RZ/A1H グループ

JPEG コーデックユニット(JCU) サンプルドライバ

R01AN1862JJ0201
Rev.2.01
2018.02.02

要旨

本アプリケーションノートでは、RZ/A1H の JPEG コーデックユニット機能を使用し、JPEG 圧縮画像のデコードおよび JPEG 圧縮画像へのエンコードを行うサンプルドライバについて説明します。

JPEG コーデックユニット(JCU) サンプルドライバの特長を以下に示します。

・JPEG 圧縮画像を、RGB565、ARGB8888、YCbCr422 形式の画像に変換します。

・YCbCr422 形式の画像を、JPEG 圧縮画像に変換します。

・変換完了は割込みで通知します。

対象デバイス

RZ/A1H グループ

RZ/A1M グループ

RZ/A1LU グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

制限事項

本 JCU ドライバのカウントモード(分割処理)は、使用禁止とします。もし使用する場合は、十分に評価してください。

目次

1. 仕様 5

2. 動作確認条件 6

3. 関連アプリケーションノート 7

4. 周辺機能説明 8

5. ハードウェア説明 9
5.1 ハードウェア構成例 9
5.2 使用端子一覧 10

6. ソフトウェア説明 11
6.1 動作概要 11
6.1.1 使用準備 12
6.2 メモリマップ 13
6.2.1 サンプルプログラムのセクション配置 14
6.2.2 MMU の設定 17
6.2.3 例外処理ベクタテーブル 18
6.3 使用割込み一覧 19
6.4 必要メモリサイズ 20
6.5 固定幅整数一覧 21
6.6 定数/エラーコード一覧 22
6.6.1 バージョン 22
6.6.2 errnum_t 22
6.6.3 jcu_errorcode_t 23
6.6.4 jcu_codec_t 23
6.6.5 jcu_continue_type_t 23
6.6.6 jcu_detail_error_t 23
6.6.7 jcu_int_detail_error_t 24
6.6.8 jcu_int_detail_errors_t 24
6.6.9 jcu_swap_t 24
6.6.10 jcu_sub_sampling_t 25
6.6.11 jcu_decode_format_t 25
6.6.12 jcu_jpeg_format_t 25
6.6.13 jcu_huff_t 25
6.6.14 jcu_table_no_t 25
6.6.15 jcu_status_information_t 26
6.6.16 jcu_sub_state_t 26
6.6.17 jcu_sub_status_t 27
6.6.18 jcu_codec_status_t 27
6.6.19 jcu_cbcr_offset_t 27
6.6.20 jcu_interrupt_line_t 27
6.6.21 jcu_interrupt_lines_t 27
6.6.22 その他 27
6.7 構造体/共用体一覧 29
6.7.1 jcu_count_mode_param_t 29
6.7.2 jcu_buffer_t 30
6.7.3 jcu_buffer_param_t 30
6.7.4 jcu_decode_param_t 30
6.7.5 jcu_image_info_t 30
6.7.6 jcu_encode_param_t 30
6.7.7 jcu_internal_information_t 31
6.7.8 jcu_async_status_t 31
6.7.9 jcu_context_t 32
6.7.10 jcu_thread_t 32

6.7.11 jcu_config_t 32
6.7.12 jcu_contexts_config_t 32
6.7.13 jcu_operator_config_t 33
6.8 変数一覧 34
6.9 関数一覧 35
6.10 関数仕様 39
6.10.1 R_JCU_Initialize 39
6.10.2 R_JCU_Terminate 39
6.10.3 R_JCU_TerminateAsync 39
6.10.4 R_JCU_TerminateEx 40
6.10.5 R_JCU_SelectCodec 40
6.10.6 R_JCU_SetCountMode 40
6.10.7 R_JCU_SetPauseForImageInfo 40
6.10.8 R_JCU_SetErrorFilter 40
6.10.9 R_JCU_Start 41
6.10.10 R_JCU_StartAsync 41
6.10.11 R_JCU_Continue 41
6.10.12 R_JCU_ContinueAsync 41
6.10.13 R_JCU_SetDecodeParam 42
6.10.14 R_JCU_GetImageInfo 42
6.10.15 R_JCU_SetEncodeParam 42
6.10.16 R_JCU_SetQuantizationTable 43
6.10.17 R_JCU_SetHuffmanTable 43
6.10.18 R_JCU_Set2ndCacheAttribute 43
6.10.19 R_JCU_GetEncodedSize 43
6.10.20 R_JCU_GetAsyncStatus 44
6.10.21 R_JCU_OnInterrupting 44
6.10.22 R_JCU_OnInterrupted 44
6.10.23 R_JCU_CONTEXTS_Initialize 44
6.10.24 R_JCU_OPERATOR_Thread 45
6.10.25 R_JCU_OPERATOR_Dispatch 45
6.10.26 R_JCU_OPERATOR_FinalizeStart 45
6.10.27 R_JCU_SetContextInInterrupt 45
6.10.28 R_JCU_OnInitialize 46
6.10.29 R_JCU_OnFinalize 46
6.10.30 R_JCU_SetDefaultAsync 46
6.10.31 R_JCU_SetInterruptCallbackCaller 46
6.10.32 R_JCU_OnEnableInterrupt 47
6.10.33 R_JCU_OnDisableInterrupt 47
6.10.34 R_JCU_OnInterruptDefault 47
6.10.35 R_JCU_CreateOperatorThread 48
6.10.36 R_JCU_DestroyOperatorThread 48
6.10.37 R_JCU_OnAllocateContext 48
6.10.38 R_JCU_OnFreeContext 48
6.11 補足 49
6.11.1 マルチスレッド対応があるときに、割込みから API を呼び出すときの注意点 49
6.11.2 フラグド構造体パラメータ 50

7. サンプルコード 51

8. 参考ドキュメント 51

ホームページとサポート窓口 52

改訂記録 53

製品ご使用上の注意事項 54

ご注意書き 55

1. 仕様

表 1-1 に使用する周辺機能と用途を、図 1.1 にサンプルコード実行時の動作環境を示します。

表 1-1　使用する周辺機能と用途

周辺機能	用途
JPEG コーデックユニット(JCU)	画像データの変換
割込みコントローラ(INTC)	画像データ変換の終了通知に使用
FIFO 内蔵シリアルコミュニケーションインターフェース(SCIF) Ch(2)	メッセージ表示

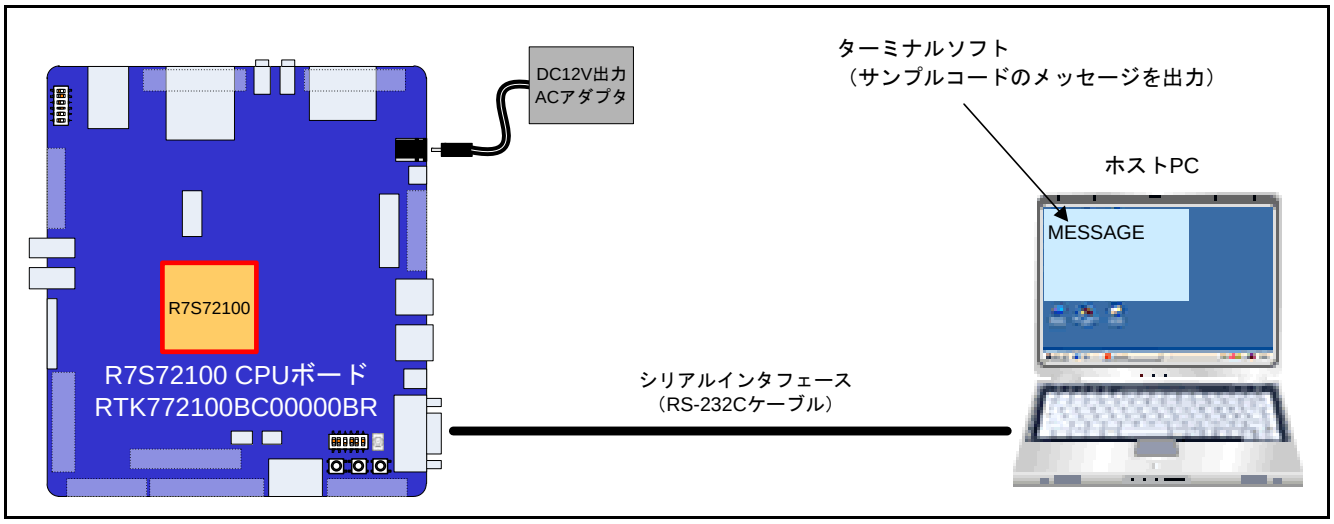

図 1.1　動作環境

2. 動作確認条件

本アプリケーションノートのサンプルコードは、下記の条件で動作を確認しています。

表 2-1　動作確認条件

項目		内容
使用マイコン		RZ/A1H
動作周波数		CPU クロック（Iφ）：400MHz 画像処理クロック（Gφ）：266.67MHz 内部バスクロック（Bφ）：133.33MHz 周辺クロック（P1φ）：66.67MHz 周辺クロック（P0φ）：33.33MHz
動作電圧		電源電圧（I/O）：3.3V 電源電圧（内部）：1.18V
ARM	統合開発環境	ARM®統合開発環境 ARM Development Studio 5（DS-5™）Version 5.16
	C コンパイラー	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
IAR	統合開発環境	IAR Embedded Workbench for ARM 7.80.4.12495
	C コンパイラー	
Renesas gcc	統合開発環境	e2 studio (Version: 5.3.0.023)
	C コンパイラー	GNUARM-NONE-EABI v16.01
動作モード		ブートモード 0 （CS0 空間 16 ビットブート）
ターミナルソフトの通信設定		・通信速度：115200bps ・データ長：8 ビット ・パリティ：なし ・ストップビット長：1 ビット ・フロー制御：なし
使用ボード		GENMAI ボード ・R7S72100 CPU ボード　RTK772100BC00000BR ・R7S72100 CPU ボード用オプションボード　RTK7721000B00000BR
使用デバイス （ボード上で使用する機能）		・LCD、シリアルインタフェース（Dsub-9 コネクタ） ※サンプルのみ使用。JCU ドライバは非使用。

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

RZ/A1H グループ 初期設定例（R01AN1646JJ）
RZ/A1H グループ レジスタ定義ヘッダ・ファイル iodefine.h（R01AN1860JJ）
RZ/A1Hグループ OS 移植層（OSPL）サンプルプログラム（R01AN1887JJ）

4. 周辺機能説明

JCU についての基本的な内容は、RZ/A1H グループ・ユーザーズマニュアル ハードウェア編に記載しています。

5.　ハードウェア説明

5.1　　ハードウェア構成例

図 5.1 にハードウェア構成例を示します。

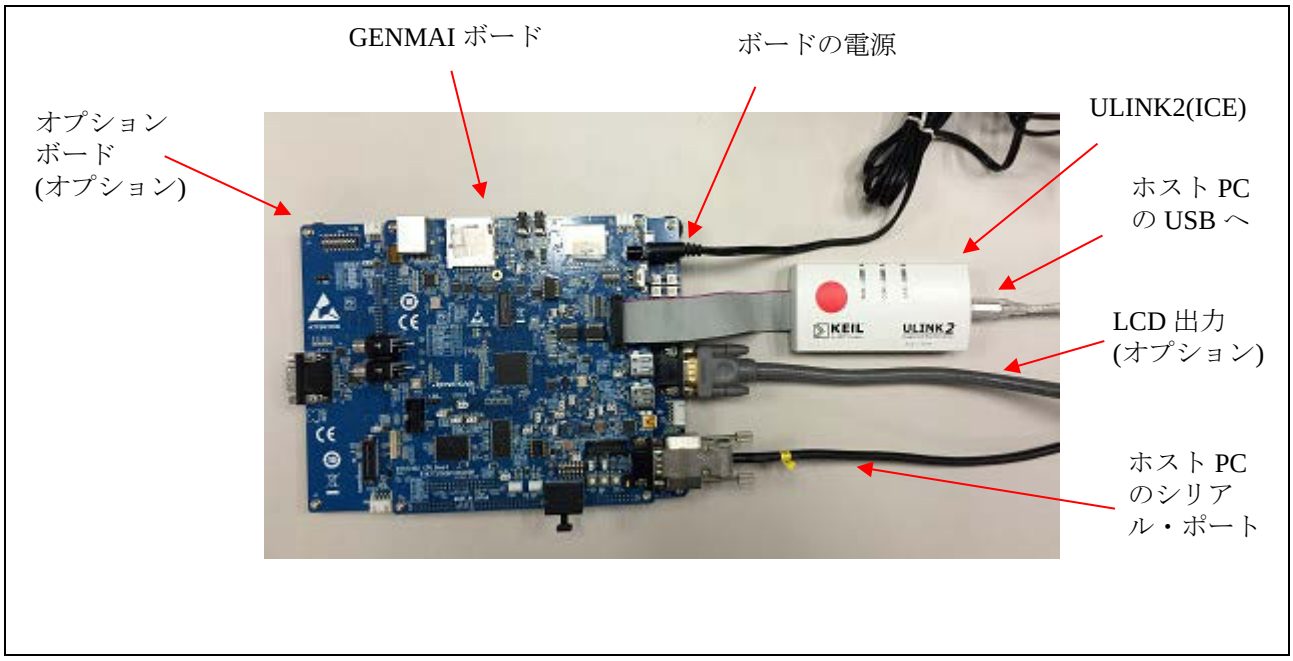

図 5.1　ハードウェア構成例

5.2　使用端子一覧

表 5-1 に使用端子と機能を示します。

表 5-1　使用端子と機能

端子名	入出力	内容
なし		

6. ソフトウェア説明

6.1 動作概要

[動作の説明]

図 6.1 に同期関数を使用して画像データを変換するシーケンスを示します。

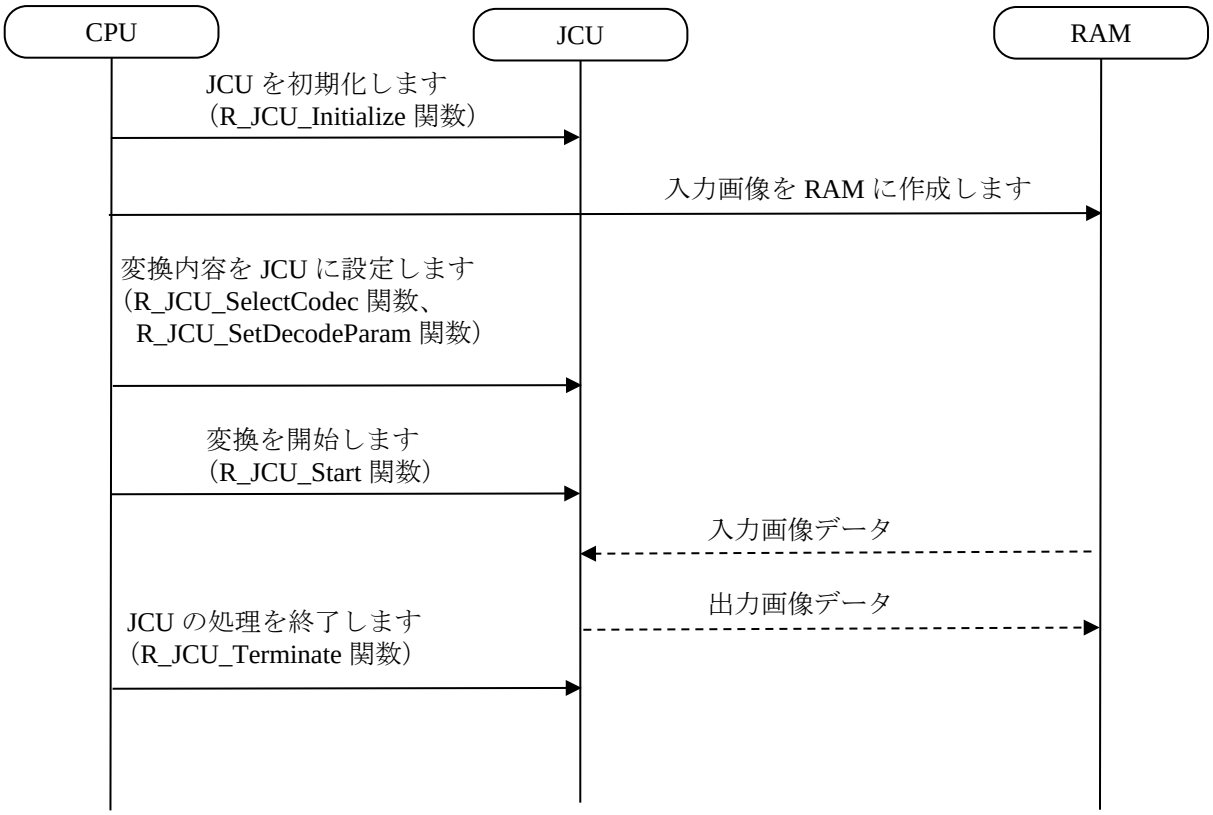

図 6.1　JCU のシーケンス（同期関数を使用）

非同期関数を使用する場合のシーケンスについては、RZ/A1H グループ PFV,JCU 向け OS 移植層 OSPL（R01AN1887JJ）を参照してください。

サンプルプログラムとしては、「JPEG 画像のデコード」処理（R_JCU_SampleDecode 関数）、JPEG 画像のデコードとエンコード処理（R_JCU_SampleDecodeEncode 関数）、JPEG 画像のデコード後に表示する処理（R_JCU_SampleDecodeAndShow 関数）の 3 種類の処理があります。

6.1.1　使用準備

本サンプルプログラムの実行準備を説明します。

1)　ホスト PC にてターミナルソフトを起動し、次のように設定します。(Tera Term の場合)

図 6.2　シリアル・ポートの設定

2)　サンプルプログラムを実行すると、次のようにターミナルでメッセージを出力します。

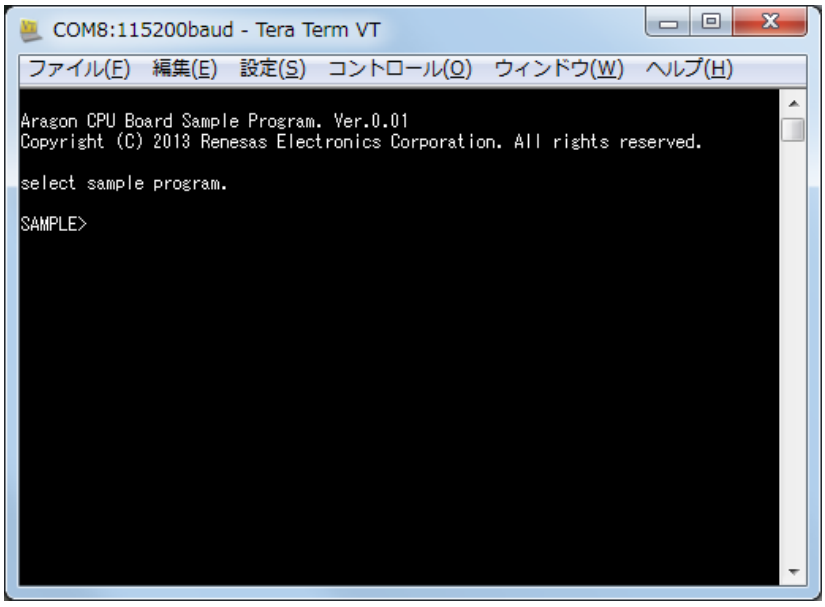

図 6.3　サンプルプログラム実行時のメッセージ出力

6.2　メモリマップ

図 6.4 に、RZ/A1H グループのアドレス空間と GENMAI ボードのメモリマッピングを示します。

参考プログラムでは、ROM 領域を使用するコードおよびデータを CS0 空間に接続した NOR フラッシュメモリに配置し、RAM 領域を使用するコードおよびデータを大容量内蔵 RAM に配置するようにしています。

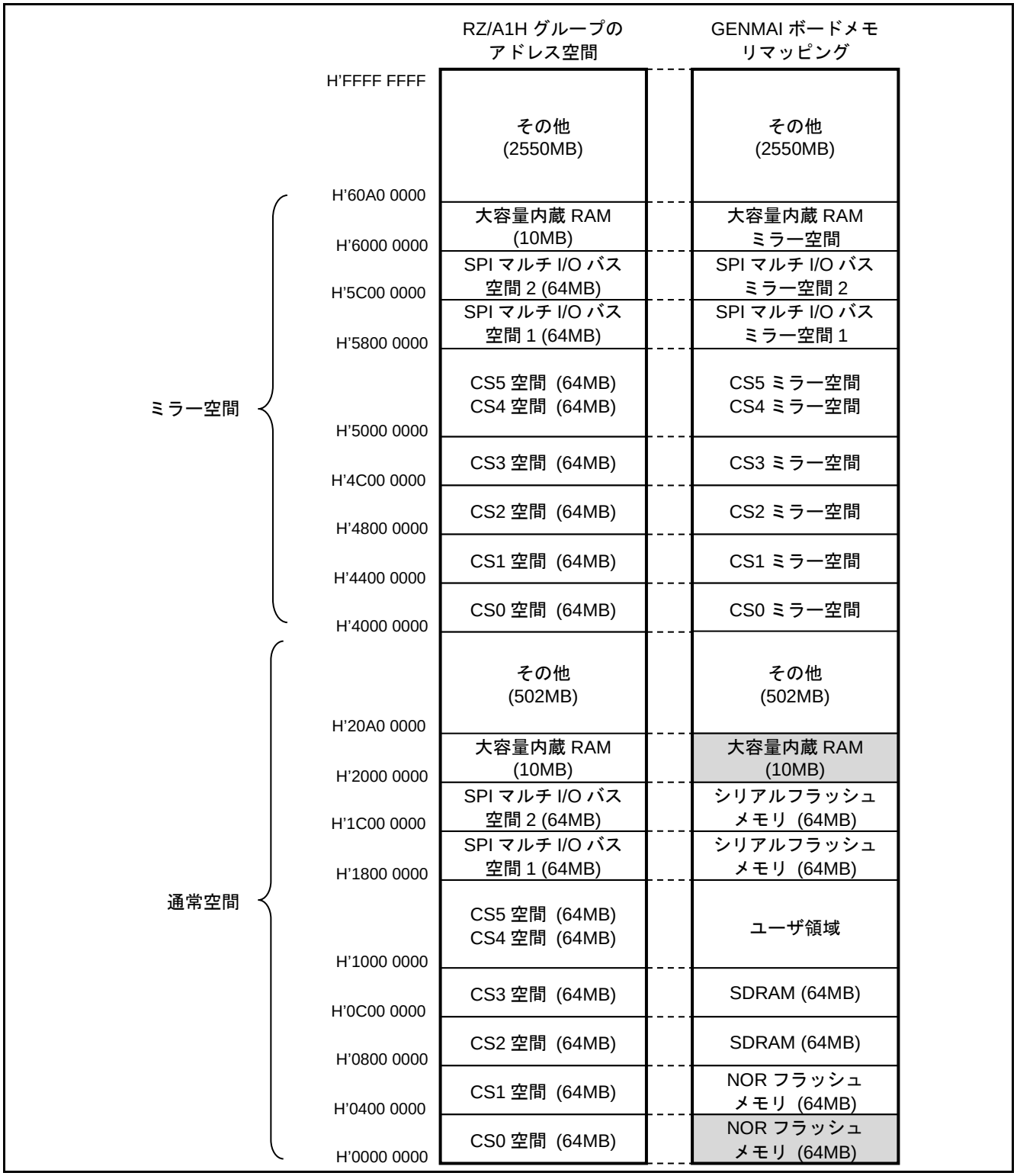

図 6.4　メモリマップ

6.2.1　サンプルプログラムのセクション配置

サンプルプログラムでは、割込み処理の高速化のため、例外処理ベクタテーブルと IRQ 割込みハンドラを大容量内蔵 RAM 上に配置して、これらの処理を大容量内蔵 RAM 上で実行するようにしています。例外処理ベクタテーブルおよび IRQ 割込みハンドラのプログラムコードの NOR フラッシュメモリ領域から大容量内蔵 RAM 領域への転送処理、初期値なしデータセクションのゼロクリア処理、および初期値ありデータセクションの初期化処理は、スキャッタローディング機能を使用して行っています。スキャッタローディング機能の詳細は、ARM より提供される「ARM コンパイラツールチェーン／リンカの使用／イメージの構造と生成の章」を参照してください。

表 6-1 に参考プログラムで使用するセクションを示し、図 6.5 にサンプルプログラムの初期状態のセクション配置（ロードビュー）と、スキャッタローディング機能を使用後のセクション配置（実行ビュー）を示します。

表 6-1　使用するセクション(1/2)

領域の名前	内容	タイプ	ロード領域	実行領域
VECTOR_TABLE	例外処理ベクタテーブル	Code	FLASH	FLASH
RESET_HANDLER	リセットハンドラ処理のプログラムコード領域 この領域は以下のセクションから構成されています ・INITCA9CACHE（L1 キャッシュ設定） ・INIT_TTB（MMU 設定） ・RESET_HANDLER（リセットハンドラ）	Code	FLASH	FLASH
CODE_BASIC_SETUP	動作周波数とフラッシュメモリ最適化のためのプログラムコード領域	Code	FLASH	FLASH
InRoot	この領域は C 標準ライブラリなどのルート領域に配置するセクションから構成されています	Code および RO Data	FLASH	FLASH
CODE_FPU_INIT	NEON および VFP 初期設定のプログラムコード領域 この領域は以下のセクションから構成されています ・CODE_FPU_INIT ・FPU_INIT	Code	FLASH	FLASH
CODE_RESET	ハードウェア初期設定のプログラムコード領域 この領域は以下のセクションから構成されています ・CODE_RESET（スタートアップ処理） ・INIT_VBAR（ベクタベース設定）	Code	FLASH	FLASH
CODE_IO_REGRW	IO レジスタのリード/ライト関数のプログラムコード領域	Code	FLASH	FLASH
CODE	デフォルトのプログラムコード領域 C ソースでセクション名を定義しない Code タイプのセクションは、すべてこの領域に配置されます	Code	FLASH	FLASH

表 6-2　使用するセクション(2/2)

領域の名前	内容	タイプ	ロード領域	実行領域
CONST	デフォルトの定数データ領域 C ソースでセクション名を定義しない RO Data タイプのセクションは、すべてこの領域に配置されます	RO Data	FLASH	FLASH
VECTOR_MIRROR_TABLE	例外処理ベクタテーブル （大容量内蔵 RAM に転送して実行するためのセクション）	Code	FLASH	LRAM
CODE_HANDLER_JMPTBL	IRQ 割込みハンドラのユーザ定義関数のプログラムコード領域	Code	FLASH	LRAM
CODE_HANDLER	IRQ 割込みハンドラのプログラムコード領域 この領域は以下のセクションから構成されています ・CODE_HANDLER ・IRQ_FIQ_HANDLER	Code	FLASH	LRAM
DATA_HANDLER_JMPTBL	IRQ 割込みハンドラのユーザ定義関数の登録テーブルデータ領域	RW Data	FLASH	LRAM
ARM_LIB_STACK	アプリケーションスタック領域	ZI Data	－	LRAM
IRQ_STACK	IRQ モードのスタック領域	ZI Data	－	LRAM
FIQ_STACK	FIQ モードのスタック領域	ZI Data	－	LRAM
SVC_STACK	スーパバイザ（SVC）モードのスタック領域	ZI Data	－	LRAM
ABT_STACK	アボート（ABT）モードのスタック領域	ZI Data	－	LRAM
TTB	MMU 変換テーブル領域	ZI Data	－	LRAM
ARM_LIB_HEAP	アプリケーションヒープ領域	ZI Data	－	LRAM
DATA	デフォルトの初期値ありデータ領域 C ソースでセクション名を定義しない RW Data タイプのセクションは、すべてこの領域に配置されます	RW Data	FLASH	LRAM
BSS	デフォルトの初期値なしデータ領域 C ソースでセクション名を定義しない ZI Data タイプのセクションは、すべてこの領域に配置されます	ZI Data	－	LRAM

【注】 1. 表中のロード領域および実行領域において、FLASH は NOR フラッシュメモリの領域を、LRAM は大容量内蔵 RAM の領域を表します。
2. セクションの名前は基本的に領域と同じ名前にしていますが、RESET_HANDLER、InRoot、CODE_FPU_INIT、CODE_RESET、CODE、CONST、CODE_HANDLER、DATA、BSS の各領域は複数のセクションから構成されています。領域とセクションについては、ARM コンパイラツールチェーンのマニュアルを参照してください。

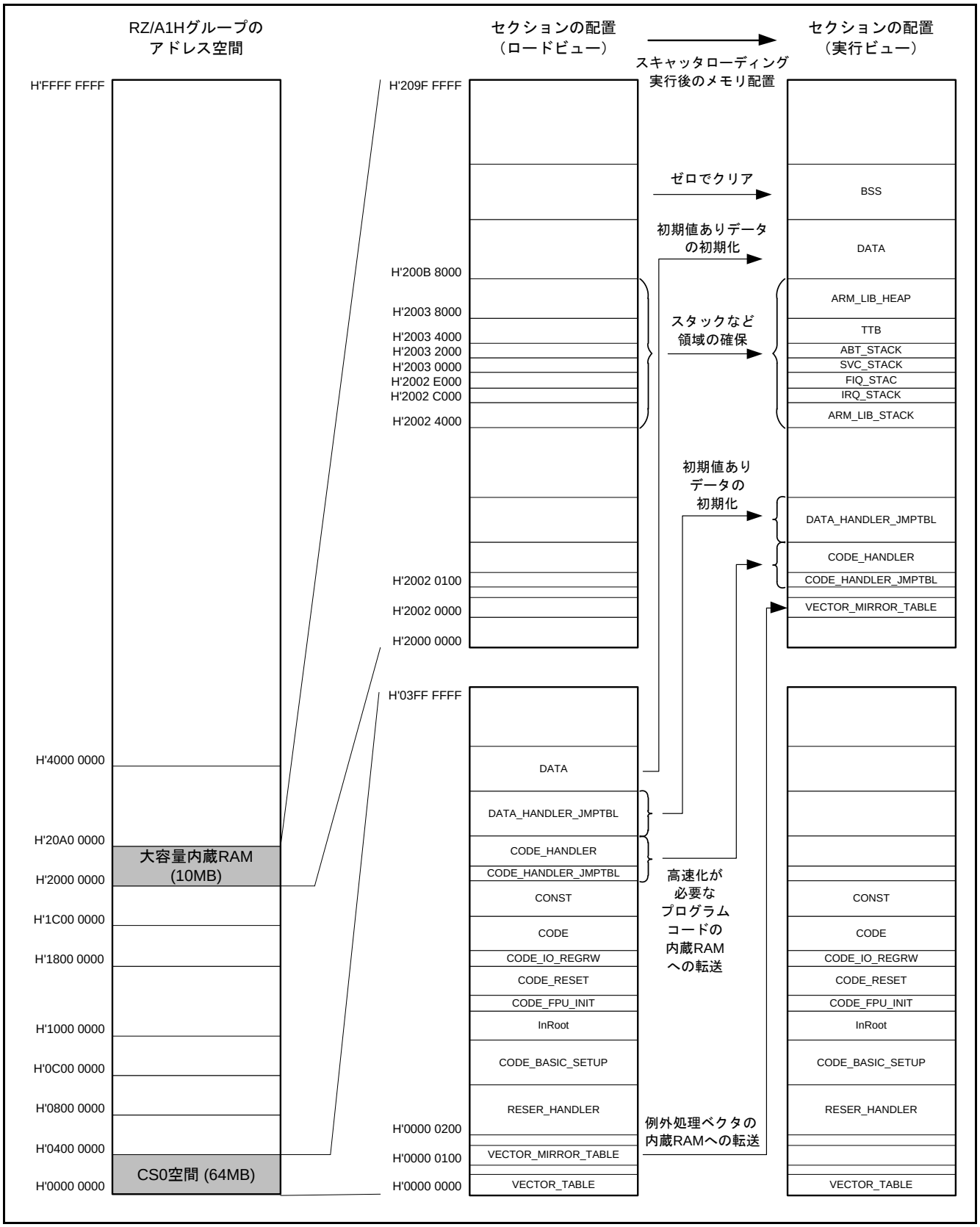
RZ/A1Hグループの
アドレス空間
セクションの配置
（ロードビュー）
スキャッタローディング
実行後のメモリ配置
セクションの配置
（実行ビュー）
H'FFFF FFFF
H'4000 0000
H'20A0 0000
大容量内蔵RAM
(10MB)
H'2000 0000
H'1C00 0000
H'1800 0000
H'1000 0000
H'0C00 0000
H'0800 0000
H'0400 0000
CS0空間 (64MB)
H'0000 0000
H'209F FFFF
H'200B 8000
H'2003 8000
H'2003 4000
H'2003 2000
H'2003 0000
H'2002 E000
H'2002 C000
H'2002 4000
H'2002 0100
H'2002 0000
H'2000 0000
H'03FF FFFF
DATA
DATA_HANDLER_JMPTBL
CODE_HANDLER
CODE_HANDLER_JMPTBL
CONST
CODE
CODE_IO_REGRW
CODE_RESET
CODE_FPU_INIT
InRoot
CODE_BASIC_SETUP
RESER_HANDLER
H'0000 0200
VECTOR_MIRROR_TABLE
H'0000 0100
VECTOR_TABLE
H'0000 0000
ゼロでクリア
初期値ありデータ
の初期化
スタックなど
領域の確保
初期値あり
データの
初期化
高速化が
必要な
プログラム
コードの
内蔵RAM
への転送
例外処理ベクタの
内蔵RAMへの転送
BSS
DATA
ARM_LIB_HEAP
TTB
ABT_STACK
SVC_STACK
FIQ_STAC
IRQ_STACK
ARM_LIB_STACK
DATA_HANDLER_JMPTBL
CODE_HANDLER
CODE_HANDLER_JMPTBL
VECTOR_MIRROR_TABLE
CONST
CODE
CODE_IO_REGRW
CODE_RESET
CODE_FPU_INIT
InRoot
CODE_BASIC_SETUP
RESER_HANDLER
VECTOR_TABLE

図 6.5　セクション配置

6.2.2　MMU の設定

RZ/A1H GENMAI ボードで使用するハードウェア資源のメモリマップにあわせて、H'0000 0000 番地から 1MB 単位で 4GB の領域を MMU で管理するように設定しています（ttb_init.s ファイルで設定しています）。システムにあわせてカスタマイズする場合は、最少単位を 1MB としてください。

表 6-3 に、サンプルプログラムの MMU の設定を示します。

表 6-3　MMU の設定

定義名	内容	アドレス	サイズ	メモリタイプ
M_SIZE_NOR	CS0、CS1 空間（NOR フラッシュメモリ）	H'0000 0000 ～ H'07FF FFFF	128MB	L1 キャッシュ有効、ノーマルメモリ
M_SIZE_SDRAM	CS2、CS3 空間（SDRAM）	H'0800 0000 ～ H'0FFF FFFF	128MB	L1 キャッシュ有効、ノーマルメモリ
M_SIZE_CS45	CS4、CS5 空間	H'1000 0000 ～ H'17FF FFFF	128MB	ストロングリオーダメモリ（L1 キャッシュ無効）
M_SIZE_SPI	SPI マルチ IO バス空間 1 および 2（シリアルフラッシュメモリ）	H'1800 0000 ～ H'1FFF FFFF	128MB	L1 キャッシュ有効、ノーマルメモリ
M_SIZE_RAM	大容量内蔵 RAM 空間	H'2000 0000 ～ H'209F FFFF	10MB	L1 キャッシュ有効、ノーマルメモリ
M_SIZE_IO_1	内蔵周辺モジュールおよび予約エリア	H'20A0 0000 ～ H'3FFF FFFF	502MB	ストロングリオーダメモリ（L1 キャッシュ無効）
M_SIZE_NOR_M	CS0、CS1 ミラー空間	H'4000 0000 ～ H'47FF FFFF	128MB	L1 キャッシュ無効、ノーマルメモリ
M_SIZE_SDRAM_M	CS2、CS3 ミラー空間	H'4800 0000 ～ H'4FFF FFFF	128MB	L1 キャッシュ無効、ノーマルメモリ
M_SIZE_CS45_M	CS4、CS5 ミラー空間	H'5000 0000 ～ H'57FF FFFF	128MB	ストロングリオーダメモリ（L1 キャッシュ無効）
M_SIZE_SPI_M	SPI マルチ IO バスミラー空間 1 および 2	H'5800 0000 ～ H'5FFF FFFF	128MB	L1 キャッシュ無効、ノーマルメモリ
M_SIZE_RAM_M	大容量内蔵 RAM ミラー空間	H'6000 0000 ～ H'609F FFFF	10MB	L1 キャッシュ無効、ノーマルメモリ
M_SIZE_IO_2	内蔵周辺モジュールおよび予約エリア	H'60A0 0000 ～ H'FFFF FFFF	2550MB	ストロングリオーダメモリ（L1 キャッシュ無効）

6.2.3　例外処理ベクタテーブル

RZ/A1H には 7 種類の例外処理（リセット、未定義命令、ソフトウェア割込み、プリフェッチアボート、データアボート、IRQ、FIQ）があり、ブートモード 0 の場合、リセット解除後に例外処理のベクタテーブルは H'0000 0000 番地から 32 バイトの領域（H'0000 0000 番地～H'0000 001F 番地）に配置されます。例外処理のベクタテーブルには、各例外処理への分岐命令を記述します。

図 6.6 に例外処理ベクタテーブルの記述例として、サンプルコードの例外処理ベクタテーブル内容を示します。

```
vector_table
  LDR pc, =reset_handler       ; 0x0000_0000 : Reset exception
  LDR pc, =undefined_handler   ; 0x0000_0004 : Undefined instructions
exception
  LDR pc, =svc_handler         ; 0x0000_0008 : Software interrupts exceptions
  LDR pc, =prefetch_handler    ; 0x0000_000c : Prefetch abort exception
  LDR pc, =abort_handler       ; 0x0000_0010 : Data abort exception
  LDR pc, =reserved_handler    ; 0x0000_0014 : Reserved
  LDR pc, =irq_handler         ; 0x0000_0018 : IRQ exception
  LDR pc, =fiq_handler         ; 0x0000_001c : FIQ exception
```

図 6.6　例外処理ベクタテーブルの記述例

6.3　使用割込み一覧

表 6.4 にサンプルコードで使用する割込みを示します。

表 6.4　サンプルコードで使用する割込み

割込み(要因 ID)	優先度	処理概要
JEDI	JCU_INT_PRI(=2)	圧縮伸長処理
JDTI	JCU_INT_PRI(=2)	データ転送処理

6.4　必要メモリサイズ

表 6.5 に、サンプル プログラムの実行に必要なメモリサイズを示します。

表 6.5　必要メモリサイズ

使用メモリ	サイズ（バイト）	備考
ROM	11912	JCU - Code
ROM	68	JCU - Read Only Data
ROM, RAM	16	JCU - Read Write Data
RAM	76	JCU - Zero Initialized Data
RAM	1060	サンプル プログラムのスタック

【注】 必要メモリサイズは C コンパイラーのバージョンやコンパイルオプションにより異なります。

6.5　固定幅整数一覧

表 6-6 にサンプルコードで使用する固定幅整数を示します。

表 6-6　サンプルコードで使用する固定幅整数

シンボル	内容
char_t	8 ビット文字
bool_t	論理型。値は true(=1), false(=0)
int_t	高速な整数、符号あり、本サンプルコードでは 32 ビット整数。
int8_t	8 ビット整数、符号あり
int16_t	16 ビット整数、符号あり
int32_t	32 ビット整数、符号あり
uint8_t	8 ビット整数、符号なし
uint16_t	16 ビット整数、符号なし
uint32_t	32 ビット整数、符号なし

6.6　定数/エラーコード一覧

表 6-7　サンプルコードで使用する定数

章	型名	内容
6.6.1	-	バージョン。
6.6.2	errnum_t	エラー情報。
6.6.3	jcu_errorcode_t	エラーコード。0=エラー無し
6.6.4	jcu_codec_t	モードの指定(圧縮・伸長)
6.6.5	jcu_continue_type_t	JCU が中断された場合に、再開するモード
6.6.6	jcu_detail_error_t	JCU の出力したエラー
6.6.7	jcu_int_detail_error_t	デコードエラーの詳細情報
6.6.8	jcu_int_detail_errors_t	jcu_int_detail_error_t 型の値のビット・フラグ値の論理和
6.6.9	jcu_swap_t	データのスワップ設定
6.6.10	jcu_sub_sampling_t	伸長時の出力画像の間引き設定
6.6.11	jcu_decode_format_t	伸長時の出力画像のピクセルフォーマット
6.6.12	jcu_jpeg_format_t	JPEG 画像のピクセルフォーマット
6.6.13	jcu_huff_t	ハフマンテーブルの AC/DC 成分
6.6.14	jcu_table_no_t	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
6.6.15	jcu_status_information_t	ドライバのステータス
6.6.16	jcu_sub_state_t	ドライバのサブステータス
6.6.17	jcu_sub_status_t	jcu_sub_state_t 型の値のビット・フラグ値の論理和
6.6.18	jcu_codec_status_t	現在の処理
6.6.19	jcu_cbcr_offset_t	Cb/Cr の範囲。ピクセルフォーマットが YCbCr 以外の時は、JCU_CBCR_OFFSET_0 を設定してください。
6.6.20	jcu_interrupt_line_t	割込み線の種類。
6.6.21	jcu_interrupt_lines_t	jcu_interrupt_line_t 型の値のビット・フラグ値の論理和
6.6.22	-	その他

6.6.1　バージョン

表 6-8 バージョン情報

定数名	設定値	内容
JCU_VERSION	200	JCU のバージョン番号
JCU_VERSION_STRING	"2.00"	JCU のバージョン番号の文字列

6.6.2　errnum_t

表 6-9 エラー情報

定数名	設定値	内容
0	0	エラーなし
E_OTHERS	1	その他のエラー
E_FEW_ARRAY	2	固定長配列が不足したときのエラー
E_FEW_MEMORY	3	ヒープメモリー不足
E_FIFO_OVER	4	キューに入りきらなかったエラー

E_NOT_FOUND_SYMBOL	5	シンボルが定義されていない
E_NO_NEXT	6	次のリスト要素がない
E_ACCESS_DENIED	7	読み込み書き込み拒否エラー
E_NOT_IMPLEMENT_YET	9	未実装
E_ERRNO	0x0E(=14)	errno を参照
E_LIMITATION	0x0F(=15)	暫定の制限事項
E_STATE	0x10(=16)	現在の状態では実行できないというエラー
E_NOT_THREAD	0x11(=17)	スレッドではないエラー、割込みコンテキストから呼べない
E_PATH_NOT_FOUND	0x12(=18)	ファイルやフォルダーが見つからない
E_BAD_COMMAND_ID	0x16(=22)	コマンド ID が範囲外
E_TIME_OUT	0x17(=23)	タイムアウト
E_STACK_OVERFLOW	0x1C(=28)	スタック オーバーフロー
E_NO_DEBUG_TLS	0x1D(=29)	デバッグ用ワーク領域がない
E_EXIT_TEST	0x1E(=30)	テストの停止要求

6.6.3　jcu_errorcode_t

表 6-10 エラーコード

定数名	設定値	内容
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

6.6.4　jcu_codec_t

表 6-11 圧縮・伸長の指定

定数名	設定値	内容
JCU_ENCODE	0	圧縮処理
JCU_DECODE	1	伸長処理
JCU_NOT_SELECTED	2	未指定状態

6.6.5　jcu_continue_type_t

表 6-12 JCU が中断された場合に、再開するモード

定数名	設定値	内容
JCU_INPUT_BUFFER	0	入力バッファの一時停止を再開する
JCU_OUTPUT_BUFFER	1	出力バッファの一時停止を再開する
JCU_GET_IMAGE_INFO	2	画像情報取得後の一時停止を再開する

6.6.6　jcu_detail_error_t

表 6-13 JCU の出力したエラー

定数名	設定値	内容

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	対象外のピクセルフォーマットを検出
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	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 *1
JCU_JCDERR_LAST_MCU_DATA	0x452D	最終 MCU データ数エラーを検出
JCU_JCDERR_BLOCK_DATA	0x452E	ブロックデータ数エラーを検出

6.6.7　jcu_int_detail_error_t

表 6-14 デコードエラー検出の種類

定数名	設定値	内容
JCU_INT_ERROR_RESTART_INTERVAL_DATA	0x80	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
JCU_INT_ERROR_SEGMENT_TOTAL_DATA	0x40	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
JCU_INT_ERROR_MCU_BLOCK_DATA	0x20	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

6.6.8　jcu_int_detail_errors_t

表 6-15 jcu_int_detail_error_t 型の値のビット・フラグ値の論理和

定数名	設定値	内容
JCU_INT_ERROR_ALL	0xE0	すべてのエラー

6.6.9　jcu_swap_t

表 6-16 データのスワップ設定

定数名	設定値	内容
JCU_SWAP_NONE	0x00	スワップなし

*1 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。

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

6.6.10　jcu_sub_sampling_t

表 6-17 伸長時の出力画像の間引き設定

定数名	設定値	内容
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

6.6.11　jcu_decode_format_t

表 6-18 伸長時の出力画像のピクセルフォーマット

定数名	設定値	内容
JCU_OUTPUT_YCbCr422	0x00	YCbCr422
JCU_OUTPUT_ARGB8888	0x01	ARGB8888
JCU_OUTPUT_RGB565	0x02	RGB565

6.6.12　jcu_jpeg_format_t

表 6-19 JPEG 画像のピクセルフォーマット

定数名	設定値	内容
JCU_JPEG_YCbCr444	0x00	YCbCr444
JCU_JPEG_YCbCr422	0x01	YCbCr422
JCU_JPEG_YCbCr420	0x02	YCbCr420
JCU_JPEG_YCbCr411	0x06	YCbCr411

6.6.13　jcu_huff_t

表 6-20 ハフマンテーブルの AC/DC 成分

定数名	設定値	内容
JCU_HUFFMAN_AC	0x00	AC 成分を示します
JCU_HUFFMAN_DC	0x01	DC 成分を示します

6.6.14　jcu_table_no_t

表 6-21 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

定数名	設定値	内容
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

6.6.15 jcu_status_information_t

表 6-22 ドライバのステータス

定数名	設定値	内容
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 次、または、現在実行する関数が、 R_JCU_OnInterrupted

6.6.16 jcu_sub_state_t

表 6-23 ドライバのサブステータス

定数名	設定値	内容
JCU_SUB_INFOMATION_READY	0x0008	画像サイズ／ピクセルフォーマットを読み出し可能なとき 1 に設定されます。
JCU_SUB_DECODE_OUTPUT_PAUSE	0x0100	伸長時、出力画像データを jcu_count_mode_param_t::outputBuffer.dataCount に示すライン数分書き出したとき、1 に設定されます。
JCU_SUB_DECODE_INPUT_PAUSE	0x0200	伸長時、入力符号データを jcu_count_mode_param_t::inputBuffer.dataCount に示すデータ数分読み出したとき、1 に設定されます。
JCU_SUB_ENCODE_OUTPUT_PAUSE	0x1000	圧縮時、出力符号データを jcu_count_mode_param_t::outputBuffer.dataCount に示すデータ数分書き出したとき、1 に設定されます。

JCU_SUB_ENCODE_INPUT_PAUSE	0x2000	圧縮時、入力画像データを jcu_count_mode_param_t::inputBuffer.dataCount に示すライン数分読み出したとき、1 に設定されます。

6.6.17　jcu_sub_status_t

表 6-24 jcu_sub_state_t 型の値のビット・フラグ値の論理和

定数名	設定値	内容
JCU_SUB_PAUSE_ALL	0x3308	一時停止のすべての要因

6.6.18　jcu_codec_status_t

表 6-25 現在の処理

定数名	設定値	内容
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

6.6.19　jcu_cbcr_offset_t

表 6-26 Cb/Cr の範囲

定数名	設定値	内容
JCU_CBCR_OFFSET_0	0	Cb/Cr の範囲が-128~127
JCU_CBCR_OFFSET_128	1	Cb/Cr の範囲が 0~255

ピクセルフォーマットが YCbCr 以外の場合は、JCU_CBCR_OFFSET_0 を使用すること。

6.6.20　jcu_interrupt_line_t

表 6-27 割込み線の種類。ビット・フラグ値として扱う。

定数名	設定値	内容
JCU_INTERRUPT_LINE_JEDI	0x00000001u	JEDI 割込み
JCU_INTERRUPT_LINE_JDTI	0x00000002u	JDTI 割込み

6.6.21　jcu_interrupt_lines_t

表 6-28 jcu_interrupt_line_t 型の値のビット・フラグ値の論理和

定数名	設定値	内容
JCU_INTERRUPT_LINE_ALL	0x00000003u	JEDI、JDTI 両方の割込み

6.6.22　その他

表 6-29　その他

定数名	設定値	内容
JCU_NUMBER_OF_QUANTIZATION_TABLE_DATA	64	量子化テーブルのサイズ
JCU_NUMBER_OF_HUFFMAN_TABLE_DATA_DC	28	ハフマンテーブルの DC 成分のサイズ
JCU_NUMBER_OF_HUFFMAN_TABLE_DATA_AC	178	ハフマンテーブルの AC 成分のサイズ
JCU_MULTI_THREAD	0 or 1	マルチスレッド対応=1、非対応=0

6.7　構造体/共用体一覧

表 6-30　構造体/共用体一覧

章	型名	概要
6.7.1	jcu_count_mode_param_t	カウントモード(分割処理)を行うためのパラメータ
6.7.2	jcu_buffer_t	入出力バッファを設定するための構造体
6.7.3	jcu_buffer_param_t	デコード用の入出力バッファを設定するためのパラメータ
6.7.4	jcu_decode_param_t	デコード用のオプションを設定するためのパラメータ
6.7.5	jcu_image_info_t	デコードした画像の情報を取得するための構造体
6.7.6	jcu_encode_param_t	エンコード用のオプションを設定するためのパラメータ
6.7.7	jcu_internal_information_t	ドライバ内部の状態を設定するための構造体
6.7.8	jcu_async_status_t	状態と割込みステータス
6.7.9	jcu_context_t	コンテキスト
6.7.10	jcu_thread_t	スレッドに関する情報
6.7.11	jcu_config_t	初期化・終了するときのパラメータ
6.7.12	jcu_contexts_config_t	R_JCU_CONTEXTS_Initialize のパラメータ
6.7.13	jcu_operator_config_t	制御スレッドのパラメータ

6.7.1　jcu_count_mode_param_t

概　要　　カウントモード(分割処理)を行うためのパラメータ

ヘッダ　　r_jcu_api.h

説　明

メンバー変数

bool_t　inputBuffer. isEnable	false:　入力バッファの分割処理を行いません。 true:　入力バッファの分割処理を行います。
bool_t　inputBuffer. 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。
uint32_t*　inputBuffer. restartAddress	inputBuffer. isEnable が true の時の、初期化アドレスを指定します。
uint32_t　inputBuffer. dataCount	入力バッファの分割サイズを指定します。 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。 指定できる値は、８の倍数です。
bool_t　outputBuffer. isEnable	false:　出力バッファの分割処理を行いません。 true:　出力バッファの分割処理を行います。
bool_t　outputBuffer. isInitAddress	false: 一時停止時に、出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、出力バッファのアドレスを初期化します。
uint32_t*　outputBuffer. restartAddress	outputBuffer. isEnable が true の時の、初期化アドレスを指定します。
uint32_t　outputBuffer. dataCount	出力バッファの分割サイズを指定します。

	デコードする場合は、指定したライン数（YCbCr420 のデータをデコードした場合は、指定したライン数の 2 倍）のデータを JCU が出力すると、一時停止します。 エンコードする場合は、指定したバイト数のデータを JCU が出力すると、一時停止します。 指定できる値は、8 の倍数です。

6.7.2 jcu_buffer_t

概　要　入出力バッファを設定するための構造体
ヘッダ　r_jcu_api.h
説　明
メンバー変数

jcu_swap_t swapSetting	データのスワップ方法を選択します。
uint32_t*　address	バッファのアドレスを選択します。

6.7.3 jcu_buffer_param_t

概　要　デコード用の入出力バッファを設定するためのパラメータ
ヘッダ　r_jcu_api.h
説　明
メンバー変数

jcu_buffer_t　source	入力バッファの設定を行います。
jcu_buffer_t　destination	出力バッファの設定を行います。
int16_t　lineOffset	フレームバッファのオフセットを選択します。

6.7.4 jcu_decode_param_t

概　要　デコード用のオプションを設定するためのパラメータ
ヘッダ　r_jcu_api.h
説　明
メンバー変数

jcu_sub_sampling_t verticalSubSampling	垂直方向の間引き設定を行います
jcu_sub_sampling_t horizontalSubSampling	水平方向の間引き設定を行います
jcu_decode_format_t 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
jcu_cbcr_offset_t outputCbCrOffset	出力画像データの Cb/Cr 範囲設定を行います。 YCbCr 以外のときは JCU_CBCR_OFFSET_0 を設定してください。
uint8_t　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。

6.7.5 jcu_image_info_t

概　要　デコードした画像の情報を取得するための構造体
ヘッダ　r_jcu_api.h
説　明
メンバー変数

uint32_t　width	画像の幅を格納します。
uint32_t　height	画像の高さを格納します。
jcu_jpeg_format_t encodedFormat	JPEG ファイルのピクセルフォーマットを格納します。

6.7.6 jcu_encode_param_t

概　要　エンコード用のオプションを設定するためのパラメータ

ヘッダ　　r_jcu_api.h
説　明
メンバー変数

jcu_jpeg_format_t 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。
int32_t QuantizationTable[]	圧縮時に使用する量子化テーブルを選択します
int32_t　HuffmanTable[]	圧縮時に使用するハフマンテーブルを選択します
uint32_t　DRI_value	圧縮データの DRI(Define Restart Interval)値を選択します
uint32_t　width	入力画像の幅を選択します
uint32_t　height	入力画像の高さを選択します
jcu_cbcr_offset_t inputCbCrOffset	入力画像データの Cb/Cr 範囲設定を行います

6.7.7　jcu_internal_information_t

概　要　　ドライバ内部の状態
ヘッダ　　r_jcu_api.h
説　明
メンバー変数

jcu_codec_status_t Codec	選択されているコーデック種別です。
bool_t　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。
jcu_int_detail_errors_t ErrorFilter	異常があったときにエラーにするかどうかのフィルター
jcu_async_status_t AsyncStatus	jcu_async_status_t 型構造体。参考 6.7.8。
r_ospl_caller_t InterruptCallbackCaller	割込みコールバック関数の管理する構造体
jcu_i_lock_t*　I_Lock	OSPL の I-ロックの管理構造体
r_ospl_i_lock_vtable_t* I_LockVTable	OSPL の I-ロックの V-Table
bool_t　Is_I_LockMaster	I-ロックのマスターかどうか
r_ospl_async_t* AsyncForFinalize	終了処理の非同期に関する構造体

6.7.8　jcu_async_status_t

概　要　　ドライバの非同期実行に関する状態
ヘッダ　　r_jcu_api.h
説　明
メンバー変数

jcu_status_information_t Status	メインの状態。
bit_flags_fast32_t SubStatusFlags	一時停止の要因の状態。 jcu_sub_status_t 型。
bool_t　IsPaused	一時停止中かどうか。
bool_t IsEnabledInterrupt	チャンネルに関わるすべての割込み線を許可しているかどうか。
r_ospl_flag32_t InterruptEnables	チャンネルに関わる割込み線のうち、割込み許可している線。
r_ospl_flag32_t InterruptFlags	割込み状態レジスタのコピー。 各ビットに対応するシンボルはドライバ内部で定義し、ドライバによっては、非公開です。

r_ospl_flag32_t CancelFlags	停止処理の開始から完了までの状態を扱うドライバ内部のフラグ。

6.7.9　jcu_context_t

JCU のコンテキスト。 JCU_MULTI_THREAD = 1 のときのみ定義されます。

JCU_MULTI_THREAD = 1 の設定では、内部で 1 つの制御スレッドが存在します。それぞれのスレッドが JCU の API を呼び出したとき、制御スレッドに通知を行い、制御スレッドだけが JCU のレジスタにアクセスします。

メンバー変数は内部で使用します。アクセスしないでください。

6.7.10　jcu_thread_t

JCU ドライバが管理するスレッドに関する情報。JCU_MULTI_THREAD = 1 のときのみ構造体は定義されます。

メンバー変数は内部で使用します。アクセスしないでください。

6.7.11　jcu_config_t

概　要　JCU ドライバを初期化・終了するときのパラメータ。

ヘッダ　r_jcu_api.h

説　明　JCU_MULTI_THREAD = 1 のときのみ構造体は定義されます。
通常、Flags = 0 に設定します。
割込みから JCU の API を呼び出せるようにするときは、スレッドから呼び出す R_JCU_Initialize・R_JCU_TerminateEx の引数に、Flags = F_JCU_OwnerThread, OwnerThread = R_OSPL_THREAD_NULL を指定します。

メンバー変数

bit_flags_fast32_t　Flags	フラグド構造体（必須）参考：(6.11.2)。 F_JCU_OwnerThread F_JCU_OwnerContext
r_ospl_thread_id_t OwnerThread	初期化時は、通常、省略します。 R_JCU_Initialize・R_JCU_Terminate 以外の API を呼び出すスレッド。 R_OSPL_THREAD_NULL = 割込みコンテキスト。 省略時は、カレント スレッド。
jcu_context_t* OwnerContext	初期化時は、通常、省略します。 使用するコンテキスト。 省略時は、すでに OwnerThread メンバー変数に割り当てられているコンテキストがあれば、それを使います。なければ、内部で生成します。

6.7.12　jcu_contexts_config_t

概　要　R_JCU_CONTEXTS_Initialize のパラメータ

ヘッダ　r_jcu_api.h

説　明　JCU_MULTI_THREAD = 1 のときのみ構造体は定義されます。

メンバー変数

bit_flags_fast32_t　Flags	フラグド構造体（必須）参考：(6.11.2)。 F_JCU_ThreadMemory F_JCU_ThreadMemoryArray
r_ospl_table_t* ThreadMemory	JCU ドライバが管理するスレッドに関する情報を格納する配列。初期化済みの配列番号表。 要素数は、JCU の API を呼び出すスレッドの最大数かそれより大きくしてください。 スレッドの数 ＝

	（R_JCU_Initialize を呼び出した回数）－ （R_JCU_Terminate を呼び出した回数）です。
jcu_thread_t* ThreadMemoryArray	jcu_thread_t 型(6.7.10) の配列の先頭アドレス。初期化は不要です。要素数は、ThreadMemory 引数に指定した配列番号表の要素数と同じか大きくしてください。

6.7.13　jcu_operator_config_t

概　要　制御スレッドのパラメータ。

ヘッダ　r_jcu_api.h

説　明　内部用です。

JCU_MULTI_THREAD = 1 のときのみ構造体は定義されます。

メンバー変数

bit_flags_fast32_t　Flags	フラグド構造体（必須）参考：(6.11.2)。 F_JCU_ReturnValue F_JCU_MultiThreadQueue F_JCU_ReadyOrFinishedEvent
errnum_t　ReturnValue	制御スレッドの関数の返り値。制御スレッドが設定します。
r_ospl_queue_id_t MultiThreadQueue	API を呼び出したスレッドから制御スレッドへ通知するキュー
r_ospl_async_t ReadyOrFinishedEvent	制御スレッドから API を呼び出したスレッドへの通知設定

6.8　変数一覧

表 6-31 に global 変数を示します。

表 6-31　global 変数

型	変数名	内容
jcu_internal_information_t	gs_jcu_internal_information	ドライバ内部の状態
jcu_operator_config_t	gs_OperatorConfig	制御スレッドの状態

6.9　関数一覧

表 6-32 に API 関数一覧を示します。

表 6-32 API 関数一覧

章	関数名	概要
6.10.1	R_JCU_Initialize	JCU ドライバを初期化します。
6.10.2	R_JCU_Terminate	JCU ドライバを終了します。（同期処理）
6.10.3	R_JCU_TerminateAsync	JCU ドライバを終了します。（非同期処理）
6.10.4	R_JCU_TerminateEx	JCU ドライバを終了します。（パラメータ付き）
6.10.5	R_JCU_SelectCodec	圧縮または伸長を選択します。
6.10.6	R_JCU_SetCountMode	カウントモード(分割処理)を選択します。
6.10.7	R_JCU_SetPauseForImageInfo	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。
6.10.8	R_JCU_SetErrorFilter	有効とするデコードエラー検出の種類(jcu_int_detail_error_t)を設定する。
6.10.9	R_JCU_Start	JCU の動作を開始します。（同期処理）
6.10.10	R_JCU_StartAsync	JCU の動作を開始します。（非同期処理）
6.10.11	R_JCU_Continue	一時停止した JCU の動作を再開します。（同期処理）
6.10.12	R_JCU_ContinueAsync	一時停止した JCU の動作を再開します。（非同期処理）
6.10.13	R_JCU_SetDecodeParam	デコード用のパラメータを設定します。
6.10.14	R_JCU_GetImageInfo	JPEG ファイルの画像情報を取得します。
6.10.15	R_JCU_SetEncodeParam	エンコード用のパラメータを設定します。
6.10.16	R_JCU_SetQuantizationTable	量子化テーブルを設定します。
6.10.17	R_JCU_SetHuffmanTable	ハフマンテーブルを設定します。
6.10.18	R_JCU_Set2ndCacheAttribute	L2 キャッシュの属性を設定します。
6.10.19	R_JCU_GetEncodedSize	圧縮したデータのサイズを取得します。
6.10.20	R_JCU_GetAsyncStatus	割込みや非同期処理の状況を示す構造体へのポインタを取得します。
6.10.21	R_JCU_OnInterrupting	割込みを受信します。
6.10.22	R_JCU_OnInterrupted	割込み応答処理を行います。
6.10.23	R_JCU_CONTEXTS_Initialize	スレッドの管理を初期化します。
6.10.24	R_JCU_OPERATOR_Thread	制御スレッドが実行する関数。
6.10.25	R_JCU_OPERATOR_Dispatch	制御スレッドの一部を実行します。
6.10.26	R_JCU_OPERATOR_FinalizeStart	制御スレッドを終了するように要求します。
6.10.27	R_JCU_SetContextInInterrupt	使用するコンテキストを設定します。

表 6-33 ユーザ定義関数一覧

章	関数名	概要
6.10.28	R_JCU_OnInitialize	ユーザ定義部の初期化をします。
6.10.29	R_JCU_OnFinalize	ユーザ定義部の終了処理をします。
6.10.30	R_JCU_SetDefaultAsync	r_ospl_async_t 型の構造体のデフォルト値を設定します。

6.10.31	R_JCU_SetInterruptCallbackCaller	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。
6.10.32	R_JCU_OnEnableInterrupt	割込み許可の設定を行います。
6.10.33	R_JCU_OnDisableInterrupt	割込み禁止の設定を行います。
6.10.34	R_JCU_OnInterruptDefault	デフォルトの割込みコールバック関数。
6.10.35	R_JCU_CreateOperatorThread	制御スレッドを生成します。
6.10.36	R_JCU_DestroyOperatorThread	制御スレッドを削除します。
6.10.37	R_JCU_OnAllocateContext	コンテキストのメモリ領域を確保します。
6.10.38	R_JCU_OnFreeContext	コンテキストのメモリ領域を解放します。

本ドライバの状態遷移図と状態遷移表は、以下の通りです。

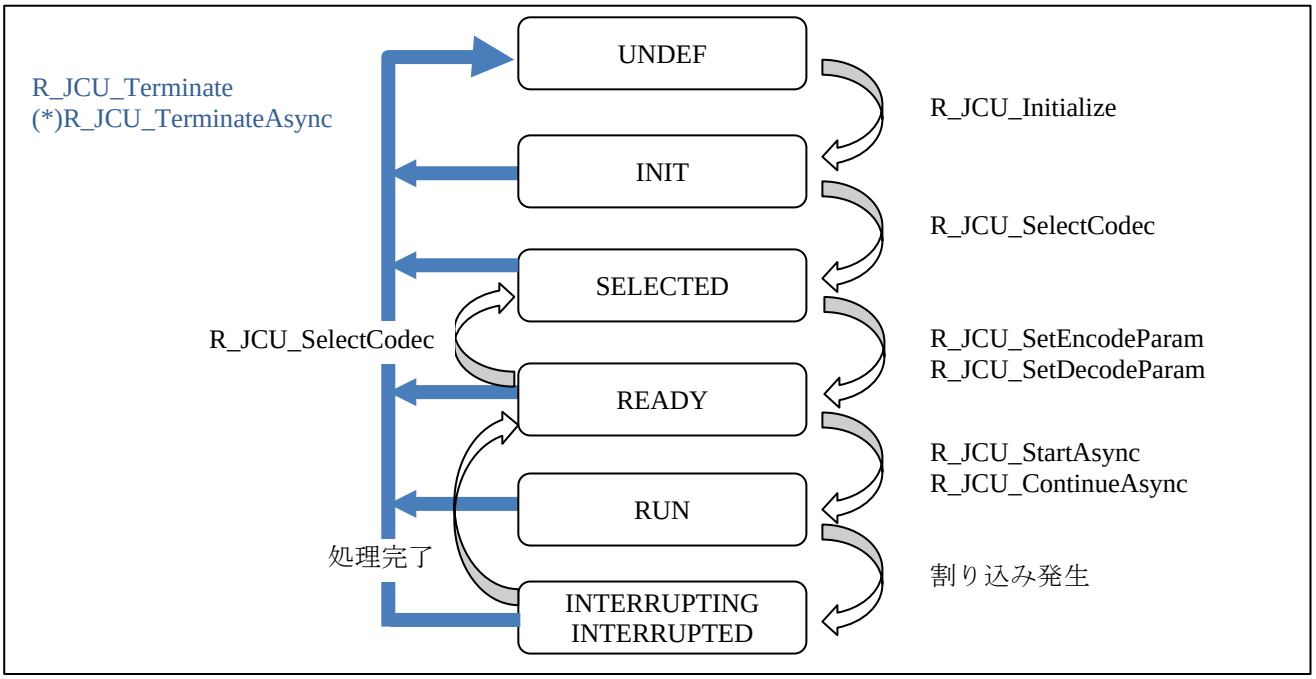

図 6.7　状態遷移図

* Run 状態で、R_JCU_TerminateAsync 実行した場合は、遷移しない。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。

表 6-34 状態遷移表

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*	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	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	NG	NG	変更なし
圧縮用 API							
R_JCU_SetEncodeParam	NG	NG	OK	OK	NG	NG	Ready
R_JCU_SetQuantization Table	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

* Run 中に実行した場合は、実行後の状態は「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。

6.10　関数仕様

サンプルコードの関数仕様を以下に示します。

6.10.1　R_JCU_Initialize

概　要	JCU ドライバを初期化します。
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_Initialize (jcu_config_t*　in_out_Config);
説　明	・　ドライバ内部の状態(gs_jcu_internal_information)を初期化します。 ・　ユーザ定義関数(R_JCU_OnInitialize)を実行します。 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

引　数	jcu_config_t*　in_out_Config	NULL またはパラメータ(6.7.11)。
リターン値	エラーコード。	

6.10.2　R_JCU_Terminate

概　要	JCU ドライバを終了します。（同期関数）
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_Terminate(void);
説　明	JCU ドライバを終了する処理です。終了処理が完了するまで、リターンしない同期関数です。 ・ドライバ内部の状態を変更します。 ・ユーザ定義関数(R_JCU_OnFinalize)を実行します。 ユーザ定義関数では、以下の処理を行ってください。 1. JCU モジュールへのクロック供給停止 2. JCU に関係する割込みの優先度の設定クリア 3. その他処理に必要な環境固有の設定 もし、jcu_context_t::StatusOfContext が JCU_STATUS_RUN の場合は、JCU が動作停止するまで待ちます。 割込みから JCU の API を呼び出したときは、R_JCU_Terminate の代わりに R_JCU_TerminateEx を呼び出してください。

引　数	なし	
リターン値	エラーコード。	

6.10.3　R_JCU_TerminateAsync

概　要	JCU ドライバを終了します。（非同期関数）
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_TerminateAsync(r_ospl_async_t* const　async);
説　明	JCU ドライバを終了する処理です。終了処理が完了する前に、すぐにリターンする非同期関数です。 async 引数の詳細は、RZ/A1H グループ OS 移植層 OSPL （R01AN1887JJ）にある R_DRIVER_TransferAsync 関数の説明を参照してください。 処理の内容は、R_JCU_Terminate 処理を参照してください。

引　数	r_ospl_async_t* async	通知設定
リターン値	エラーコード。	

6.10.4　R_JCU_TerminateEx

概　要	JCU ドライバを終了します。（パラメータ付き）
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_TerminateEx(jcu_config_t*　in_out_Config);
説　明	割込みから JCU の API を呼び出したときは、in_out_Config にパラメータを設定してください。

引　数	jcu_config_t*　in_out_Config	NULL またはパラメータ(6.7.11)。
リターン値	エラーコード。	

6.10.5　R_JCU_SelectCodec

概　要	圧縮または伸長を選択します。
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_SelectCodec(const jcu_codec_t codec);
説　明	・JCU の処理状態を選択（伸長または圧縮）します。 ・この関数を呼び出すと、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るので、全て再設定してください。

引　数	jcu_codec_t codec	コーデック種別。
リターン値	エラーコード。	

6.10.6　R_JCU_SetCountMode

概　要	カウントモード(分割処理)を選択します。
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_SetCountMode(const jcu_count_mode_param_t *const buffer);
説　明	・カウントモードの設定を行います。 ・入力バッファの分割処理と、出力バッファの分割処理の同時実行は出来ません。

引　数	jcu_count_mode_param_t * buffer	カウントモードの設定。
リターン値	エラーコード。	

6.10.7　R_JCU_SetPauseForImageInfo

概　要	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。
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_SetPauseForImageInfo(const bool_t is_pause);
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、引数で指定します。

引　数	bool_t is_pause	TRUE:一時停止する FALSE:一時停止しない
リターン値	エラーコード。	

6.10.8　R_JCU_SetErrorFilter

概　要	有効とするデコードエラー検出の種類(jcu_int_detail_error_t)を設定する。
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_SetErrorFilter(jcu_int_detail_error_t filter);
説　明	・有効とするデコードエラー詳細情報を設定します。

・有効としたデコードエラーが発生した場合は、割込みが発生します。

引　数	jcu_int_detail_error_t filter	有効とするエラー検出の種類(jcu_int_detail_error_t)のビット・フラグ値。
リターン値	エラーコード。	

6.10.9　R_JCU_Start

概　要	JCU の動作を開始します。（同期関数）
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_Start(void);
説　明	・JCU をスタートします。デコードまたはエンコード処理が、完了または中断するまで、リターンしない同期関数です。 スタートする前に、R_JCU_SetDecoderParam または R_JCU_SetEncoderParam の API 関数で、パラメータを設定して下さい。 JCU をスタートした後は、キャンセルできません。

引　数	なし	
リターン値	エラーコード。	

6.10.10　R_JCU_StartAsync

概　要	JCU の動作を開始します。（非同期関数）
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_StartAsync(r_ospl_async_t* const　async);
説　明	・JCU をスタートします。デコードまたはエンコード処理が、完了または中断する前に、すぐにリターンする非同期関数です。 async 引数の詳細は、RZ/A1H グループ OS 移植層 OSPL （R01AN1887JJ）にある R_DRIVER_TransferAsync 関数の説明を参照してください。 処理の内容は、R_JCU_Start 処理を参照してください。

引　数	r_ospl_async_t* async	通知設定
リターン値	エラーコード。	

6.10.11　R_JCU_Continue

概　要	一時停止した JCU の動作を再開します。（同期関数）
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_Continue(const jcu_continue_type_t type);
説　明	・中断した JCU の動作（モード）を再開させます。再開した動作が、完了または中断するまで、リターンしない同期関数です。 再開させる動作は、引数で指定します。

引　数	jcu_continue_type_t type	JCU が再開するモード。
リターン値	エラーコード。	

6.10.12　R_JCU_ContinueAsync

概　要	一時停止した JCU の動作を再開します。（非同期関数）
ヘッダ	r_jcu_api.h
宣　言	jcu_errorcode_t R_JCU_ContinueAsync(const jcu_continue_type_t type, r_ospl_async_t* const　async);

説　明　・中断した JCU の動作（モード）を再開させます。再開した動作が、完了または中断する前に、すぐにリターンする非同期関数です。再開させる動作は、引数で指定します。

引　数		
	jcu_continue_type_t type	JCU が再開するモード。
	r_ospl_async_t* const async	通知設定。NULL 不可。
リターン値	エラーコード。	

6.10.13　R_JCU_SetDecodeParam

概　要　デコード用のパラメータを設定します。

ヘッダ　r_jcu_api.h

宣　言　jcu_errorcode_t R_JCU_SetDecodeParam(const jcu_decode_param_t *const decode, const jcu_buffer_param_t *const buffer);

説　明　・デコード用のパラメータを設定します。
　ピクセルフォーマットが ARGB8888 以外のときは、decode.alpha には 0 を設定してください。
　ピクセルフォーマットが YCbCr 以外のときは、decode.outputCbCrOffset には JCU_CBCR_OFFSET_0 を設定してください。

引　数		
	jcu_decode_param_t *decode	デコード用のパラメータ。
	jcu_buffer_param_t *buffer	入出力バッファ。
リターン値	エラーコード。	

6.10.14　R_JCU_GetImageInfo

概　要　JPEG ファイルの画像情報を取得します。

ヘッダ　r_jcu_api.h

宣　言　jcu_errorcode_t R_JCU_GetImageInfo(jcu_image_info_t *const buffer);

説　明　・JPEG ファイルの画像情報（幅、高さ、エンコードフォーマット）を取得します。
・画像情報取得の割込み発生前は、不定値となります。
・取得した JPEG ファイル画像情報のピクセルフォーマットが、jcu_jpeg_format_t の範囲外であった場合は、エラーと判定して、デコードは行わないでください。

引　数		
	jcu_image_info_t * buffer	画像情報格納用メモリのポインタ。
リターン値	エラーコード。	

6.10.15　R_JCU_SetEncodeParam

概　要　エンコード用のパラメータを設定します。

ヘッダ　r_jcu_api.h

宣　言　jcu_errorcode_t R_JCU_SetEncodeParam(const jcu_encode_param_t *const encode, const jcu_buffer_param_t *const buffer);

説　明　・エンコード用のパラメータを設定します。

引　数		
	jcu_encode_param_t *encode	エンード用のパラメータ。
	jcu_buffer_param_t *buffer	入出力バッファ。
リターン値	エラーコード。	

6.10.16　R_JCU_SetQuantizationTable

概　要　　量子化テーブルを設定します。
ヘッダ　　r_jcu_api.h
宣　言　　jcu_errorcode_t R_JCU_SetQuantizationTable(const jcu_table_no_t tableNo, const uint8_t *const table);
説　明　　・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、「RZ/A1H グループ ユーザーズマニュアル ハードウェア編」の、45.3.1 (4)章を参照してください。
　付属の QuantizationTable_Generator.html ファイルで量子化テーブルの設定値のサンプルを計算することができます。

引　数		
	jcu_table_no_t tableNo	データをセットするテーブル番号。
	uint8_t * table	設定する量子化テーブル。
リターン値	エラーコード。	

6.10.17　R_JCU_SetHuffmanTable

概　要　　ハフマンテーブルを設定します。
ヘッダ　　r_jcu_api.h
宣　言　　jcu_errorcode_t R_JCU_SetHuffmanTable(const jcu_table_no_t tableNo, const jcu_huff_t type, const uint8_t *const table);
説　明　　・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、「RZ/A1H グループ ユーザーズマニュアル ハードウェア編」の、45.3.1 (4)章を参照してください。

引　数		
	jcu_table_no_t tableNo	データをセットするハフマンテーブル番号。
	jcu_huff_t type	ハフマンテーブルに設定する成分
	uint8_t * table	設定するハフマンテーブル
リターン値	エラーコード。	

6.10.18　R_JCU_Set2ndCacheAttribute

概　要　　JCU が L2 キャッシュにアクセスするときの属性を設定します。
ヘッダ　　r_jcu_api.h
宣　言　　jcu_errorcode_t　R_JCU_Set2ndCacheAttribute(r_ospl_axi_cache_attribute_t const read_cache_attribute,　r_ospl_axi_cache_attribute_t const　write_cache_attribute);
説　明
引　数

	r_ospl_axi_cache_attribute_t read_cache_attribute	ARCACHE[3:0]信号 参考：r_ospl_axi_cache_attribute_t
	r_ospl_axi_cache_attribute_t write_cache_attribute	AWCACHE[3:0]信号 参考：r_ospl_axi_cache_attribute_t
リターン値	エラーコード。エラーなし＝0	

6.10.19　R_JCU_GetEncodedSize

概　要　　エンコードしたデータのサイズを取得します。
ヘッダ　　r_jcu_api.h
宣　言　　jcu_errorcode_t R_JCU_GetEncodedSize(size_t *const out_Size);
説　明　　・エンコードして圧縮したデータのサイズを取得します。
　・エンコード完了の割込み発生前は、不定値となります。

引　数		
	size_t * out_Size	サイズ情報格納用メモリのポインタ。

リターン値	エラーコード。

6.10.20　R_JCU_GetAsyncStatus

概　要　割込みや非同期処理の状況を示す構造体へのポインタを取得します。
ヘッダ　r_jcu_api.h
宣　言　R_JCU_GetAsyncStatus(const jcu_async_status_t** const　out_Status);
説　明　・out_Status 引数に指定するポインタ変数には、const 修飾子が必要です。

引　数		
	jcu_async_status_t** out_Status	（出力） 割込みや非同期処理の状況を示す構造体へのポインタ
リターン値	エラーコード。エラーなし＝0	

6.10.21　R_JCU_OnInterrupting

概　要　割込みを受信します。
ヘッダ　r_jcu_api.h
宣　言　errnum_t R_JCU_OnInterrupting(const r_ospl_interrupt_t* const　InterruptSource);
説　明　通常、デフォルトの割込みコールバック関数から本関数が自動的に呼び出されます。本関数は、割込みステータス・レジスタから、jcu_async_status_t::InterruptFlags 変数に割込み通知を伝達し、割込みをクリアします。
詳細は、RZ/A1H グループ PFV,JCU 向け OS 移植層 OSPL （R01AN1887JJ）にある R_DRIVER_OnInterrupting 関数の説明を参照

引　数		
	r_ospl_interrupt_t* InterruptSource	割込み発信元
リターン値	エラーコード。エラーなし＝0	

6.10.22　R_JCU_OnInterrupted

概　要　割込み応答処理を行います。
ヘッダ　r_jcu_api.h
宣　言　errnum_t　R_JCU_OnInterrupted(void);
説　明　通常、デフォルトの割込みコールバック関数から本関数が自動的に呼び出されます。R_JCU_OnInterrupting 関数によって 1 に設定された jcu_async_status_t::InterruptFlags 変数のビットを 0 にクリアして、割込み応答処理を行います。
詳細は、RZ/A1H グループ PFV,JCU 向け OS 移植層 OSPL （R01AN1887JJ）にある R_DRIVER_OnInterrupted 関数の説明を参照。

引　数		
	なし	
リターン値	エラーコード。エラーなし＝0	

6.10.23　R_JCU_CONTEXTS_Initialize

概　要　スレッドの管理を初期化します。
ヘッダ　r_jcu_api.h
宣　言　errnum_t　R_JCU_CONTEXTS_Initialize(jcu_contexts_config_t*　in_out_Config);
説　明　スレッドとコンテキストの関係の管理を初期化します。

引　数		
	jcu_contexts_config_t* in_out_Config	パラメータ(6.7.12)
リターン値	エラーコード。エラーなし＝0	

6.10.24　R_JCU_OPERATOR_Thread

概　要　制御スレッドが実行する関数。
ヘッダ　r_jcu_api.h
宣　言　void　R_JCU_OPERATOR_Thread(void* in_Argument);
説　明
引　数

void* in_Argument	パラメータ(6.7.13) jcu_operator_config_t* 型から void* 型へキャストした値を指定してください。

リターン値　なし

6.10.25　R_JCU_OPERATOR_Dispatch

概　要　制御スレッドの一部を実行します。
ヘッダ　r_jcu_api.h
宣　言　errnum_t　R_JCU_OPERATOR_Dispatch(r_ospl_thread_id_t　in_ThreadID);
説　明　R_OSPL_IS_PREEMPTION=0 に定義されているときのみ、本関数は存在します。
制御スレッド以外のスレッドの中で、制御スレッドの処理を待っているときに呼び出します。
引　数

r_ospl_thread_id_t in_ThreadID	制御スレッドのスレッド ID

リターン値　エラーコード。エラーなし＝0

6.10.26　R_JCU_OPERATOR_FinalizeStart

概　要　制御スレッドを終了するように要求します。
ヘッダ　r_jcu_api.h
宣　言　errnum_t　R_JCU_OPERATOR_FinalizeStart(r_ospl_async_t* ref_Async);
説　明
引　数

r_ospl_async_t* ref_Async	制御スレッドが終了したときの通知設定

リターン値　エラーコード。エラーなし＝0

6.10.27　R_JCU_SetContextInInterrupt

概　要　割込みから API を呼び出すときに使用するコンテキストを設定します。
ヘッダ　r_jcu_pl.h
宣　言　jcu_context_t*　R_JCU_SetContextInInterrupt(jcu_context_t* in_NewCurrentContext);
説　明　本関数は、割込み関数から呼び出してください。 スレッドから本関数を呼び出さないでください。
本関数を呼び出してから、ドライバの API を呼び出してください。
割込み関数の最後でも、本関数を呼び出してください。その引数には、割込み関数の最初で設定されていたコンテキストを指定してください。このようにコンテキストを戻す理由は、多重割込みに対応するためです。
参照：(6.11.1)
引　数

jcu_context_t* in_NewCurrentContext	設定するコンテキスト

リターン値　これまで設定されていたコンテキスト

6.10.28　R_JCU_OnInitialize

概　要　　ユーザ定義部の初期化をします。
ヘッダ　　r_jcu_pl.h
宣　言　　errnum_t R_JCU_OnInitialize (void);
説　明　　JCU ドライバの初期化処理で実行されるユーザ定義関数です。
必要ならば、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

引　数	なし	
リターン値	エラーコード。	

6.10.29　R_JCU_OnFinalize

概　要　　ユーザ定義部の終了処理をします。
ヘッダ　　r_jcu_pl.h
宣　言　　errnum_t R_JCU_OnFinalize (errnum_t e);
説　明　　JCU ドライバの終了処理で実行されるユーザ定義関数です。
必要ならば、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の停止
- JCU に関係する割込みの優先度のクリア
- その他処理に必要な環境固有の設定

引　数	errnum_t e	エラーコード。そのまま戻り値に使用してください。
リターン値	エラーコード。	

6.10.30　R_JCU_SetDefaultAsync

概　要　　r_ospl_async_t 型の構造体のデフォルト値を設定します。
ヘッダ　　r_jcu_pl.h
宣　言　　void R_JCU_SetDefaultAsync(r_ospl_async_t* const　ref_Async,
r_ospl_async_type_t in_AsyncType);
説　明　　JCU ドライバの各非同期処理で実行されるユーザ定義関数です。
以下の処理を行ってください。
- r_ospl_async_t 型の構造体の Flags メンバー変数の値のうち、0 になっているビットに対応する、それぞれのメンバー変数をデフォルト値に設定

ReturnValue メンバー変数は、それぞれの呼び出し元の非同期処理関数で初期化されます。

引　数	r_ospl_async_t* ref_Async	通知設定。NULL 不可。
	r_ospl_async_type_t in_AsyncType	同期処理の種類。
リターン値	なし	

6.10.31　R_JCU_SetInterruptCallbackCaller

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。
ヘッダ　　r_jcu_pl.h
宣　言　　errnum_t　R_JCU_SetInterruptCallbackCaller(const r_ospl_caller_t* const Caller);

説　明　　JCU ドライバの各非同期処理で実行されるユーザ定義関数です。
以下の処理を行ってください。
- 割込みハンドラから、Caller 引数の値を指定した R_OSPL_CallInterruptCallback 関数を呼び出すような設定を行う。
詳細は、RZ/A1H グループ PFV,JCU 向け OS 移植層 OSPL （R01AN1887JJ）にある R_DRIVER_OnInterrupted 関数の説明を参照してください。
なお、Caller 引数が指す構造体の本体は、ドライバ呼び出し元が管理します。本ユーザ関数の内部では、チャンネル番号のチェックは不要です。

引　数	r_ospl_caller_t* Caller	R_OSPL_CallInterruptCallback 関数に渡す値
リターン値	エラーコード。エラーなし＝0	

6.10.32　R_JCU_OnEnableInterrupt

概　要　　割込み許可の設定を行います。
ヘッダ　　r_jcu_pl.h
宣　言　　void　R_JCU_OnEnableInterrupt(jcu_interrupt_lines_t const　Enables);
説　明　　JCU ドライバの割込み許可要求処理から実行されるユーザ定義関数です。
以下の処理を行ってください。
- JCU の割込みを許可するように、割込み関連を設定

引　数	jcu_interrupt_lines_t Enables	許可する割込み線の種類を 1 にした、ビット・フラグ値
リターン値	なし	

6.10.33　R_JCU_OnDisableInterrupt

概　要　　割込み禁止の設定を行います。
ヘッダ　　r_jcu_pl.h
宣　言　　void　R_JCU_OnDisableInterrupt(jcu_interrupt_lines_t const　Disables);
説　明　　JCU ドライバの割込み禁止要求処理から実行されるユーザ定義関数です。
以下の処理を行ってください。
- JCU の割込みを禁止するように、割込み関連を設定

引　数	jcu_interrupt_lines_t Disables	禁止する割込み線の種類を 1 にした、ビット・フラグ値
リターン値	なし	

6.10.34　R_JCU_OnInterruptDefault

概　要　　デフォルトの割込みコールバック関数。
ヘッダ　　r_jcu_pl.h
宣　言　　errnum_t　R_JCU_OnInterruptDefault(const r_ospl_interrupt_t* const InterruptSource,　const r_ospl_caller_t* const　Caller);
説　明　　割込み応答処理を行うタイミングで実行されるユーザ定義関数です。
R_JCU_StartAsync 関数などの Async 引数の .InterruptCallback メンバー変数を指定しなかったときに、本関数がコールバック関数として登録されます。

引　数	r_ospl_interrupt_t* InterruptSource	割込み発信元
	r_ospl_caller_t* Caller	R_OSPL_CallInterruptCallback 関数に渡された値
リターン値	エラーコード。エラーなし＝0	

6.10.35　R_JCU_CreateOperatorThread

概　要	制御スレッドを生成します。
ヘッダ	r_jcu_local.h
宣　言	errnum_t　R_JCU_CreateOperatorThread();
説　明	JCU_MULTI_THREAD=1 のときに定義が必要です。 スレッドを生成する前に、R_JCU_CONTEXTS_Initialize 関数を呼び出してください。生成したスレッドは、R_JCU_OPERATOR_Thread 関数を実行してください。

引　数	なし	
リターン値	エラーコード。エラーなし＝0	

6.10.36　R_JCU_DestroyOperatorThread

概　要	制御スレッドを削除します。
ヘッダ	r_jcu_local.h
宣　言	errnum_t　R_JCU_DestroyOperatorThread();
説　明	JCU_MULTI_THREAD=1 のときに定義が必要です。 R_JCU_OPERATOR_FinalizeStart を呼び出して制御スレッドの終了を要求し、終了するまで待ちます。

引　数	なし	
リターン値	エラーコード。エラーなし＝0	

6.10.37　R_JCU_OnAllocateContext

概　要	コンテキストのメモリ領域を確保します。
ヘッダ	r_jcu_local.h
宣　言	errnum_t　R_JCU_OnAllocateContext(jcu_context_t**　out_AddressOfContext);
説　明	JCU_MULTI_THREAD=1 のときに定義が必要です。

引　数	jcu_context_t** out_AddressOfContext	出力：確保したアドレス
リターン値	エラーコード。エラーなし＝0	

6.10.38　R_JCU_OnFreeContext

概　要	コンテキストのメモリ領域を解放します。
ヘッダ	r_jcu_local.h
宣　言	errnum_t　R_JCU_OnFreeContext(jcu_context_t**　in_out_AddressOfContext);
説　明	JCU_MULTI_THREAD=1 のときに定義が必要です。

引　数	jcu_context_t** in_out_AddressOfContext	入力：開放するアドレス 出力：NULL
リターン値	エラーコード。エラーなし＝0	

6.11　補足

6.11.1　マルチスレッド対応があるときに、割込みから API を呼び出すときの注意点

JCU_MULTI_THREAD=1 のとき、他の割込みから JCU の API を呼び出すときは、次のコードが必要です。

- 割込みが発生する前にスレッドから R_JCU_Initialize 関数を呼び出し、その引数に割込みから呼び出す設定をすること
- 割込みから JCU の API を呼び出す前に、R_JCU_SetContextInInterrupt を呼び出して割込み用コンテキストに設定すること
- 割込みから戻る前にコンテキストの設定を戻すこと
- JCU ドライバを終了するときは、スレッドから R_JCU_TerminateEx 関数を呼び出し、その引数に割込みから呼び出した設定をすること

```
void  main()
{
   jcu_config_t  jcu_config_for_interrupt;
   jcu_config_for_interrupt.Flags = F_JCU_OwnerThread;
   jcu_config_for_interrupt.OwnerThread = R_OSPL_THREAD_NULL; /* Interrupt */

   jcu_error = R_JCU_Initialize( &jcu_config_for_interrupt );
         IF( jcu_error != JCU_ERROR_OK ) { e=E_OTHERS; goto fin; }
   thread_config->JCU_Config = &jcu_config_for_interrupt;

   /* メイン処理 */

fin:
   jcu_error = R_JCU_TerminateEx( &jcu_config_for_interrupt );
}

void  OnOtherInterrupt()
{
   jcu_context_t*  this_level_context =
      thread_config->JCU_Config->OwnerContext;
   jcu_context_t*  low_level_context =
      R_JCU_SetContextInInterrupt( this_level_context );

   /* JCU の API を呼び出す */

fin:
   R_JCU_SetContextInInterrupt( low_level_context );
}
```

6.11.2　フラグド構造体パラメータ

構造体の中の Flags メンバー変数をビットフィールドとして使い、ビットが 1 であれば、対応するメンバー変数を有効にするというコーディングパターンです。ビットが 0 ならば、メンバー変数の値は、省略されたものとして、デフォルトの値が設定されるか設定を変更しません。バージョンアップしたら構造体のメンバーが増えた場合でも、バイナリ互換にできます。

```
FuncA_ConfigClass  config;

config.Flags  = F_FuncA_Param1 | F_FuncA_Param2;
config.Param1 = 10;
config.Param2 = 2;
FuncA( &config );
```

Flags |= F_FuncA_Param3 が無いため、config.Param3 はデフォルト値。

7. サンプルコード

サンプルコードは、ルネサス エレクトロニクスホームページから入手してください。

8. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RZ/A1H グループ ユーザーズマニュアル ハードウェア編
（最新版をルネサス エレクトロニクスホームページから入手してください。）

R7S72100 CPU ボード RTK772100BC00000BR（GENMAI）ユーザーズマニュアル
（最新版をルネサス エレクトロニクスホームページから入手してください。）

R7S72100 CPU ボード用オプションボード RTK7721000B00000BR（GENMAI）ユーザーズマニュアル
（最新版をルネサス エレクトロニクスホームページから入手してください。）

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
（最新版を ARM ホームページから入手してください。）

ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0
（最新版を ARM ホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

ARM ソフトウェア開発ツール（ARM Compiler toolchain、ARM DS-5 等）に関しては、ARM ホームページから入手してください。
（最新版を ARM ホームページから入手してください。）

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/contact/

改訂記録

Rev.	発行日	改訂内容
2.01	2018.02.02	マルチスレッド対応 エンコードするカウントモード（分割エンコード）の中断に対応 サンプルアプリケーションをキャッシュ領域のフレームバッファに対応 OSPL のバージョンを 1.60 に更新 以下は、コードの変更内容 ● 修正：分割デコードの最後の分割で２回割込みが入ったときに停止する不具合に対処 ● 修正：RTX 版で GIC_EndInterrupt を呼び出さないように修正 ● 修正：デコード完了時の判定方法の修正（判定を INS6 から DBTF|CBTF に修正） ● 修正：リセット方法の修正（JCU のスタンバイリクエストに対応） ● 修正：IsPaused 変数の値の不具合の修正 ● 修正：各種状態チェックの不足または余計なエラーの修正
1.03	2016.02.29	初期設定例をバージョン 1.01 に更新。 OSPL バージョン 0.96 に更新。 複数の要因で一時停止するケースに対応。 追加：R_JCU_Set2ndCacheAttribute。 量子化テーブルの設定値のサンプルを計算するツールを追加。 JPEG 画像をデコードしてアニメーション表示するサンプルを追加。 ビデオ入力から JPEG にエンコードしてデコードするサンプルを追加。
1.00	2014.06.20	初版

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. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレス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、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

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生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、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またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 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、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング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）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、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（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合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、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他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。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. 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。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
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。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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直接的、間接的に支配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。

(Rev.4.0-1 2017.11)

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　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■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：https://www.renesas.com/contact/