU_Terminate	OK	OK	OK	OK	OK	OK	UNDEF
R_JCU_TerminateAsync	OK	OK	OK	OK	OK*1	OK	UNDEF
R_JCU_SelectCodec	NG	OK	OK	OK	NG	NG	SELECTED
R_JCU_SetCountMode	NG	NG	OK	OK	NG	NG	変更なし
R_JCU_Start	NG	NG	NG	OK	NG	NG	変更なし
R_JCU_StartAsync	NG	NG	NG	OK	NG	NG	RUN
R_JCU_Continue	NG	NG	NG	OK	NG	NG	変更なし
R_JCU_ContinueAsync	NG	NG	NG	OK	NG	NG	RUN
R_JCU_GetAsyncStatus	OK	OK	OK	OK	OK	OK	変更なし
R_JCU_OnInterrupting*2	NG	NG	NG	NG	NG	OK	INTERRUPTED
R_JCU_OnInterrupted	NG	NG	NG	NG	NG	OK	READY または RUN
伸長用 API							
R_JCU_SetPauseForImageInfo	NG	NG	OK	OK	NG	NG	変更なし
R_JCU_SetErrorFilter	NG	OK	OK	OK	NG	NG	変更なし
R_JCU_SetDecodeParam	NG	NG	OK	OK	NG	NG	Ready
R_JCU_GetImageInfo	NG	NG	NG	OK*3	NG	NG	変更なし
R_JCU_GetErrorInfo	NG	NG	OK	OK	NG	NG	変更なし
圧縮用 API							
R_JCU_SetEncodeParam	NG	NG	OK	OK	NG	NG	Ready
R_JCU_SetQuantizationTable	NG	NG	OK	OK	NG	NG	変更なし
R_JCU_SetHuffmanTable	NG	NG	OK	OK	NG	NG	変更なし
R_JCU_GetEncodedSize	NG	NG	NG	OK	NG	NG	変更なし

*1: RUN 中に実行した場合、直後の状態は「変更なし」となる。その後の割込み発生（コールバック関数実行）のタイミングで、「UNDEF」状態に遷移する

*2: OSPL のデフォルトコールバック内から呼び出す。呼び出す直前に「INTERRUPTING」状態に遷移する

*3: JCU_INT_GET_IMAGE_INFO の割り込み発生後に確認可能

1.7　割り込みハンドラ

本ドライバは割り込みハンドラを実装しています。割り込みハンドラを以下に示します。

Table 6　割り込みハンドラ

割り込み要因	割り込みベクタ		割り込みハンドラ
	番号	アドレス	
JEDI 圧縮伸長処理割り込み要求	181	0x000002D4 ~ 0x000002D7	void INT_JCU_JEDI (void);
JDTI データ転送処理割り込み要求	182	0x000002D8 ~ 0x000002DB	void INT_JCU_JDTI (void);

ユーザが JCU の割り込み処理を利用する場合、Table 6 の関数を割り込みハンドラとして登録する必要があります。割り込みハンドラの登録機能を実装している OS 等を使用する場合は、その機能を利用することで Table 6 の割り込みハンドラの関数を登録してください。そうでない場合は、ユーザが Table 6 に示された関数をベクタテーブルへ登録してください。

1.8　コンパイラスイッチ

本ドライバでは"jcu_user.h"ファイルにおいてコンパイルスイッチが定義されています。

1.8.1　パラメータチェック

"JCU_PARAMETER_CHECK"の定義を有効にすると、API のコール時に引数のチェックを行います。パラメータチェックの結果、エラーがある場合はエラーコードを返します。エラーコードについては「2.1.2(1) jcu_errorcode_t」と「2.1.2(4) jcu_detail_error_t」の項を参照してください。

1.8.2　割り込みハンドラの定義

本ドライバには割り込みハンドラ用の関数が用意されています(参照 1.7 割り込みハンドラ)。RTOS を使用しない場合は、"IS_USE_RTOS"を未定義にして下さい。"#pragma　interrupt"宣言により、ハンドラ用の関数が、割り込み関数としてコンパイラに処理されます。この場合は、割込みベクタテーブルに、ハンドラ関数を静的に登録する必要があります。

OS 等の機能を利用して割り込みハンドラ用の関数を登録する場合には、"IS_USE_RTOS"の定義を有効にしてください。

1.9　制限事項

1.9.1　予約語

本ドライバでは他のプログラムと区別する為、関数や変数名などのシンボル名称にプレフィックス"JCU_"を付加しています。大文字、小文字を問わず”JCU”から始まるシンボルは使用しないでください。

1.9.2　中断処理

本ハードウェアには、処理中に中断する機能は含まれていません。
処理を中断したい場合は、現在の圧縮・伸長処理が終了した上で再度実行する必要があります。

また処理中に、モジュールスタンバイやソフトウェアリセットを使用して、処理を中断させることもしないでください。

1.9.3　出力分割処理

出力エンコードおよび分割デコードの出力分割機能は、SH7269 のハードウェアの仕様から削除されました。

2. API

2.1 一般定義

2.1.1 基本型

本ドライバでは以下に示した基本型宣言を使用します。基本型宣言は、”r_typedefs.h”にて定義されています。 (参照 1.3 ファイル構成)。

Table 7　基本型

基本型	定義
int8_t	typedef signed char
uint8_t	typedef unsigned char
int16_t	typedef signed short
uint16_t	typedef unsigned short
int32_t	typedef signed int
uint32_t	typedef unsigned int
char_t	typedef char
bool_t	typedef int
int_fast32_t	typedef int
uint_fast32_t	typedef unsigned int
bit_flags_fast32_t	typedef unsigned int

2.1.2 列挙型・変数型および定数の定義

本ドライバでは以下に示した列挙型・変数型および定数を使用します。

Table 8　列挙型・変数型

section	列挙型・変数型
(1)	jcu_errorcode_t
(2)	jcu_codec_t
(3)	jcu_continue_type_t
(4)	jcu_detail_error_t
(5)	jcu_int_detail_error_t
(6)	jcu_int_detail_errors_t
(7)	jcu_interrupt_line_t
(8)	jcu_interrupt_lines_t
(9)	jcu_swap_t
(10)	jcu_sub_sampling_t
(11)	jcu_decode_format_t
(12)	jcu_jpeg_format_t
(13)	jcu_huff_t
(14)	jcu_table_no_t
(15)	jcu_color_element_t
(16)	jcu_status_information_t
(17)	jcu_codec_status_t
(18)	jcu_sub_state_t
(19)	jcu_sub_status_t

(1) jcu_errorcode_t

jcu_errorcode_t 型は、API からのリターン値（エラーコード）を示す型です。jcu_errorcode_t 型の変数には、以下の enum 値を使用します。

```
typedef errnum_t  jcu_errorcode_t;
enum {
    JCU_ERROR_OK          = 0x0000,
    JCU_ERROR_PARAM       = 0x4501,
    JCU_ERROR_STATUS      = 0x4502,
    JCU_ERROR_CODEC_TYPE = 0x4503,
    JCU_ERROR_LIMITATION = 0x4504
};
```

Enum	Value	概要
JCU_ERROR_OK	0x0000	正常終了
JCU_ERROR_PARAM	0x4501	パラメータエラー
JCU_ERROR_STATUS	0x4502	状態エラー
JCU_ERROR_CODEC_TYPE	0x4503	圧縮/伸長のタイプエラー
JCU_ERROR_LIMITATION	0x4504	制限事項に引っかかった

(2) jcu_codec_t

jcu_codec_t は圧縮・伸長を示す列挙型です。

```
typedef enum {
  JCU_ENCODE    = 0,
  JCU_DECODE    = 1
} jcu_codec_t;
```

Enum	Value	概要
JCU_ENCODE	0	圧縮処理
JCU_DECODE	1	伸長処理

(3) jcu_continue_type_t

jcu_continue_type_t は JCU が中断された場合に、再開するモードを示す列挙型です。

```
typedef enum {
  JCU_INPUT_BUFFER,
  JCU_OUTPUT_BUFFER,
  JCU_GET_IMAGE_INFO
} jcu_continue_type_t;
```

Enum	Value	Summary
JCU_INPUT_BUFFER	0	入力バッファの一時停止を再開する
JCU_OUTPUT_BUFFER	1	使用できません
JCU_GET_IMAGE_INFO	2	画像情報取得後の一時停止を再開する

(4) jcu_detail_error_t

jcu_detail_error_t 型は、デコード時に発生したエラー検出の種類を表す変数型です。jcu_detail_error_t 型の変数には、以下の enum 値を使用します。

エラー検出の詳細については、SH7268/69 ユーザーズマニュアル ハードウェア編 表 40.3 および表 40.4 を参照ください。

```
typedef errnum_t  jcu_detail_error_t;
enum {
  JCU_JCDERR_OK                     = 0x0000,
  JCU_JCDERR_SOI_NOT_FOUND          = 0x4521,
  JCU_JCDERR_INVALID_SOF            = 0x4522,
  JCU_JCDERR_UNPROVIDED_SOF         = 0x4523,
  JCU_JCDERR_SOF_ACCURACY           = 0x4524,
  JCU_JCDERR_DQT_ACCURACY           = 0x4525,
  JCU_JCDERR_COMPONENT_1            = 0x4526,
  JCU_JCDERR_COMPONENT_2            = 0x4527,
  JCU_JCDERR_NO_SOF0_DQT_DHT        = 0x4528,
  JCU_JCDERR_SOS_NOT_FOUND          = 0x4529,
  JCU_JCDERR_EOI_NOT_FOUND          = 0x452A,
  JCU_JCDERR_RESTART_INTERVAL_NUM   = 0x452B,
  JCU_JCDERR_IMAGE_SIZE             = 0x452C,
  JCU_JCDERR_LAST_MCU_NUM           = 0x452D,
  JCU_JCDERR_BLOCK_NUM              = 0x452E
};
```

Enum	Value	Summary
JCU_JCDERR_OK	0x0000	正常
JCU_JCDERR_SOI_NOT_FOUND	0x4521	SOI 未検出。EOI 検出まで SOI 未検出
JCU_JCDERR_INVALID_SOF	0x4522	SOF1～SOFF の検出
JCU_JCDERR_UNPROVIDED_SOF	0x4523	対象外の間引きを検出
JCU_JCDERR_SOF_ACCURACY	0x4524	SOF 精度異常。「8」以外を検出
JCU_JCDERR_DQT_ACCURACY	0x4525	DQT 精度異常。「0」以外を検出
JCU_JCDERR_COMPONENT_1	0x4526	コンポーネント異常 1。SOF0 ヘッダのコンポーネント数が「1」「3」「4」以外を検出
JCU_JCDERR_COMPONENT_2	0x4527	コンポーネント異常 2。SOF0 ヘッダのコンポーネント数と SOS のコンポーネント数が異なる場合
JCU_JCDERR_NO_SOF0_DQT_DHT	0x4528	SOS 検出時に SOF0、DQT、DHT 未検出
JCU_JCDERR_SOS_NOT_FOUND	0x4529	SOS 未検出。EOI 検出までに SOS 未検出
JCU_JCDERR_EOI_NOT_FOUND	0x452A	EOI 未検出（デフォルト）
JCU_JCDERR_RESTART_INTERVAL_NUM	0x452B	リスタートインターバルデータ数エラーを検出
JCU_JCDERR_IMAGE_SIZE	0x452C	画像サイズエラーを検出*
JCU_JCDERR_LAST_MCU_NUM	0x452D	最終 MCU データ数エラーを検出
JCU_JCDERR_BLOCK_NUM	0x452E	ブロックデータ数エラーを検出

* JPEG データの中の圧縮データ部分（ハフマン符号化セグメント、マーカーはない）の後ろに EOI マーカー以外があると JCU_JCDERR_IMAGE_SIZE エラーが発生することがあります。R_JCU_SetErrorFilter 関数（JINTE0 レジスターの一部）に JCU_INT_ERROR_SEGMENT_TOTAL_DATA と JCU_INT_ERROR_MCU_BLOCK_DATA のビットを指定しなければ、JCU_JCDERR_IMAGE_SIZE エラーを検出しなくなります。

(5)　jcu_int_detail_error_t

jcu_int_detail_error_t 型は、詳細なエラー検出の種類を表す列挙型です。jcu_int_detail_error_t 型は、以下の値の型です。jcu_int_detail_errors_t 型の変数に格納できます。

```
typedef enum {
  JCU_INT_ERROR_RESTART_INTERVAL_DATA   = 0x80u,
  JCU_INT_ERROR_SEGMENT_TOTAL_DATA      = 0x40u,
  JCU_INT_ERROR_MCU_BLOCK_DATA          = 0x20u,
  JCU_INT_ERROR_ALL                     = ( JCU_INT_ERROR_RESTART_INTERVAL_DATA |
              JCU_INT_ERROR_SEGMENT_TOTAL_DATA | JCU_INT_ERROR_MCU_BLOCK_DATA )
}  jcu_int_detail_error_t
```

Enum	Value	Summary
JCU_INT_ERROR_RESTART_INTERVAL_DATA	0x80u	ハフマン符号化セグメント内のリスタートインターバル間のデータ数に異常があるエラー
JCU_INT_ERROR_SEGMENT_TOTAL_DATA	0x40u	ハフマン符号化セグメント内の総データ数に異常があるエラー
JCU_INT_ERROR_MCU_BLOCK_DATA	0x20u	ハフマン符号化セグメント内の最終 MCU データ数に異常があるエラー
JCU_INT_ERROR_ALL	0xE0	全てのエラー

(6)　jcu_int_detail_errors_t

jcu_int_detail_errors_t 型は、jcu_int_detail_error_t の各値の論理和を表す変数型です。jcu_int_detail_errors_t 型の変数には、jcu_int_detail_error_t 型の値を使用する事ができます。

```
typedef bit_flags_fast32_t  jcu_int_detail_errors_t;
```

(7)　jcu_interrupt_line_t

jcu_interrupt_line_t 型は、JCU の割込みラインの種類を表す列挙型です。jcu_interrupt_line_t 型は、以下の値の型です。jcu_interrupt_lines_t 型の変数に格納できます。

```
typedef enum {
  JCU_INTERRUPT_LINE_JEDI   = 0x00000001u,
  JCU_INTERRUPT_LINE_JDTI   = 0x00000002u,
  JCU_INTERRUPT_LINE_ALL    =
                    ( JCU_INTERRUPT_LINE_JEDI | JCU_INTERRUPT_LINE_JDTI )
} jcu_interrupt_line_t;
```

Enum	Value	Summary
JCU_INTERRUPT_LINE_JEDI	0x00000001u	伸長時の画像情報取得可能時、圧縮データエラー発生時、正常終了時に発生する IEDI 割込み
JCU_INTERRUPT_LINE_JDTI	0x00000002u	分割処理時の一時停止発生時、全データ書き出し終了時に発生する JDTI 割込み
JCU_INTERRUPT_LINE_ALL	0x00000003u	JEDI、JDTI 両方の割込み

(8)　jcu_interrupt_lines_t

jcu_interrupt_lines_t 型は、jcu_interrupt_line_t の各値の論理和を表す変数型です。jcu_interrupt_lines_t 型の変数には、jcu_interrupt_line_t 型の値を使用する事ができます。

```
typedef bit_flags_fast32_t  jcu_interrupt_lines_t;
```

(9) jcu_swap_t

jcu_swap_t 型は、入出力のデータのスワップを示す列挙型です。jcu_swap_t 型の変数には、以下の以下の enum 値を使用します。

```
typedef enum {
  JCU_SWAP_NONE                          0x00
  JCU_SWAP_BYTE                          0x01
  JCU_SWAP_WORD                          0x02
  JCU_SWAP_WORD_AND_BYTE                 0x03
  JCU_SWAP_LONG_WORD                     0x04
  JCU_SWAP_LONG_WORD_AND_BYTE            0x05
  JCU_SWAP_LONG_WORD_AND_WORD            0x06
  JCU_SWAP_LONG_WORD_AND_WORD_AND_BYTE   0x07
} jcu_swap_t;
```

Enum	Value	Summary
JCU_SWAP_NONE	0x00	スワップなし
JCU_SWAP_BYTE	0x01	バイトスワップ
JCU_SWAP_WORD	0x02	ワードスワップ
JCU_SWAP_WORD_AND_BYTE	0x03	ワード バイトスワップ
JCU_SWAP_LONG_WORD	0x04	ロングワードスワップ
JCU_SWAP_LONG_WORD_AND_BYTE	0x05	ロングワード バイトスワップ
JCU_SWAP_LONG_WORD_AND_WORD	0x06	ロングワード ワードスワップ
JCU_SWAP_LONG_WORD_AND_WORD_AND_BYTE	0x07	ロングワード ワード バイトスワップ

(10) jcu_sub_sampling_t

jcu_sub_sampling_t は伸長時の出力画像の間引き設定を示す列挙型です。

```
typedef enum {
    JCU_SUB_SAMPLING_1_1 = 0x00,
    JCU_SUB_SAMPLING_1_2 = 0x01,
    JCU_SUB_SAMPLING_1_4 = 0x02,
    JCU_SUB_SAMPLING_1_8 = 0x03
} jcu_sub_sampling_t;
```

Enum	Value	Summary
JCU_SUB_SAMPLING_1_1	0x00	間引きなし
JCU_SUB_SAMPLING_1_2	0x01	1/2 に間引き
JCU_SUB_SAMPLING_1_4	0x02	1/4 に間引き
JCU_SUB_SAMPLING_1_8	0x03	1/8 に間引き

(11) jcu_decode_format_t

jcu_decode_format_t は伸長時の出力画像のピクセルフォーマットを示す列挙型です。

```
typedef enum {
    JCU_OUTPUT_YCbCr422  = 0x00,
    JCU_OUTPUT_ARGB8888  = 0x01,
    JCU_OUTPUT_RGB565    = 0x02
} jcu_decode_format_t;
```

Enum	Value	Summary
JCU_OUTPUT_YCbCr422	0x00	YCbCr4:2:2
JCU_OUTPUT_ARGB8888	0x01	ARGB8888
JCU_OUTPUT_RGB565	0x02	RGB565

(12) jcu_jpeg_format_t

jcu_jpeg_format_t は伸長時の画像情報を取得するときのピクセルフォーマットを示す列挙型です。

```
typedef enum {
    JCU_JPEG_YCbCr422    = 0x01,
    JCU_JPEG_YCbCr420    = 0x02
} jcu_jpeg_format_t;
```

Enum	Value	Summary
JCU_JPEG_YCbCr422	0x01	YCbCr4:2:2
JCU_JPEG_YCbCr420	0x02	YCbCr4:2:0

(13) jcu_huff_t

jcu_huff_t はハフマンテーブルの AC/DC 成分を示す列挙型です。

```
typedef enum {
    JCU_HUFFMAN_AC,
    JCU_HUFFMAN_DC
} jcu_huff_t;
```

Enum	Value	Summary
JCU_HUFFMAN_AC	0x00	AC 成分を示します
JCU_HUFFMAN_DC	0x01	DC 成分を示します

(14) jcu_table_no_t

jcu_table_no_t は量子化テーブルまたはハフマンテーブルのテーブル番号を示す列挙型です。

```
typedef enum {
    JCU_TABLE_NO_0    = 0,
    JCU_TABLE_NO_1    = 1,
    JCU_TABLE_NO_2    = 2,
    JCU_TABLE_NO_3    = 3
} jcu_table_no_t;
```

Enum	Value	Summary
JCU_TABLE_NO_0	0x00	量子化テーブル番号 0(JCQTBL0)または DC/AC ハフマンテーブル番号 0(JCHTBD0/JCHTBA0)を選択する
JCU_TABLE_NO_1	0x01	量子化テーブル番号 1(JCQTBL1)または DC/AC ハフマンテーブル番号 1(JCHTBD1/JCHTBA1)を選択する
JCU_TABLE_NO_2	0x02	量子化テーブル番号 2(JCQTBL2)を選択する ハフマンテーブルでは使用できません
JCU_TABLE_NO_3	0x03	量子化テーブル番号 3(JCQTBL3)を選択する ハフマンテーブルでは使用できません

(15) jcu_color_element_t

jcu_color_element_t は使用する量子化テーブルまたはハフマンテーブルのインデックスを示す列挙型です。

```
typedef enum {
    JCU_ELEMENT_Y,
    JCU_ELEMENT_Cb,
    JCU_ELEMENT_Cr
} jcu_color_element_t;
```

Enum	Value	Summary
JCU_ELEMENT_Y	0x00	輝度用テーブルを選択する
JCU_ELEMENT_Cb	0x01	色差(Cb)用テーブルを選択する
JCU_ELEMENT_Cr	0x02	色差(Cr)用テーブルを選択する

(16) jcu_status_information_t

jcu_status_information_t はドライバのステータスを示す列挙型です。

```
typedef enum {
    JCU_STATUS_UNDEF        = 0x00,
    JCU_STATUS_INIT         = 0x01,
    JCU_STATUS_SELECTED     = 0x02,
    JCU_STATUS_READY        = 0x08,
    JCU_STATUS_RUN          = 0x10,
    JCU_STATUS_INTERRUPTING = 0x40,
    JCU_STATUS_INTERRUPTED  = 0x80
} jcu_status_information_t;
```

Enum	Value	Summary
JCU_STATUS_UNDEF	0x00	初期化前状態
JCU_STATUS_INIT	0x01	初期化済状態
JCU_STATUS_SELECTED	0x02	圧縮・伸長を選択済み
JCU_STATUS_READY	0x08	実行準備が整った または 実行完了
JCU_STATUS_RUN	0x10	実行中
JCU_STATUS_INTERRUPTING	0x40	割込み発生した後の状態
JCU_STATUS_INTERRUPTED	0x80	割込み処理実行後の状態

(17) jcu_codec_status_t

jcu_codec_status_t は現在の処理が圧縮処理か伸長処理かを示す列挙型です。

```
typedef enum {
    JCU_CODEC_NOT_SELECTED  = -1,
    JCU_STATUS_ENCODE       = 0,
    JCU_STATUS_DECODE       = 1
} jcu_codec_status_t;
```

Enum	Value	Summary
JCU_CODEC_NOT_SELECTED	-1	圧縮･伸長が選択されていない
JCU_STATUS_ENCODE	0	圧縮処理が選択されている
JCU_STATUS_DECODE	1	伸長処理が選択されている

(18) jcu_sub_state_t

jcu_sub_state_t 型は、JCU ドライバのサブステータスの種類を表す列挙型です。jcu_sub_state_t 型は、以下の値の型です。jcu_sub_status_t 型の変数に格納できます。

```
typedef enum {
   JCU_SUB_INFORMATION_READY         = 0x00000008u,
   JCU_SUB_DECODE_OUTPUT_PAUSE       = 0x00000100u,
   JCU_SUB_DECODE_INPUT_PAUSE        = 0x00000200u,
   JCU_SUB_ENCODE_OUTPUT_PAUSE       = 0x00001000u,
   JCU_SUB_ENCODE_INPUT_PAUSE        = 0x00002000u,
   JCU_SUB_PAUSE_ALL                 = JCU_SUB_INFORMATION_READY |
         JCU_SUB_DECODE_INPUT_PAUSE | JCU_SUB_ENCODE_OUTPUT_PAUSE  |
         JCU_SUB_ENCODE_OUTPUT_PAUSE | JCU_SUB_ENCODE_INPUT_PAUSE
} jcu_sub_state_t;
```

Enum	Value	Summary
JCU_SUB_INFORMATION_READY	0x00000008u	デコード処理中に、情報取得可能なポーズ状態になった事を示す
JCU_SUB_DECODE_OUTPUT_PAUSE	0x00000100u	出力分割デコード処理中に、出力分割によるポーズ状態になった事を示す
JCU_SUB_DECODE_INPUT_PAUSE	0x00000200u	入力分割デコード処理中に、入力分割によるポーズ状態になった事を示す
JCU_SUB_ENCODE_OUTPUT_PAUSE	0x00001000u	出力分割エンコード処理中に、出力分割によるポーズ状態になった事を示す
JCU_SUB_ENCODE_INPUT_PAUSE	0x00002000u	入力分割エンコード処理中に、入力分割によるポーズ状態になった事を示す
JCU_SUB_PAUSE_ALL	0x00003308u	全てのポーズ原因の論理和を示す

(19) jcu_sub_status_t

jcu_sub_status_t は jcu_sub_state_t の各値の論理和を表す変数型です。jcu_sub_status_t 型の変数には、jcu_sub_state_t 型の値を使用する事ができます。

```
typedef uint_fast32_t  jcu_sub_status_t;
```

2.1.3　構造体型の定義

本ドライバでは以下に示した構造体型を使用します。以下に、構造体型の一覧表を示します。

Table 9　構造体型

section	構造体型
(1)	jcu_count_mode_param_t
(2)	jcu_buffer_t
(3)	jcu_buffer_param_t
(4)	jcu_decode_param_t
(5)	jcu_image_info_t
(6)	jcu_encode_param_t
(7)	jcu_async_status_t
(8)	jcu_internal_information_t

(1)　jcu_count_mode_param_t

jcu_count_mode_param_t はカウントモード(分割処理)を行うための構造体です。

メンバ inputBuffer および outputBuffer はそれぞれ JCU に対する入力側、出力側を意味します。

なお、本ドライバでは、出力側の分割機能を使用する事は出来ません。出力側の isEnable メンバは、必ず false を設定して下さい。

```
typedef struct {
    struct {
        bool_t      isEnable;
        bool_t      isInitAddress;
        uint32_t*    restartAddress;
        uint32_t    dataCount;
    } inputBuffer;
    struct {
        bool_t      isEnable;
        bool_t      isInitAddress;
        uint32_t*    restartAddress;
        uint32_t    dataCount;
    } outputBuffer;
} jcu_count_mode_param_t;
```

Member	Summary
isEnable	false: 入力/出力バッファの分割処理を行いません true: 入力/出力バッファの分割処理を行います **出力側は、必ず false を設定して下さい**
isInitAddress	false: 一時停止時に、入力バッファのアドレスを初期化しません true: 一時停止時に、入力バッファのアドレスを初期化します
restartAddress	isInitAddress が true のときの初期化アドレスを指定します
dataCount	入力バッファの分割サイズを指定します。指定できる値は、８の倍数です。 デコードする場合は、指定したバイト数のデータを JCU に入力すると、一時停止します。 エンコードする場合は、指定したライン数のデータを JCU に入力すると、一時停止します。

(2) jcu_buffer_t

jcu_buffer_t は入出力バッファを設定するための構造体です。

```
typedef struct {
    jcu_swap_t     swapSetting;
    uint32_t*      address;
} jcu_buffer_t;
```

Member	Summary
swapSetting	データのスワップ方法を選択します。
address	バッファのアドレスを選択します。

(3) jcu_buffer_param_t

jcu_buffer_param_t はエンコードおよびデコード時の入出力バッファを設定するための構造体です。

```
typedef struct {
    jcu_buffer_t   source;
    jcu_buffer_t   destination;
    int16_t        lineOffset;
} jcu_buffer_param_t;
```

Member	Summary
source	入力バッファの設定を行います。
destination	出力バッファの設定を行います。
lineOffset	フレームバッファのオフセットを選択します。 圧縮時: 入力画像のラインオフセットを意味します 伸長時: 出力画像のラインオフセットを意味します

(4) jcu_decode_param_t

jcu_decode_param_t は伸長時のオプションを選択します。

```
typedef struct {
    jcu_sub_sampling_t   verticalSubSampling;
    jcu_sub_sampling_t   horizontalSubSampling;
    jcu_decode_format_t  decodeFormat;
    uint8_t              alpha;
} jcu_decode_param_t;
```

Member	Summary
verticalSubSampling	垂直方向の間引き設定を行います
horizontalSubSampling	水平方向の間引き設定を行います
decodeFormat	デコード後のデータのピクセルフォーマットを選択します
alpha	ピクセルフォーマットに ARGB888 を選択したときに設定するα値を選択します。 ARGB8888 以外のときは 0 を設定してください。

(5) jcu_image_info_t

jcu_image_info_t は伸長時の対象ファイルの画像情報を取得するための構造体です。

```
typedef struct {
    uint32_t            width;
    uint32_t            height;
    jcu_jpeg_format_t   encodedFormat;
} jcu_image_info_t;
```

Member	Summary
width	画像の幅を格納します
height	画像の高さを格納します
encodedFormat	JPEG ファイルのピクセルフォーマットを格納します

(6) jcu_encode_param_t

jcu_encode_param_t は圧縮時のオプションを選択します。

```
typedef struct {
    jcu_jpeg_format_t    encodeFormat;
    int_t                QuantizationTable[JCU_COLOR_ELEMENT_NUM];
    int_t                HuffmanTable[JCU_COLOR_ELEMENT_NUM];
    uint32_t             DRI_value;
    uint32_t             width;
    uint32_t             height;
} jcu_encode_param_t;
```

Member	Summary
encodeFormat	圧縮時のピクセルフォーマットを選択します JCU_JPEG_YCbCr422 を指定してください。
QuantizationTable	圧縮時に使用する量子化テーブルを選択します
HuffmanTable	圧縮時に使用するハフマンテーブルを選択します
DRI_value	圧縮データの DRI(Define Restart Interval)値を選択します
width	入力画像の幅を選択します
height	入力画像の高さを選択します

(7) jcu_async_status_t

jcu_async_status_t は、ドライバの内部状態と割込みステータスを管理するための構造体です。

```
typedef struct st_jcu_async_status_t  jcu_async_status_t;
struct st_jcu_async_status_t {
    jcu_status_information_t   Status;
    jcu_sub_status_t           SubStatusFlags;
    bool_t                     IsPaused;
    bool_t                     IsEnabledInterrupt;
    r_ospl_flag32_t            InterruptEnables;
    r_ospl_flag32_t            InterruptFlags;
    r_ospl_flag32_t            CancelFlags;
};
```

Member	Summary
Status	ドライバの内部状態を定義します。
jcu_sub_states_t	ドライバのサブ状態を定義します。状態は、ビットフラグで表します。2.1.2(18)章参照。
IsPaused	false:一時停止が無効（伸長時の画像情報を取得後の割込みが無効、かつ、分割処理による一時停止が無効） true:一時停止が有効
IsEnabledInterrupt	false:JCU の I-ロック状態が設定中 true: JCU の I-ロック状態が解除中
InterruptEnables	JCU の有効な割込みを登録します。
InterruptFlags	JCU の割込み処理内で制御されるフラグです。
CancelFlags	JCU の割込み処理内で参照されるフラグです。

(8) jcu_internal_information_t

jcu_internal_information_t は、ドライバの内部状態を設定するための構造体です。

```
typedef struct {
    jcu_codec_status_t            Codec;
    bool_t                        IsCountMode;
    jcu_int_detail_errors_t       ErrorFilter;
    jcu_async_status_t            AsyncStatus;
    r_ospl_caller_t               InterruptCallbackCaller;
    jcu_i_lock_t*                 I_Lock;
    const r_ospl_i_lock_vtable_t* I_LockVTable;
    bool_t                        Is_I_LockMaster;
    r_ospl_async_t*               AsyncForFinalize;
} jcu_internal_information_t;
```

Member	Summary
Codec	選択されているコーデック種別です。
IsCountMode	false:JCU ドライバはカウントモード（分割処理）ではありません。 true: JCU ドライバはカウントモード（分割処理）です。
ErrorFilter	有効であるデコードエラー情報です。
AsyncStatus	割込みや非同期処理の状況です。
InterruptCallbackCaller	OSPL に登録した割込みコールバック関数です。
I_Lock	I-ロック(割込み禁止を伴う排他制御)状態です。
I_LockVTable	I-ロック制御を行う関数のポインタをあつめたものです。
IS_I_LockMaster	false:I_LockVTable は設定されていません。 true: I_LockVTable は設定されています。
AsyncForFinalize	RUN 状態中に、R_JCU_TerminateAsync が実行された状況で設定される、OSPL 用のパラメータです。

(9)　jcu_i_lock_t

jcu_i_lock_t は、ドライバの内部状態を設定するための構造体です。

```
typedef struct st_jcu_i_lock_t   jcu_i_lock_t;

struct st_jcu_i_lock_t {
  bool_t       IsLock;
  bool_t       IsRequestedFinalize;
};
```

Member	Summary
IsLock	false:JCU ドライバは I-ロック設定されていません。 true: JCU ドライバは I-ロック設定されています。
IsRequestedFinalize	false:JCU ドライバは終了要求されていません。 true: JCU ドライバは終了要求されています。

2.1.4　OS 移植層（OSPL）の定義型

本ドライバでは、OS 移植層（OSPL）の関数を呼び出す時に、引数や戻り値として、OSPL の汎用の型を使用します。本ドライバで使用する OSPL の汎用型は、以下の通りです。これらの詳細については、OSPL のユーザーズマニュアルを参照願います。

Table 10　OSPL 使用型一覧

型名	概要
errnum_t	エラー情報
r_ospl_async_t	通知設定を管理する構造体
r_ospl_flag32_t	32 ビットフラグ型
r_ospl_interrupt_t	割込みの発信元を管理する構造体
r_ospl_caller_t	割込みコールバックが管理する構造体
r_ospl_i_lock_vtable_t	I-ロックに関する構造体
r_ospl_async_type_t	非同期処理の種類

2.2　API 関数

2.2.1　R_JCU_Initialize

API	jcu_errorcode_t R_JCU_Initialize(void* const NullConfig);	
ヘッダ	#include　"r_jcu_api.h"	
パラメータ	[in] void* const NullConfig	使用しません。ヌルを指定してください。
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_STATUS	エラーコード 正常終了 API コール時のステータスが正しくない
概要	本関数では以下の処理を行います。 ユーザ定義関数(R_JCU_OnInitialize)の実行 ドライバの管理情報の初期化 ドライバ内部の状態の初期化	
使用可能な状態	本 API は以下の状態で使用可能です。 UNDEF 状態	
説明	ドライバの初期化処理を実施します。 また初期化処理に先立ち、ユーザ定義関数 R_JCU_OnInitialize を呼び出します。ユーザ定義関数では、以下の処理を行ってください。 JCU モジュールへのクロック供給 JCU に関係する割り込みの優先度の設定 その他処理に必要な環境固有の設定	
補足		

2.2.2　R_JCU_Terminate

API	jcu_errorcode_t R_JCU_Terminate(void);
ヘッダ	#include　"r_jcu_api.h"
パラメータ	無し
戻り値	jcu_errorcode_t　エラーコード JCU_ERROR_OK　正常終了 JCU_ERROR_PARAM　ユーザ定義関数の戻り値がエラー
概要	本関数では以下の処理を行います。本関数は、以下の処理が終了するまで、リターンしない同期関数です。 ユーザ定義関数(R_JCU_OnFinalize)の実行 ドライバの内部の状態の変更(JCU_STATUS_UNDEF へ遷移)
使用可能な状態	本 API は、どの状態でも使用可能です。もし、JCU Driver 状態が JCU_STATUS_RUN の時に使用した場合は、デコードまたはエンコードが終了して JCU が動作停止するまで待ちます。本関数から戻ったときは、JCU_STATUS_UNDEF 状態になります。
説明	本関数では、ユーザ定義関数 R_JCU_OnFinalize を呼び出します。ユーザ定義関数では以下の処理を行ってください。 JCU モジュールへのクロック供給停止 割り込み優先度の設定クリア その他環境固有の設定
補足	

2.2.3　R_JCU_TerminateAsync

API	jcu_errorcode_t R_JCU_TerminateAsync(r_ospl_async_t* const　async);	
ヘッダ	#include　"r_jcu_api.h"	
パラメータ	[in] r_ospl_async_t* const　async	OS 移植層の汎用引数。2.1.4 章を参照
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM	エラーコード 正常終了 引数がヌル、または、ユーザ定義関数の戻り値がエラー
概要	R_JCU_Terminate 処理を参照して下さい。 本関数は、処理が終了する前に、すぐにリターンする非同期関数です。	
使用可能な状態	本 API は、どの状態でも使用可能です。もし、JCU Driver 状態が JCU_STATUS_RUN の時に使用した場合は、すぐに終了処理を行わず、そのままの状態でリターンします。(デコードまたはエンコードが終了して JCU が動作停止した時に、処理終了を行います)	
説明	R_JCU_Terminate 処理を参照して下さい。 引数 async については、OS 移植層（OSPL）のマニュアルの R_DRIVER_TransferAsync 関数を参照してください。	
補足		

2.2.4　R_JCU_SelectCodec

API	jcu_errorcode_t R_JCU_SelectCodec(const jcu_codec_t codec);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const jcu_codec_t codec	コーデック種別
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 引数が正しくない API コール時のステータスが正しくない
概要	本関数では以下の処理を行います。 JCU の処理状態を選択する(圧縮または伸長から選択する)	
使用可能な状態	本 API は以下の状態で使用可能です。 INIT 状態 SELECTED 状態 READY 状態	
説明	JCU の動作モードを選択します。	
補足	本 API 関数を実行すると、デコードとエンコードに使うパラメータと、カウントモードの設定が初期化されますので、全て再設定して下さい。	

2.2.5　R_JCU_SetCountMode

API	jcu_errorcode_t R_JCU_SetCountMode(const jcu_count_mode_param_t* const buffer);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const jcu_count_mode_param_t* const buffer	カウントモード（分割処理）設定
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 引数が正しくない API コール時のステータスが正しくない
概要	本関数では以下の処理を行います。 カウントモード（分割処理）の設定を行う	
使用可能な状態	本 API は以下の状態で使用可能です。 SELECTED 状態 READY 状態	

説明

JCU の分割処理を設定します。
なお、SH7269 では、出力分割の処理を行う事はできません。出力分割処理を、”JCU_PARAMETER_CHECK”の定義を有効にして実行した場合は、エラーとなります。無効にして実行した場合は、メモリ破壊などの深刻な問題が発生する可能性があります。

R_JCU_Start を呼び出す前に本 API で設定を行うことで、分割処理が実施されます。

jcu_count_mode_param_t の inputBuffer および outputBuffer の対象は次のようになります。

処理設定	inputBuffer の対象データ	outputBuffer の対象データ
エンコード	入力画像	―
デコード	入力 JPEG データ	―

また inputBuffer および outputBuffer の dataCount は次のようになります。

処理設定	inputBuffer の処理単位	outputBuffer の処理単位
エンコード	8Line 単位	―
デコード	8byte 単位	―

jcu_count_mode_param_t::inputBuffer.isEnable = false　なら分割処理ではなくなります。

補足

2.2.6 R_JCU_SetPauseForImageInfo

API	jcu_errorcode_t R_JCU_SetPauseForImageInfo(const bool_t is_pause);	
ヘッダ	#include "r_jcu_api.h"	
パラメータ	[in] const bool_t is_pause	true:一時停止する false:一時停止しない
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_STATUS	エラーコード 正常終了 API コール時のステータスが正しくない
概要	JCU のデコード中に、画像情報取得可能になったら一時停止するかどうかを選択します。	
使用可能な状態	本 API は以下の状態で、かつ、JCU_DECODE を選択している場合のみ使用できます。 SELECTED 状態 READY 状態	
説明	R_JCU_GetImageInfo 関数で、JPEG ファイルの画像情報が取得できる状態になったら、一時停止するかどうかを設定します。	
補足		

2.2.7 R_JCU_SetErrorFilter

API	jcu_errorcode_t R_JCU_SetErrorFilter(jcu_int_detail_errors_t filter);	
ヘッダ	#include "r_jcu_api.h"	
パラメータ	[in] jcu_int_detail_errors_t filter	有効とするエラー検出の種類 (jcu_int_detail_errors_t)のビット・フラグ値
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 引数が正しくない API コール時のステータスが正しくない
概要	有効とするデコードエラー検出の種類(jcu_int_detail_errors_t)を設定します。	
使用可能な状態	本 API は以下の状態で使用できます。 INIT 状態 SELECTED 状態 READY 状態	
説明	有効とするデコードエラー検出の種類を設定します。有効としたデコードエラーが発生した場合は、内部で割込みが発生して、R_JCU_Start 関数の戻り値をエラーコードにします。	
補足		

2.2.8　R_JCU_Start

API	jcu_errorcode_t R_JCU_Start(void);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	なし	
戻り値	jcu_errorcode_t	エラーコード
	JCU_ERROR_OK	正常終了
	JCU_ERROR_STATUS	API コール時のステータスが正しくない
概要	JCU モジュールのデコードまたはエンコード動作を開始します。本関数は、デコードまたはエンコードの処理が、完了または中断するまで、リターンしない同期関数です。	
使用可能な状態	本 API は以下の状態で使用可能です。 READY 状態	
説明	JCU のデコードまたはエンコードを実行します。 本 API を使用する際は、あらかじめ R_JCU_SetDecodeParam または R_JCU_SetEncodeParam の API を使用してパラメータの設定を行ってください。 JCU をスタートした後は、キャンセルできません。	
補足	R_JCU_SetEncodeParam または R_JCU_SetDecodeParam で設定したパラメータ設定が正しくない場合でも本 API はエラーを返しません。	

2.2.9　R_JCU_StartAsync

API	jcu_errorcode_t R_JCU_StartAsync(r_ospl_async_t* const　async);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] r_ospl_async_t* const　async	OS 移植層の汎用引数。2.1.4 章を参照
戻り値	jcu_errorcode_t	エラーコード
	JCU_ERROR_OK	正常終了
	JCU_ERROR_STATUS	API コール時のステータスが正しくない
概要	JCU モジュールのデコードまたはエンコード動作を開始します。本関数は、デコードまたはエンコードの処理が終了する前に、すぐにリターンする非同期関数です。(コールバック関数で、デコードまたはエンコード終了した事を判断します)	
使用可能な状態	本 API は以下の状態で使用可能です。 READY 状態	
説明	R_JCU_Start 処理を参照して下さい。 引数 async については、OS 移植層（OSPL）のマニュアルの R_DRIVER_TransferAsync 関数を参照してください。	
補足	R_JCU_SetEncodeParam または R_JCU_SetDecodeParam で設定したパラメータ設定が正しくない場合でも本 API はエラーを返しません。	

2.2.10　R_JCU_Continue

API	jcu_errorcode_t R_JCU_Continue(const jcu_continue_type_t type);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const jcu_continue_type_t type	JCU が再開するモード
戻り値	jcu_errorcode_t _t JCU_ERROR_OK JCU_ERROR_STATUS	エラーコード 正常終了 API コール時のステータスが正しくない
概要	一時中断した JCU モジュールのデコードまたはエンコード動作を再開します。本関数は、デコードまたはエンコードの処理が、完了または中断するまで、リターンしない同期関数です。	
使用可能な状態	本 API は以下の状態で使用可能です。 READY 状態	
説明	一時中断していた JCU モジュールのデコードまたはエンコード動作を再開させます。再開するモード（中断原因）は、引数で指定します。	
補足	中断していない動作を引数で指定した場合、エラーは返しません。	

2.2.11　R_JCU_ContinueAsync

API	jcu_errorcode_t R_JCU_Continue(const jcu_continue_type_t type , r_ospl_async_t* const async);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const jcu_continue_type_t type [in] r_ospl_async_t* const　async	JCU が再開するモード OS 移植層の汎用引数。2.1.4 章を参照
戻り値	jcu_errorcode_t _t JCU_ERROR_OK JCU_ERROR_STATUS	エラーコード 正常終了 API コール時のステータスが正しくない
概要	一時中断した JCU モジュールのデコードまたはエンコード動作を再開します。本関数は、デコードまたはエンコードの処理が終了する前に、すぐにリターンする非同期関数です。(コールバック関数で、デコードまたはエンコード終了した事を判断します)	
使用可能な状態	本 API は以下の状態で使用可能です。 READY 状態	
説明	R_JCU_Continue 処理を参照して下さい。 引数 async については、OS 移植層（OSPL）のマニュアルの R_DRIVER_TransferAsync 関数を参照してください。	
補足	中断していない動作を引数で指定した場合、エラーは返しません。	

2.2.12　R_JCU_GetAsyncStatus

API	void R_JCU_GetAsyncStatus(const jcu_async_status_t** const out_Status);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[out] const jcu_async_status_t** const out_Status	ドライバの内部状態を格納するメモリへのポインタ
戻り値	なし	なし
概要	JCU ドライバの内部状態と割込み、非同期処理の状況を示すメモリ（構造体）へのポインタを取得します。	
使用可能な状態	本 API は、どの状態でも使用可能です。	
説明	JCU ドライバの内部状態と割込みの状態を参照します。	
補足	引数に指定するポインタ変数には、const 修飾子が必要です。	

2.2.13　R_JCU_OnInterrupting

API	errnum_t R_JCU_OnInterrupting(const r_ospl_interrupt_t* const　InterruptSource);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const r_ospl_interrupt_t* const InterruptSource	割込み発信元の情報を格納するメモリへのポインタ。2.1.4 章を参照
戻り値	errnum_t 0 E_OTHERS E_STATE	エラー情報 エラーなし その他のエラー 現在の状態では実行できないというエラー
概要	割込みを受信します。	
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 OSPL のデフォルトコールバック関数「R_JCU_OnInterruptDefault」内から本 API がコールバックされます（コールバックされる直前で、OSPL により INTERRUPTING 状態に遷移します）。詳細は 3.1.7 章を参照して下さい。	
説明	本関数により、JCU ドライバ自身に、割込み通知を伝達し、割込みをクリアします。割込み通知は、イベントフラグ待ちでウェイトしている非同期処理の、処理再開のトリガなどに使用されます。	
補足	R_JCU_OnInterruptDefault は、続けて R_JCU_OnInterrupted 関数を呼び出すか、イベントフラグのセットを行います。	

2.2.14　R_JCU_OnInterrupted

API	errnum_t R_JCU_OnInterrupted(void);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	なし	なし
戻り値	errnum_t 0 E_OTHERS E_STATE jcu_detail_error_t の各値	エラー情報 エラーなし その他のエラー 現在の状態では実行できないというエラー デコードエラーが発生した時のエラー値
概要	割込み応答処理を行います。	
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 デフォルトコールバック関数「R_JCU_OnInterruptDefault」内から本 API がコールバックされます（コールバックされる直前で、「R_JCU_OnInterrupting」を実行し、それによって、INTERRUPTED 状態に遷移します）。詳細は 2.2.13 章、3.1.7 章を参照して下さい。	
説明	本関数により、R_JCU_OnInterrupting 関数で行われる「割込み通知」をクリアし、割込み応答処理を行います。割込み応答処理は、イベントフラグ待ちでウェイトしている非同期処理の再開のトリガなどに使用されます。 デコードエラーが発生していた場合は、本関数の戻り値にて、jcu_detail_error_t 型の値を返します。	
補足	OS 移植層およびコールバック処理を、デフォルトのまま使用する場合、本 API 関数の戻り値は、ASync 構造体の ReturnValue メンバに格納されます。	

2.2.15　R_JCU_SetDecodeParam

API	jcu_errorcode_t R_JCU_SetDecodeParam(const jcu_decode_param_t* const decode, const jcu_buffer_param_t* const buffer);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const jcu_decode_param_t* decode [in] const jcu_buffer_param_t* buffer	デコード用パラメータの設定 入出力バッファ設定
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 パラメータが正しくない API コール時のステータスが正しくない
概要	伸長時のパラメータ設定を行います。	
使用可能な状態	本 API は以下の状態でかつ、JCU_DECODE を選択している場合のみ使用できます。 SELECTED 状態 READY 状態	
説明	伸長時に必要なパラメータの設定を行います。	
補足	フォーマットに ARGB8888 以外を設定したときは、decode.alpha には 0 を設定して下さい。	

2.2.16　R_JCU_GetImageInfo

API	jcu_errorcode_t R_JCU_GetImageInfo(jcu_image_info_t* const buffer);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[out] jcu_image_info_t* const buffer	画像情報の保存先領域
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 パラメータが正しくない API コール時のステータスが正しくない
概要	JPEG ファイルの画像情報を取得します。	
使用可能な状態	本 API は以下の状態で、かつ、JCU_DECODE を選択している場合のみ使用できます。 READY 状態	
説明	画像情報取得の割込み発生後に、以下の JPEG ファイルの情報を取得することができます。 画像のサイズ(幅、高さ) JPEG データのエンコードフォーマット なお、取得した JPEG ファイル画像情報が、以下のいずれかの場合は、エラーと判断して、デコードは行わないで下さい。 ・エンコードフォーマットが、jcu_jpeg_format_t の範囲外 ・画像サイズ（幅、高さ）の少なくとも一方が 0	
補足	本 API は画像情報取得の割り込みが入った後のみ有効なデータが読み出せます。 それ以外のときは不定値が返ります。	

2.2.17　R_JCU_GetErrorInfo

API	jcu_errorcode_t R_JCU_GetErrorInfo(jcu_detail_error_t* const errorCode);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[out] jcu_detail_error_t* const errorCode	詳細なエラーコードの保存先領域
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 パラメータが正しくない API コール時のステータスが正しくない
概要	デコード時エラーの詳細を取得	
使用可能な状態	本 API は以下の状態で、かつ、JCU_DECODE を選択している場合のみ使用できます。 READY 状態	
説明	デコード時にエラーが発生した後、その原因を確認することができます。エラーの詳細については、2.1.2(4)を参照ください。 デコードエラーが発生していない場合は、不定値が返ります。	
補足	本 API は、OSPL 非対応の Ver0.09 以前の JCU ドライバの「JCU_GetErrorInfo」に相当する関数です。 OSPL に対応した Ver0.10 以降の JCU ドライバでは、エラー情報は、R_JCU_Start などの関数の返り値、または、Async 構造体の ReturnValue メンバに格納されますので、本 API 関数を使用する必要はありません。詳しくは、2.2.14 章を参照して下さい。	

2.2.18　R_JCU_SetEncodeParam

API	jcu_errorcode_t R_JCU_SetEncodeParam(const jcu_encode_param_t* const encode, const jcu_buffer_param_t* const buffer);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const jcu_encode_param_t* const encode	エンコード用パラメータの設定
	[in] const jcu_buffer_param_t* const buffer	入出力バッファ設定
戻り値	jcu_errorcode_t	エラーコード
	JCU_ERROR_OK	正常終了
	JCU_ERROR_PARAM	パラメータが正しくない
	JCU_ERROR_STATUS	API コール時のステータスが正しくない
概要	圧縮時のパラメータ設定を行います。	
使用可能な状態	本 API は以下の状態で、かつ、JCU_ENCODE を選択している場合のみ使用できます。 SELECTED 状態 READY 状態	
説明	本 API では圧縮時に必要なパラメータの設定を行います。	
補足		

2.2.19　R_JCU_SetQuantizationTable

API	jcu_errorcode_t R_JCU_SetQuantizationTable(const jcu_decode_format_t tableNo, const uint8_t* const table);	
ヘッダ	#include " r_jcu_api.h"	
パラメータ	[in] const jcu_decode_format_t tableNo [in] const uint8_t* const table	データをセットするテーブル番号 設定する量子化テーブル
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 パラメータが正しくない API コール時のステータスが正しくない
概要	量子化テーブルの設定を行います。 量子化テーブルの設定値は、「SH7268 グループ、SH7269 グループ ユーザーズマニュアル ハードウェア編」の、41.3.1(4)章を参照するか、サンプルに付属の量子化テーブル生成ツールをご使用下さい。	
使用可能な状態	本 API は以下の状態で、かつ、JCU_ENCODE を選択している場合のみ使用できます。 SELECTED 状態 READY 状態	
説明	選択したテーブル番号のアドレスにテーブルで与えられたデータを設定します。	
補足	量子化テーブルは、複数の画像を圧縮する時でも一度の設定で圧縮可能です(その都度設定する必要はありません)。	

2.2.20　R_JCU_SetHuffmanTable

API	jcu_errorcode_t R_JCU_SetHuffmanTable(	const jcu_decode_format_t tableNo, const jcu_huff_t type, const uint8_t* const table);
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[in] const jcu_decode_format_t tableNo [in] const jcu_huff_t type, [in] const uint8_t* const table	データをセットするハフマンテーブル番号 ハフマンテーブルに設定する成分 設定するハフマンテーブル
戻り値	jcu_errorcode_t JCU_ERROR_OK JCU_ERROR_PARAM JCU_ERROR_STATUS	エラーコード 正常終了 パラメータが正しくない API コール時のステータスが正しくない
概要	ハフマンテーブルの設定を行います。 ハフマンテーブルの設定値は、「SH7268 グループ、SH7269 グループ ユーザーズマニュアル ハードウェア編」の、41.3.1(4)章を参照して下さい。	
使用可能な状態	本 API は以下の状態で、かつ、JCU_ENCODE を選択している場合のみ使用できます。 SELECTED 状態 READY 状態	
説明	テーブル番号と AC/DC 成分によって選択されるアドレスにハフマンテーブルのデータを設定します。	
補足	ハフマンテーブルは、複数の画像を圧縮する時でも一度の設定で圧縮可能です(その都度設定する必要はありません)。	

2.2.21　R_JCU_GetEncodedSize

API	jcu_errorcode_t R_JCU_GetEncodedSize (size_t* const out_Size);	
ヘッダ	#include　" r_jcu_api.h"	
パラメータ	[out] size_t* const out_Size	圧縮データのサイズ（バイト）
戻り値	jcu_errorcode_t JCU_ERROR_OK	エラーコード 正常終了
概要	圧縮データのサイズ（バイト）を取得します。	
使用可能な状態	本 API は、どの状態でも使用可能ですが、圧縮完了直後にのみ、正しい値を返しします。	
説明	本 API は、圧縮処理を実行して作成した JPEG データのサイズを取得します。圧縮完了直後にのみ、正しい値を返し、それ以外は、不定値を返します。	
補足		

3. その他の関数・定義

3.1 ユーザ定義関数

本ドライバの jcu_pl.c は移植層であり、ここにある各関数は、必要に応じてユーザが実装および改造を行う事ができる、ユーザ定義関数となります。これらのユーザ定義関数について、以下に示します。

3.1.1 R_JCU_OnInitialize

関数名	errnum_t R_JCU_OnInitialize(void);	
ヘッダ	#include "r_jcu_pl.h"	
パラメータ	なし	
戻り値	errnum_t	エラー情報
	0	エラーなし
	E_OTHERS	その他のエラー
概要	本関数では、ユーザが定義した処理を行います。デフォルトでは、以下の処理を実行しています。 JCU の割込み（JEDI、JDTI）の優先度設定 JCU モジュールへのクロック供給	
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 ドライバの初期化処理関数「R_JCU_Initialize」内から本関数がコールバックされます。詳細は 2.2.1 章を参照して下さい。	
説明	本関数は、ユーザ定義関数となります。デフォルトでは、JCU の割込み優先度設定と、クロックの供給を行います。その他、必要な処理があれば、適宜追加して下さい。	
補足		

3.1.2 R_JCU_OnFinalize

関数名	errnum_t R_JCU_OnFinalize(void);	
ヘッダ	#include "r_jcu_pl.h"	
パラメータ	errnum_t e	エラー情報。そのまま戻り値に使用します。
戻り値	errnum_t	エラー情報。引数をそのまま戻り値にします。
概要	本関数では、ユーザが定義した処理を行います。デフォルトでは、以下の処理を実行しています。 JCU モジュールへのクロック供給停止	
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 ドライバの終了関数「R_JCU_Finalize」および「R_JCU_FinalizeAsync」内から本関数がコールバックされます。詳細は 2.2.2 章を参照して下さい。	
説明	本関数は、ユーザ定義関数となります。デフォルトでは、クロックの供給停止を行います。その他、必要な処理があれば、適宜追加して下さい。	
補足		

3.1.3　R_JCU_SetDefaultAsync

関数名	void R_JCU_SetDefaultAsync(r_ospl_async_t* const Async, r_ospl_async_type_t AsyncType);	
ヘッダ	#include　"r_jcu_pl.h"	
パラメータ	r_ospl_async_t* const Async	OS 移植層の汎用引数。2.1.4 章を参照。通知設定として使用する。NULL 指定禁止。
	r_ospl_async_type_t AsyncType	OS 移植層の汎用引数。2.1.4 章を参照。
戻り値	なし	
概要	r_ospl_async_t 型の構造体のデフォルト値を設定します。	
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 非同期の各関数「R_JCU_TerminateAsync」、「R_JCU_StartAsync」、および「R_JCU_ContinueAsync」内から本関数がコールバックされます。	
説明	本関数は、ユーザ定義関数となりますが、OSPL の設定を行うための関数であり、通常は変更する必要はありません。 デフォルトでは、r_ospl_async_t 型の構造体の Flags メンバ変数の値のうち、0 になっているビットに対応する、それぞれのメンバ変数をデフォルト値に設定します。	
補足	r_ospl_async_t 型の構造体の ReturnValue メンバは、それぞれの呼び出し元の非同期処理関数内で初期化しますので、本関数内では設定する必要はありません。	

3.1.4　R_JCU_SetInterruptCallbackCaller

関数名	errnum_t R_JCU_SetInterruptCallbackCaller(const r_ospl_caller_t* const　Caller);	
ヘッダ	#include　"r_jcu_pl.h"	
パラメータ	const r_ospl_caller_t* const　Caller	OS 移植層の汎用引数。2.1.4 章を参照。 R_OSPL_CallInterruptCallback 関数に渡す値。
戻り値	errnum_t	エラー情報。
	0	エラーなし
概要	割込みコールバック関数を呼び出すオブジェクトを、ドライバの移植層に登録します。	
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 非同期の関数「R_JCU_StartAsync」および「R_JCU_ContinueAsync」内から本関数がコールバックされます。	
説明	本関数は、ユーザ定義関数となりますが、OSPL が管理するコールバック関数（JCU の非同期関数を実行した後、割込み発生した時に実行するコールバック関数）を登録するための関数であり、通常は変更する必要はありません。 デフォルトでは、コールバック関数として、R_JCU_OnInterruptDefault 関数を登録します。	
補足		

3.1.5　R_JCU_OnEnableInterrupt

関数名	void_t R_JCU_OnEnableInterrupt(jcu_interrupt_lines_t const　Enables);
ヘッダ	#include　"r_jcu_pl.h"
パラメータ	jcu_interrupt_lines_t const　Enables　JCU の割込みを示すビットフラグ値。
戻り値	なし
概要	JCU ドライバの割込み許可を設定します。
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 OSPL が I_LOCK 解除を行う時に、本関数がコールバックされます。
説明	本関数は、ユーザ定義関数となりますが、OSPL が呼び出す割込み許可要求関数であり、通常は変更する必要はありません。 デフォルトでは、JCU の２つの割込み（JEDI、JDTI）を、引数に応じて、OSPL の割込み許可関数 R_INTC_Enable で有効化します。
補足	

3.1.6　R_JCU_OnDisableInterrupt

関数名	void_t R_JCU_OnDisableInterrupt(jcu_interrupt_lines_t const　Disables1);
ヘッダ	#include　"r_jcu_pl.h"
パラメータ	jcu_interrupt_lines_t const　Enables　JCU の割込みを示すビットフラグ値。
戻り値	なし
概要	JCU ドライバの割込み禁止を設定します。
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 OSPL が I_LOCK を行う時に、本関数がコールバックされます。
説明	本関数は、ユーザ定義関数となりますが、OSPL が呼び出す割込み禁止要求関数であり、通常は変更する必要はありません。 デフォルトでは、JCU の２つの割込み（JEDI、JDTI）を、引数に応じて、OSPL の割込み禁止関数 R_INTC_Disable で無効化します。
補足	

3.1.7　R_JCU_OnInterruptDefault

関数名	errnum_t R_JCU_OnInterruptDefault(const r_ospl_interrupt_t* const　InterruptSource, const r_ospl_caller_t* const　Caller);
ヘッダ	#include　"r_jcu_pl.h"
パラメータ	const r_ospl_interrupt_t* const　InterruptSource　割込み発信元の情報を格納するメモリへのポインタ。2.1.4 章を参照 const r_ospl_caller_t* const　Caller　コールバック関数が管理するメモリ。2.1.4 章を参照。
戻り値	なし
概要	デフォルトの割込みコールバック関数です。
使用可能な状態	本 API は、通常は、ユーザが直接使用する事はありません。 R_JCU_SetDefaultAsync 関数で登録する割り込みコールバック関数は、デフォルトでは本関数が登録されます。この場合は、JCU の割込みが発生した時に、OSPL の割込みコールバック呼び出し関数（R_OSPL_CallInterruptCallback）から、本関数がコールバックされます。
説明	本関数は、ユーザ定義関数となりますが、デフォルトのコールバック関数であり、通常は変更する必要はありません。 デフォルトでは、R_JCU_OnInterrupting 関数と R_JCU_OnInterrupted 関数を呼び出した後、Async 構造体に登録されたイベントをセットします。
補足	

3.2　OS 移植層（OSPL）の使用関数

本ドライバでは、OS 移植層（OSPL）を使用します。本ドライバで使用する OSPL の関数は、以下の通りです。これらの詳細については、OSPL のユーザーズマニュアルの、各関数名の章を参照願います。

Table 11　OSPL 使用関数一覧

関数名	概要
R_OSPL_CALLER_Initialize	（OSPL の UM に記載なし）
R_OSPL_THREAD_GetCurrentId	現在実行中のスレッド ID を取得
R_OSPL_DisableAllInterrupt	全ての割込み禁止
R_OSPL_EnableAllInterrupt	全ての割込み禁止を解除
R_OSPL_FLAG32_InitConst	（32 ビットの）フラグを全て 0 にクリア
R_OSPL_FLAG32_Set	フラグの指定したビットを 1 に設定
R_OSPL_FLAG32_Clear	フラグの指定したビットを 0 に設定
R_OSPL_FLAG32_Get	フラグの値を取得
R_OSPL_FLAG32_GetAndClear	フラグの値を取得してから 0 にクリア
R_OSPL_EVENT_Wait	（16 ビットの）スレッド付属イベントフラグの、指定されたビットがセットされるまでウェイトし、受信したフラグをクリア
R_OSPL_EVENT_Set	スレッド付属イベントの指定したビットを 1 に設定
R_OSPL_EVENT_Clear	スレッド付属イベントの指定したビットを 0 に設定

3.3　旧版(Ver0.09 以前)からの移植

新版(Ver0.10 以降)のドライバは、旧版から、基本型名、列挙型名、構造体型名、関数名を変更しております。旧版から移植する場合は、以下の表に従い、各定義を変更して下さい。

なお、"typedefine.h"と”jcu_namecnv.h”のヘッダファイルをインクルードする事で、旧版の定義を使用する事が可能です。

Table 12　基本型名一覧

基本型(new ver.)	基本型(old ver.)	定義
int8_t	_SBYTE [*1]	typedef signed char
uint8_t	_UBYTE	typedef unsigned char
int16_t	_SWORD	typedef signed short
uint16_t	_UWORD	typedef unsigned short
int32_t [*1]	_SINT [*1]	typedef signed int
uint32_t [*1]	_UINT [*1]	typedef unsigned int
int32_t [*1]	_SDWORD	typedef signed long
uint32_t [*1]	_UDWORD	typedef unsigned long
char_t	_SBYTE [*1]	typedef char
bool_t	JCU_Boolean	typedef int
int_fast32_t	_SINT [*1]	typedef int
uint_fast32_t	_UINT [*1]	typedef unsigned int

[*1] 対応する型が複数存在します。たとえば、旧版の_SBYTE 型は、新版の int8_t 型と、chat_t 型に対応します。

Table 13　列挙型名一覧

列挙型・変数型(new ver.)	列挙型・変数型(old ver.)
jcu_errorcode_t	JCU_ErrorCode
jcu_codec_t	JCU_codec
jcu_continue_type_t	JCU_ContinueType
jcu_detail_error_t	JCU_DetailError
jcu_int_detail_error_t	JCU_IntDetailError
jcu_int_detail_errors_t	JCU_IntDetailErrors
jcu_interrupt_line_t	JCU_InterruptLine
jcu_interrupt_lines_t	JCU_InterruptLines
jcu_swap_t	JCU_Swap
jcu_sub_sampling_t	JCU_SubSampling
jcu_decode_format_t	JCU_DecodeFormat
jcu_jpeg_format_t	JCU_JpegFormat
jcu_huff_t	JCU_HuffType
jcu_table_no_t	JCU_TableNo
jcu_color_element_t	JCU_ColorElement
jcu_status_information_t	jcu_statusInformation
jcu_codec_status_t	jcu_codecStatus

Table 14　構造体型名一覧

構造体型(new ver.)	構造体型(old ver.)
jcu_count_mode_param_t	JCU_CountModeParam
jcu_buffer_t	JCU_Buffer
jcu_buffer_param_t	JCU_BufferParam
jcu_decode_param_t	JCU_DecodeParam
jcu_image_info_t	JCU_ImageInfo
jcu_encode_param_t	JCU_EncodeParam
jcu_async_status_t	JCU_AsyncStatus
jcu_internal_information_t	JCU_InternalInformation

Table 15　関数名一覧

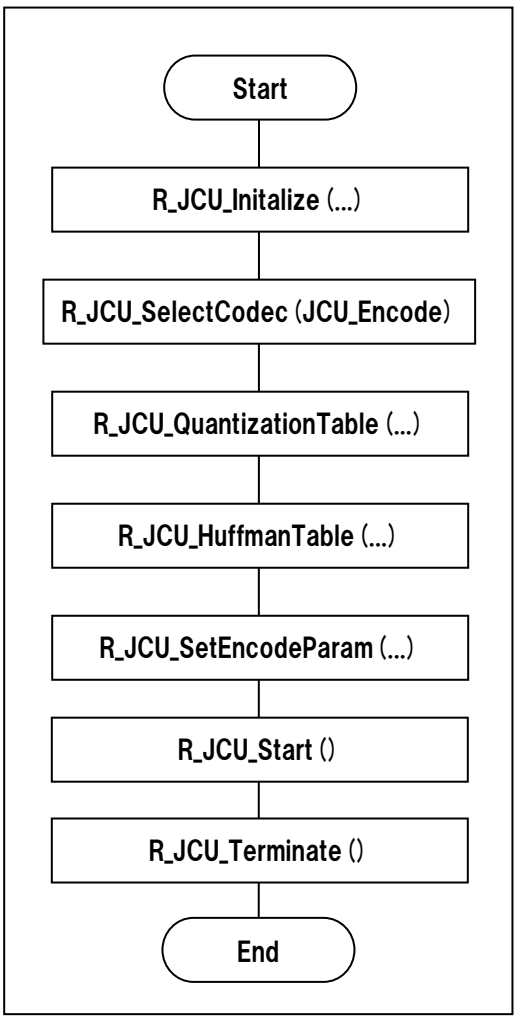
列挙型・変数型(new ver.)	列挙型・変数型(old ver.)
JCU_Initialize	R_JCU_Initialize
JCU_Terminate	R_JCU_Terminate
JCU_SelectCodec	R_JCU_SelectCodec
JCU_Start	R_JCU_Start
JCU_SetCountMode	R_JCU_SetCountMode
JCU_Continue	R_JCU_Continue
JCU_SetCallbackFunction	R_JCU_SetCallbackFunction*
JCU_SetDecodeParam	R_JCU_SetDecodeParam
JCU_GetImageInfo	R_JCU_GetImageInfo
JCU_GetErrorInfo	R_JCU_GetErrorInfo
JCU_SetQuantizationTable	R_JCU_SetQuantizationTable
JCU_SetHuffmanTable	R_JCU_SetHuffmanTable
JCU_GetEncodedSize	R_JCU_GetEncodedSize
JCU_SetEncodeParam	R_JCU_SetEncodeParam
JCU_TerminateAsync	R_JCU_TerminateAsync
JCU_GetAsyncStatus	R_JCU_GetAsyncStatus
JCU_StartAsync	R_JCU_StartAsync
JCU_SetPauseForImageInfo	R_JCU_SetPauseForImageInfo

*旧版の JCU_SetCallbackFunction 関数に相当する関数(R_JCU_SetCallbackFunction)は存在しません。使用していた場合は、使用例のフローチャートを参考にして、処理を変更して下さい。

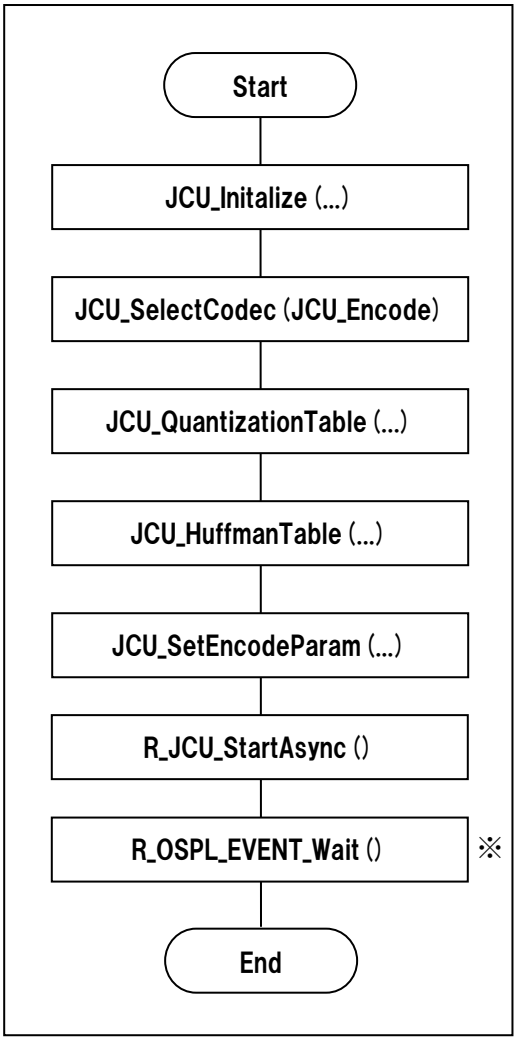
4. 使用例

本章では、ドライバを用いたときの圧縮・伸長のフローを紹介します。

4.1 圧縮（同期処理）時のドライバフローチャート

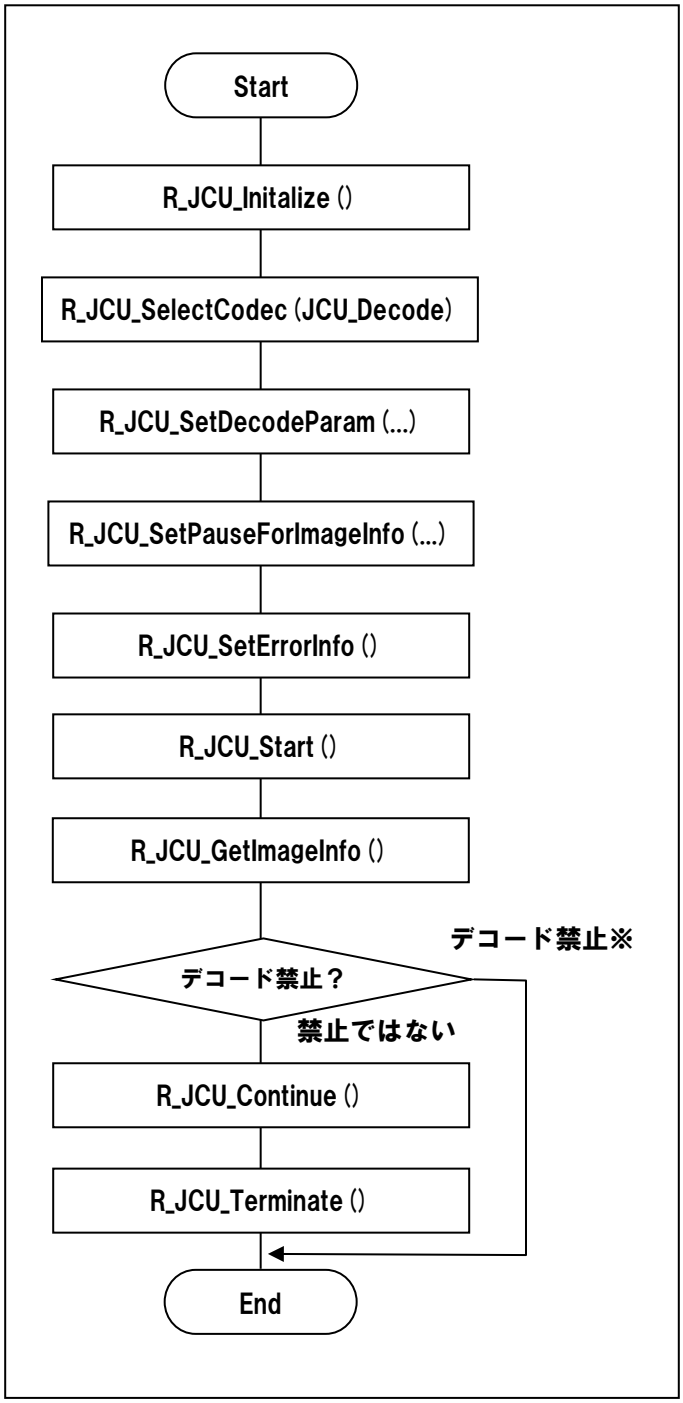


4.2　圧縮（非同期処理）時のドライバフローチャート



※ウェイト処理では、タイムアウト時間を設定する事が可能です。また、ウェイトせずにポーリングする事も可能です。

4.3　伸長（同期処理）時のドライバフローチャート



※JPEG 画像のフォーマットが、「YCbCr4:2:2」「YCbCr4:2:0」以外は、デコード禁止となります。また、JPEG 画像のサイズ（高さ、幅いずれか）がゼロの場合も、デコード禁止となります。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry

5. 改訂記録

Rev.	発行日	改訂内容
0.01	2010.10.28	初版発行
0.50	2011.02.01	構成見直しの為、全項目を修正
0.60	2012.09.12	ドライバー・バージョン 0.08 対応。
		JCU_Cull を JCU_SubSampling に変更。
		JCU_SetCountMode 関数に outputbuffer をカウントして分割する処理に対応していないことを明記。
0.69	2013.03.08	ドライバー・バージョン 0.09 対応。 JCU_GetInterruptEventFlags 関数を追加 JCU_Terminate と JCU_SetCallbackFunction を、どの状態でも呼び出せるように変更。 JCU_ERROR_LIMITATION、JCU_ERROR_INTERRUPT エラーの追加
1.00	2014.09.30	ドライバー・バージョン 0.10 対応 概要： ・OS 移植層への対応。デコード/エンコード完了を待つ手順に変更あり。 ・ネーミングルール変更。(“_UBYTE”→”uint8_t”など) ただし、旧型名・旧 API 関数名も使用可としている。 1.2 非対応の JPEG データデコード禁止を明記 1.3 ヘッダファイルの追加、変更。移植層追加。 1.6 状態遷移のルール変更。遷移表更新 INTERRUPTING 状態と INTERRUPTED 状態を追加。 1.7、1.8 割込みハンドラ関数名、登録方法変更 2.1 基本型、列挙型、変数型、構造体名の ネーミングルール変更。 OSPL の宣言型追加。 マクロ定義値を一部変更、追加、削除。 ドライバの内部状態管理用構造体変更（OSPL にて管理） 2.2 API 関数名のネーミングルール変更 R_JCU_TerminateAsync、R_JCU_StartAsync、 R_JCU_ContinueAsync、R_JCU_GetAsyncStatus、 R_JCU_OnInterrupting、R_JCU_OnInterrupted、 R_JCU_SetPauseForImageInfo、R_JCU_SetErrorFilter の各 API 関数を追加。 JCU_SetCallbackFunction、JCU_GetInterruptEventFlags の各 API 関数は削除 その他全体的に説明文を見直し。 2.2.1 旧 JCU_Initialize 関数から変更。ユーザ定義関数名を引数で指定せず固定（R_JCU_OnInitialize）名に変更。 2.2.2 旧 JCU_Terminate 関数から変更。ユーザ定義関数名を引数で指定せず固定（R_JCU_OnFinalize）名に変更。 2.2.4 旧 JCU_SelectCodec 関数から変更。SELECTED 状態からの実行を許可。 2.2.5 旧 JCU_SetCountMode 関数から変更。YCbCr4:2:0 の JPEG 画像をデコードした場合、指定したライン数の 2 倍ごとに分割される事を追記。 input と output の両方を同時に分割できない事を追記。 2.2.15,18 旧 JCU_SetDecodeParam、JCU_SetEncodeParam

		関数から変更。有効な割込みを指定する機能を削除。 2.2.16 旧 JCU_GetImageInfo 関数から変更。デコード禁止する条件を追記。 2.2.17 旧 JCU_GetErrorInfo 関数から変更。本関数は不要である（Ver0.9 までの JCU との互換性維持のためにある関数）事を明記。 3.1,2 ユーザ定義関数名を固定にして、デフォルトの関数を追加。 3.1.3 to 7 OSPL 用のユーザ定義関数を追加。デフォルトの関数を追加。通常は変更する必要は無いと明記。 4.1 to 3 OSPL 対応版のフローチャートに差替えた。 圧縮２件（同期、非同期）と伸長１件（同期）。

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部 ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

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。

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/

Colophon 3.0