RZ/A1H グループ
ピクセルフォーマットコンバータ（PFV）サンプルドライバ

R01AN1880JJ0103
Rev.1.03
2017.08.24

要旨

本アプリケーションノートでは、RZ/A1H のピクセルフォーマット コンバータ（PFV）機能を使用し画像のピクセルフォーマットを変換するサンプルドライバについて説明します。

ピクセルフォーマットコンバータ（PFV）サンプルドライバの特長を以下に示します。

- 画像データのピクセルフォーマット、RGB888（入力のみ）、ARGB8888（出力のみ）、RGB565、YCbCr422 の変換を行います。
- 画像データに対して、ブライト調整（オフセット調整）と 9 軸のゲイン調整を行うことのできるカラーマトリクスを使った変換を行います。
- 画像データを RAM から PFV に入力するダイレクト メモリアクセス コントローラ（DMAC）と PFV から出力した画像データを RAM に転送する DMAC を使って、連続して変換を行います。DMAC のチャネルは初期化関数に指定します。PFV から出力される割込み（IFEI、OFFI）は、DMAC に通知します。
- PFV サンプルドライバは、PFV、DMAC とも任意のチャンネル番号に変更できます。

対象デバイス

RZ/A1H グループ

RZ/A1M 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。

目次

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
ピクセルフォーマット コンバータ (PFV) Ch(0～1)	画像データの変換
ダイレクト メモリアクセス コントローラ（DMAC） Ch(0～15)	画像データの転送。PFV の入力側と出力側の 2 チャネル分
割込みコントローラ（INTC）	PFV および DMAC の割込み制御
シリアルコミュニケーション インターフェース（SCIF）（UART）	デバッグ情報表示

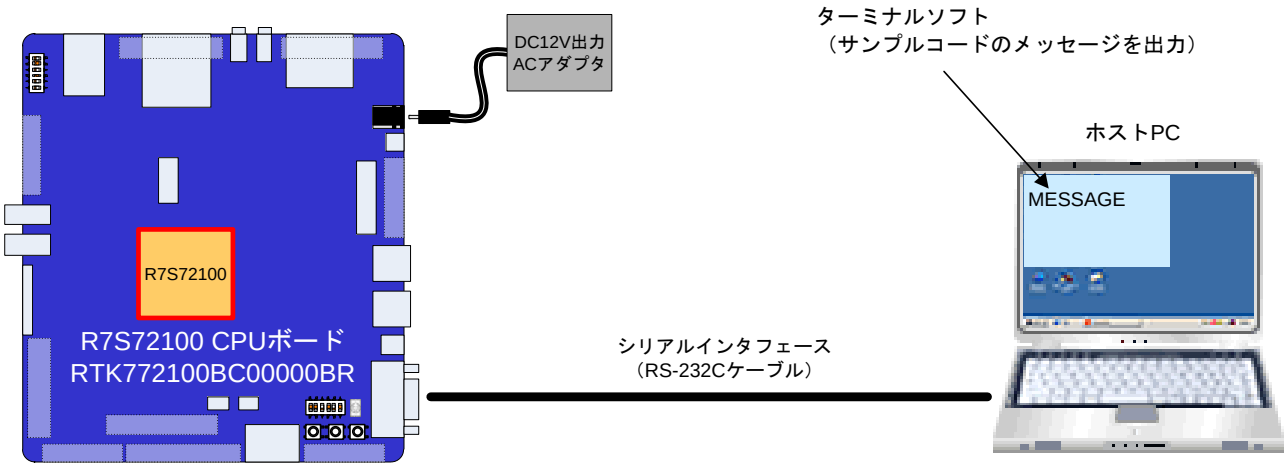


図 1.1 動作環境

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 1（P1φ）：66.67MHz 周辺クロック 0（P0φ）：33.33MHz
動作電圧		電源電圧（I/O）：3.3V 電源電圧（内部）：1.18V
統合開発環境		ARM®統合開発環境 ARM Development Studio 5（DS-5™）Version 5.16
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（サンプルのみ使用。PFV ドライバは非使用） ・シリアルインタフェース（Dsub-9 コネクタ）

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ーファイル iodefine.h（R01AN1860JJ）
- RZ/A1H グループ OS 移植層（OSPL）サンプルプログラム（R01AN1887JJ）
- RZ/A1H グループ ダイレクト メモリアクセス コントローラ（DMAC_RM）サンプルプログラム(PFV 付属)（R01AN1888JJ）

4. 周辺機能説明

PFV、DMAC 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 5-2 にブロック図を示します。

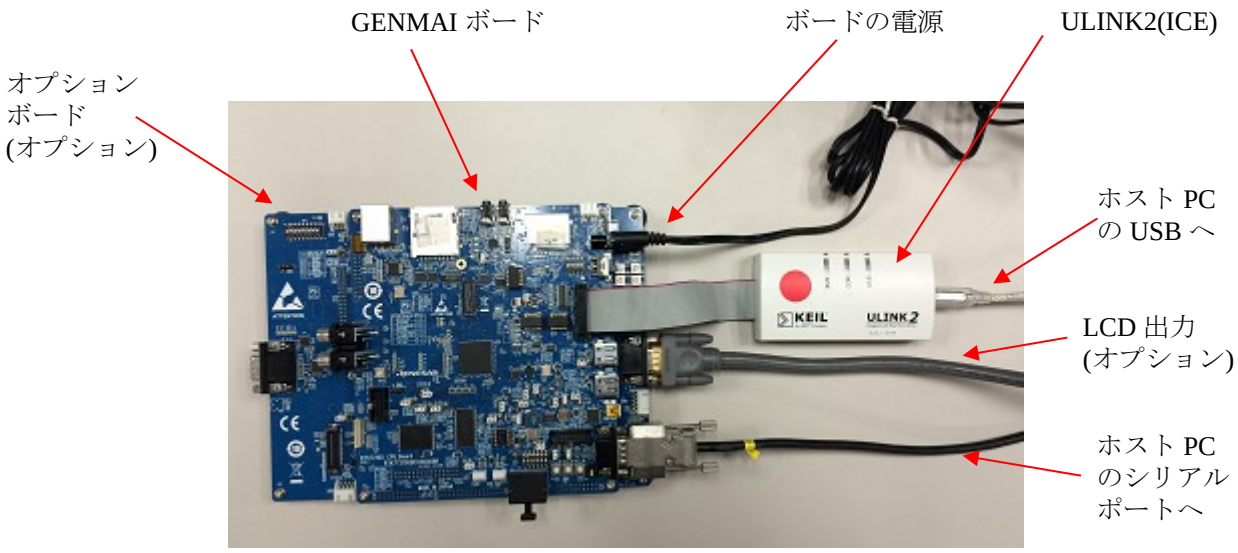
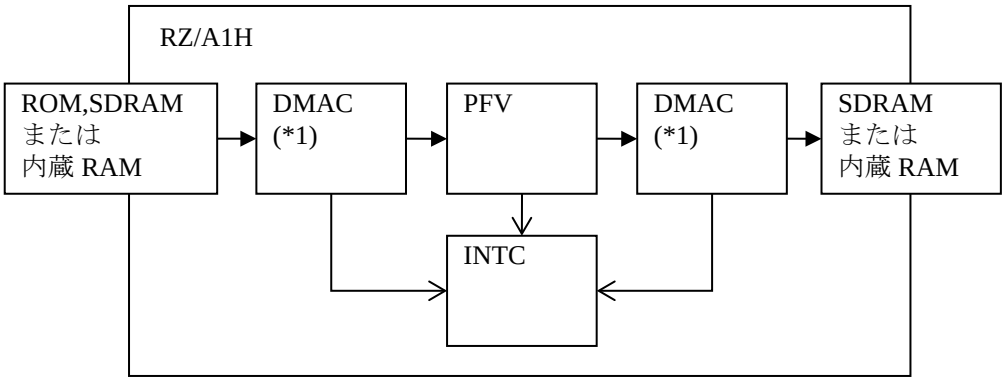


図 5.1　ハードウェア構成例



(*1) DMAC の代わりに CPU が PFV にアクセスすることもできます。

図 5-2　ブロック図

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

図 6-1 に DMAC を使うケースのシーケンスを、図 6-2 に DMAC を使わないケースのシーケンスを示します。

OS を変更する場合は、ドライバ移植層の内容の変更が必要です。

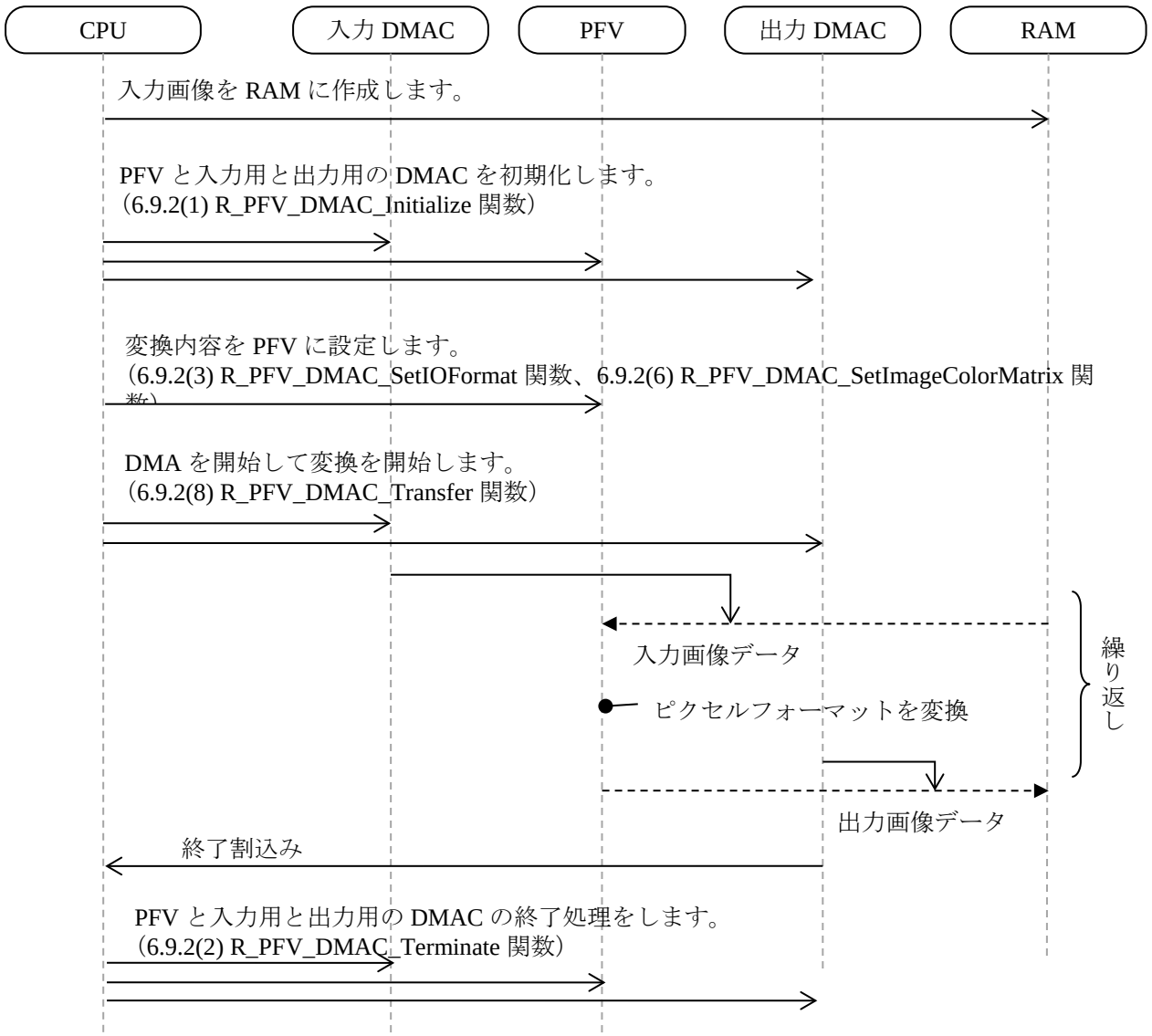


図 6-1　DMAC を使うケースのシーケンス

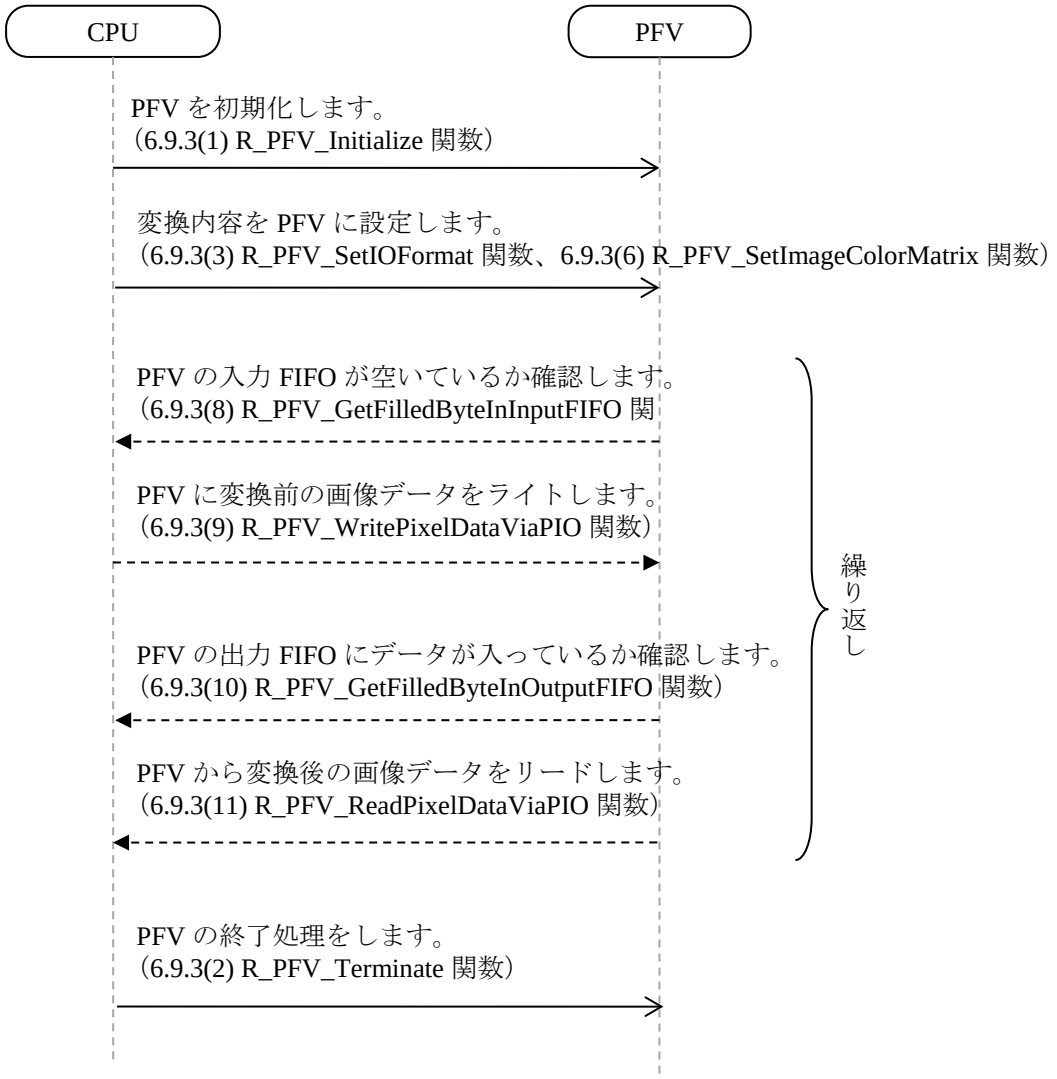


図 6-2　DMAC を使わないケースのシーケンス

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 6.3 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コマンド待ちになります。次章の説明にあるコマンドを入力することでサンプルプログラムの各動作を実行させることが可能となります。

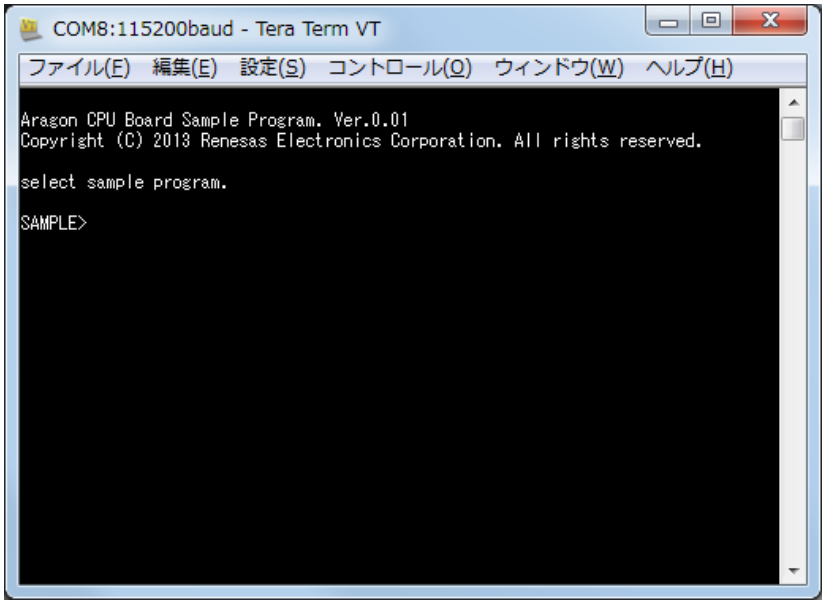


図 6.4 サンプルプログラムモジュール選択のコマンド待ち

6.2 コマンド一覧

サンプルプログラムで実装したコマンド一覧を示します。なおコマンドとパラメーターは大文字・小文字を区別しません。

表 6.1 コマンド一覧

コマンド	概要
Sample_PFV_PIO	CPU から直接 PFV の FIFO にピクセルデータを入出力して、メモリからメモリへピクセルデータの変換をします。
Sample_PFV_DMAC	DMAC から PFV の FIFO にピクセルデータを入出力して、メモリからメモリへピクセルデータの変換をします。
Sample_PFV_DMAC_Image	DMAC から PFV の FIFO にピクセルデータを入出力して、メモリからメモリへピクセルデータの変換をします。出力画像は LCD に表示されます。画像のゲインを変化させます。

6.3　メモリマップ

図 6.5 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。

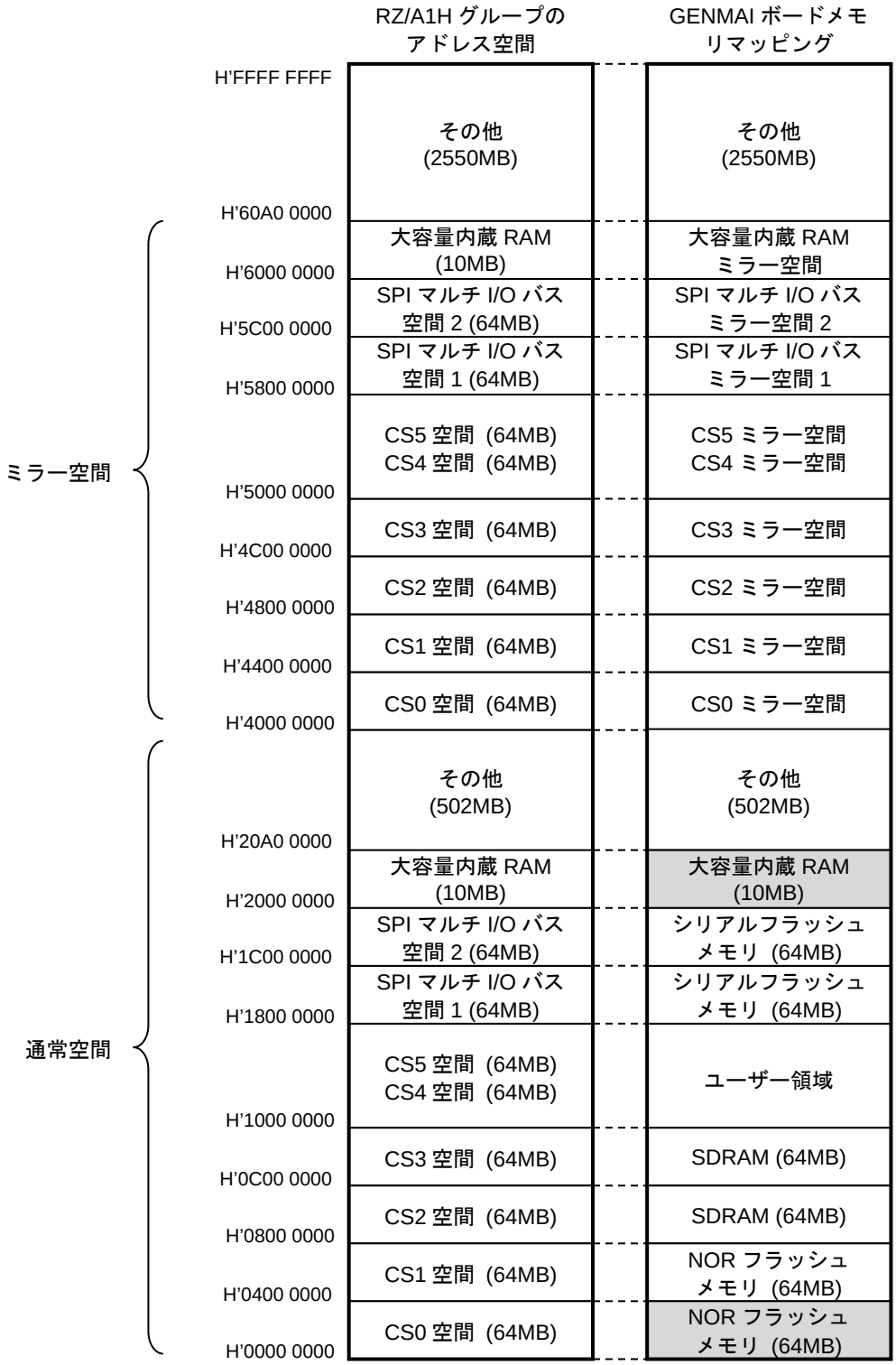


図 6.5　メモリマップ

6.3.1 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 6.2 に参考プログラムで使用するセクションを示し、図 6.6 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 6.2 使用するセクション

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	FALSH	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
CONST	デフォルトの定数データ領域	RO Data	FLASH	FLASH

	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			
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ー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ー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	ー	LRAM
IRQ_STACK	IRQ モードのスタック領域	ZI Data	ー	LRAM
FIQ_STACK	FIQ モードのスタック領域	ZI Data	ー	LRAM
SVC_STACK	スーパバイザ（SVC）モードのスタック領域	ZI Data	ー	LRAM
ABT_STACK	アボート（ABT）モードのスタック領域	ZI Data	ー	LRAM
TTB	MMU 変換テーブル領域	ZI Data	ー	LRAM
ARM_LIB_HEAP	アプリケーションヒープ領域	ZI Data	ー	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	ー	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。

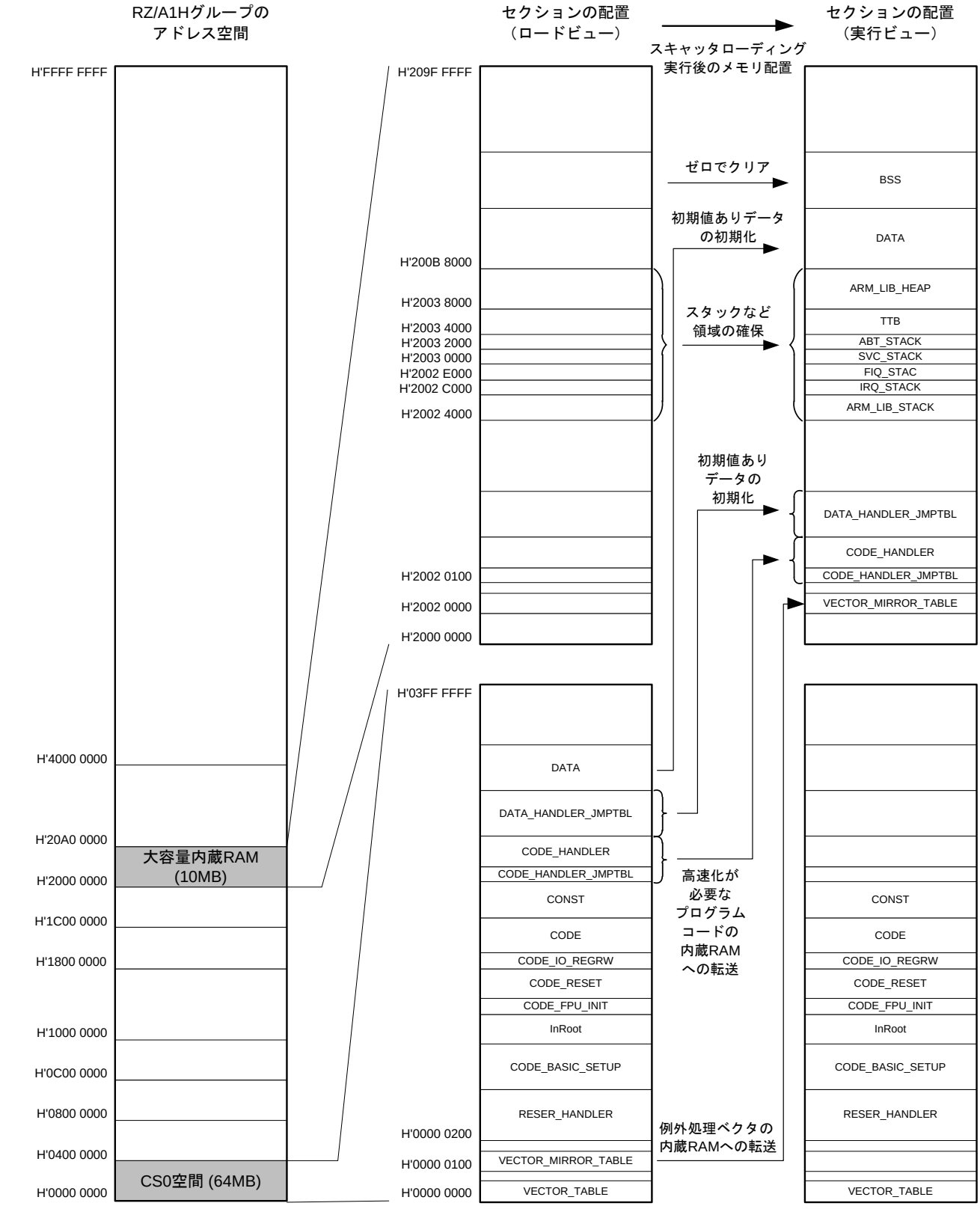

RZ/A1Hグループの
アドレス空間
セクションの配置
（ロードビュー）
スキャッタローディング
実行後のメモリ配置
セクションの配置
（実行ビュー）
H'FFFF FFFF
H'4000 0000
H'20A0 0000
大容量内蔵RAM
(10MB)
H'2000 0000
H'1C00 0000
H'1800 0000
H'1000 0000
H'0C00 0000
H'0800 0000
H'0400 0000
CS0空間 (64MB)
H'0000 0000
H'209F FFFF
H'200B 8000
H'2003 8000
H'2003 4000
H'2003 2000
H'2003 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 6.6　セクション配置

6.3.2 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）

6.3.3 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 6.7 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 6.7　例外処理ベクタテーブルの記述例

6.4 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
DMAINT0～ DMAINT#	INTERRUPT_LEVEL_OF_INPUT_DMAC(=20) INTERRUPT_LEVEL_OF_OUTPUT_DMAC(=19)	DMA 転送完了割込みの受信
IFEI0～IFEI#	INTERRUPT_LEVEL_OF_INPUT_EMPTY(=24)	PFV の入力 FIFO が空の割込みの受信。または、DMA 拡張リソースセレクタの対象。
OFFI0～OFFI#	INTERRUPT_LEVEL_OF_OUTPUT_FULL(=23)	PFV の出力 FIFO が満杯の割込みの受信。または、DMA 拡張リソースセレクタの対象。
PFVEI0～PFVEI#	INTERRUPT_LEVEL_OF_ERROR(=31)	PFV がエラーを発生したときの割込みの受信

6.5 基本型

表 6.5 にサンプルコードで使用する基本型を示します。

表 6.5 サンプルコードで使用する基本型

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（標準ライブラリにて定義）
int16_t	16 ビット整数、符号あり（標準ライブラリにて定義）
int32_t	32 ビット整数、符号あり（標準ライブラリにて定義）
int64_t	64 ビット整数、符号あり（標準ライブラリにて定義）
uint8_t	8 ビット整数、符号なし（標準ライブラリにて定義）
uint16_t	16 ビット整数、符号なし（標準ライブラリにて定義）
uint32_t	32 ビット整数、符号なし（標準ライブラリにて定義）
uint64_t	64 ビット整数、符号なし（標準ライブラリにて定義）
int_fast8_t	少なくとも 8 ビットある、高速な整数、符号あり
int_fast16_t	少なくとも 16 ビットある、高速な整数、符号あり
int_fast32_t	少なくとも 32 ビットある、高速な整数、符号あり
uint_fast8_t	少なくとも 8 ビットある、高速な整数、符号なし
uint_fast16_t	少なくとも 16 ビットある、高速な整数、符号なし
uint_fast32_t	少なくとも 32 ビットある、高速な整数、符号なし
uintptr_t	ポインターと同じビット数の符号なし整数型、または、物理アドレス
size_t	ポインターと同じビット数の符号なし整数型、または、バイトサイズ
ptrdiff_t	ポインターと同じビット数の符号あり整数型、アドレスの差分
bit_flags_fast32_t	uint_fast32_t 型の ビットフラグ型（ビットフィールド型）
bit_flags32_t	uint32_t 型の ビットフラグ型（ビットフィールド型）
float32_t	32 ビット浮動小数 （"__ARM_NEON__"を指定時、標準ライブラリにて定義）
float64_t	64 ビット浮動小数（標準ライブラリにて定義） （"__ARM_NEON__"を指定時、標準ライブラリにて定義）
float128_t	128 ビット浮動小数

6.6 定数/列挙体/エラーコード

表	型名	内容
6.6.1	-	バージョン
6.6.2	errnum_t	エラーコード
6.6.3	pfv_format_t	画像データのピクセルフォーマット
6.6.4	pfv_swap_t	画像データのオーダーの入れ替え
6.6.5	pfv_color_matrix_mode_t	画像データのピクセルフォーマットの変換前後
6.6.6	pfv_dc_offset_index_t	カラーマトリックスのオフセット成分の配列番号
6.6.7	pfv_matrix_multiply_index_t	カラーマトリックスのゲイン成分の配列番号
6.6.8	pfv_idtrg_t	入力 FIFO が Empty と判定されるときのバイト数
6.6.9	pfv_odtrg_t	出力 FIFO が Full と判定されるときのバイト数
6.6.10	pfv_interrupt_line_t	割込み線の種類
6.6.11	pfv_interrupt_lines_t	pfv_interrupt_line_t 型のビットフラグ
6.6.12	pfv_dmac_interrupt_unit_t	割込みが発生した周辺機能の種類
*1	r_ospl_interrupt_t	割込みの発信元に関する構造体
*1	r_ospl_caller_t	割込み処理に関するドライバ内部の情報
6.6.13	無名列挙体や定数	その他の定数

6.6.1 バージョン

定数名	設定値	内容
PFV_VERSION	101	PFV のバージョン番号
PFV_VERSION_STRING	"1.01"	PFV のバージョン番号の文字列

6.6.2 errnum_t 型 - エラーコード

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

*1 RZ/A1H グループ OS 移植層 OSPL （R01AN1887JJ）を参照。

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
E_PFV_HW_ERROR	0x4701(=18177)	PFV のハードウェアが検出したエラー

6.6.3 pfv_format_t

画像データのピクセルフォーマット。

定数名	値	内容
PFV_RGB888	0	32 ビットカラー。CPU のエンディアンで上位から、未使用が 8 ビット、赤が 8 ビット、緑が 8 ビット、青が 8 ビット。（入力のデフォルト）
PFV_ARGB8888	0	32 ビットカラー。CPU のエンディアンで上位から、アルファが 8 ビット、赤が 8 ビット、緑が 8 ビット、青が 8 ビット。ただし、アルファの出力値は、pfv_io_format_t::output_alpha 変数の固定値。（出力のデフォルト）
PFV_RGB565	1	16 ビットカラー。CPU のエンディアンで上位から、赤が 5 ビット、緑が 6 ビット、青が 5 ビット。
PFV_YCbCr422	3	2 ピクセルで 32 ビットのカラー。4 バイトのうち、アドレスの小さい方のバイトから、Cb（色差 U）、Y0（左のピクセルの輝度）、Cr（色差 V）、Y1（右のピクセルの輝度）。

6.6.4 pfv_swap_t

画像データのバイトオーダーの入れ替え。

定数名	値	内容
（ピクセルフォーマットが、PFV_RGB888 または PFV_ARGB8888 のとき）		
PFV_SWAP_ARGB8888	0	32 ビットカラー。CPU のエンディアンで上位から、アルファまたは未使用が 8 ビット、赤が 8 ビット、緑が 8 ビット、青が 8 ビット。（デフォルト）
PFV_SWAP_RABG8888	1	PFV_SWAP_ARGB8888 と同様に上位から、 赤、アルファまたは未使用、青、緑。
PFV_SWAP_GBAR8888	2	PFV_SWAP_ARGB8888 と同様に上位から、 緑、青、アルファまたは未使用、赤。
PFV_SWAP_BGRA8888	3	PFV_SWAP_ARGB8888 と同様に上位から、 青、緑、赤、アルファまたは未使用。
（ピクセルフォーマットが、PFV_RGB565 のとき）		
PFV_SWAP_RGB565_PIXEL10	2	16 ビットカラー×2 ピクセル。 上位 16 ビットが右側のピクセル、下位 16 ビットが左側のピクセル。CPU のエンディアンで上位から、赤が 5 ビット、緑が 6 ビット、青が 5 ビット。（デフォルト） FIFO に 16 ビットアクセスするときは、設定しないでください。
PFV_SWAP_RGB565_PIXEL01	0	同上、ただし、上位 16 ビットが左側のピクセル、下位 16 ビットが右側のピクセル。 FIFO に 16 ビットアクセスするときは、設定しないでください。

（ピクセルフォーマットが、PFV_YCbCr422 のとき）		
PFV_SWAP_YCbCr422_Cb_Y0_Cr_Y1	3	2 ピクセルで 32 ビットのカラー。4 バイトのうち、アドレスの小さい方のバイトから、Cb（色差 U）、Y0（左のピクセルの輝度）、Cr（色差 V）、Y1（右のピクセルの輝度）。（デフォルト）
PFV_SWAP_YCbCr422_Y0_Cb_Y1_Cr	2	PFV_SWAP_YCbCr422_Y1CrY0Cb と同様にアドレスの小さい方から、Y0、Cb(U)、Y1、Cr(V)。
PFV_SWAP_YCbCr422_Cr_Y1_Cb_Y0	1	PFV_SWAP_YCbCr422_Y1CrY0Cb と同様にアドレスの小さい方から、Cr(V)、Y1、Cb(U)、Y0。
PFV_SWAP_YCbCr422_Y1_Cr_Y0_Cb	0	PFV_SWAP_YCbCr422_Y1CrY0Cb と同様にアドレスの小さい方から、Y1、Cr(V)、Y0、Cb(U)。
PFV_SWAP_YCbCr422_Y1CrY0Cb	3	2 ピクセルで 32 ビットのカラー。CPU のエンディアンで上位から、Y1（右のピクセルの輝度）が 8 ビット、Cr（色差 U）が 8 ビット、Cb（色差 V）が 8 ビット。
PFV_SWAP_YCbCr422_CrY1CbY0	2	PFV_SWAP_YCbCr422_Y1CrY0Cb と同様に上位から、Cr、Y1、Cb、Y0。
PFV_SWAP_YCbCr422_Y0CbY1Cr	1	PFV_SWAP_YCbCr422_Y1CrY0Cb と同様に上位から、Y0、Cb、Y1、Cr。
PFV_SWAP_YCbCr422_CbY0CrY1	0	PFV_SWAP_YCbCr422_Y1CrY0Cb と同様に上位から、Cb、Y0、Cr、Y1。

参考：RZ/A1H グループ ユーザーズマニュアル ハードウェア編 - 47.3.2 入出力データ形式

6.6.5 pfv_color_matrix_mode_t

画像データのピクセルフォーマットの変換の種類。

参考：(6.9.4(1)) R_PFV_STATIC_GetColorMatrixMode 関数

定数名	値	内容
PFV_GBR_TO_GBR	0	RGB 系から RGB 系への変換。
PFV_GBR_TO_YCbCr	1	RGB 系から YCbCr 系への変換。
PFV_YCbCr_TO_GBR	2	YCbCr 系から RGB 系への変換。
PFV_YCbCr_TO_YCbCr	3	YCbCr 系から YCbCr 系への変換。
PFV_MATRIX_MODE_COUNT	4	pfv_color_matrix_mode_t 型の値の数

6.6.6 pfv_dc_offset_index_t

カラーマトリックスのオフセット成分の配列番号。

入力画像データの各成分が加算されます。ブライト調整に使用します。

YCbCr 系から変換するときは、Cb（色差 U）、Cr（色差 V）は常に-128 されます。

RGB565 のときは、8 ビット整数のうち上位 5 または 6 ビットのみ有効なので、下位ビットだけ変化する小さなオフセット調整を行っても、反映されません。

参考：(6.7.5)pfv_color_matrix_t 型のメンバー変数 dc_offset_plus_128

定数名	値	内容
PFV_DC_OFFSET_Y_OR_GREEN	0	Y（輝度）または緑のオフセットが入る配列番号
PFV_DC_OFFSET_BLUE	1	青のオフセットが入る配列番号
PFV_DC_OFFSET_RED	2	赤のオフセットが入る配列番号

6.6.7 pfv_matrix_multiply_index_t

カラーマトリックスのゲイン成分の配列番号。

オフセット成分を演算した後の各成分が積算されます。ゲイン調整に使用します。

参考：(6.7.5)pfv_color_matrix_t 型のメンバー変数 matrix_multiply_256

定数名	値	内容
PFV_COLOR_MATRIX_GG	0	RGB 系へ変換するとき： G1 = GG*G0 + GB*B0 + GR*R0 B1 = BG*G0 + BB*B0 + BR*R0 R1 = RG*G0 + RB*B0 + RR*R0 YCbCr 系へ変換するとき： Y1 = YY*G0 + GB*B0 + GR*R0 Cb1 = BG*G0 + BB*B0 + BR*R0 + 128 Cr1 = RG*G0 + RB*B0 + RR*R0 + 128 ただし、R0,G0,B0 は、オフセット成分の演算後の、赤または Cr（色差 V）、緑または Y（輝度）、青または Cb（色差 U）。 R1,G1,B1,Y1,Cb1,Cr1 は、出力画像データの各成分。 GG,GB,GR,BG,BB,BR,RG,RB,RR,YY は、左記のシンボルの末尾に対応。
PFV_COLOR_MATRIX_GB	1	
PFV_COLOR_MATRIX_GR	2	
PFV_COLOR_MATRIX_BG	3	
PFV_COLOR_MATRIX_BB	4	
PFV_COLOR_MATRIX_BR	5	
PFV_COLOR_MATRIX_RG	6	
PFV_COLOR_MATRIX_RB	7	
PFV_COLOR_MATRIX_RR	8	
PFV_COLOR_MATRIX_YY	0	

6.6.8 pfv_idtrg_t

入力 FIFO が Empty と判定されるときの FIFO に空いているデータのバイト数。 FIFO のバイト数は 32。

定数名	値	内容
PFV_IDTRG_SPACED_2_BYTE	0	2 バイト
PFV_IDTRG_SPACED_4_BYTE	1	4 バイト
PFV_IDTRG_SPACED_16_BYTE	2	16 バイト
PFV_IDTRG_SPACED_32_BYTE	3	32 バイト

6.6.9 pfv_odtrg_t

出力 FIFO が Full と判定されるときの FIFO に入っているデータのバイト数。 FIFO のバイト数は 32。

定数名	値	内容
PFV_ODTRG_FILLED_2_BYTE	0	2 バイト
PFV_ODTRG_FILLED_4_BYTE	1	4 バイト
PFV_ODTRG_FILLED_16_BYTE	2	16 バイト
PFV_ODTRG_FILLED_32_BYTE	3	32 バイト

6.6.10 pfv_interrupt_line_t

割込み線の種類。

定数名	値	内容
PFV_INTERRUPT_LINE_PFVEIn	1	PFVEI 割込み。PFV エラー
PFV_INTERRUPT_LINE_IFEIn	2	IFEI 割込み。入力 FIFO エンプティ
PFV_INTERRUPT_LINE_OFFIn	4	OFFIn 割込み。出力 FIFO フル

6.6.11 pfv_interrupt_lines_t

pfv_interrupt_line_t 型の値を複数持つビットフラグ。

割込み線の許可または禁止を指定するのに使用します。

定数名	値	内容
PFV_INTERRUPT_LINE_ALL	-	すべての割込み線の種類

6.6.12 pfv_dmac_interrupt_unit_t

割込みが発生した周辺機能の種類。

定数名	値	内容
PFV_INTERRUPT_UNIT_PFV	1	PFV
PFV_INTERRUPT_UNIT_DMAC	2	DMAC

6.6.13 無名列挙体の値や定数

定数名	値	内容
PFV_CHANNEL_MIN	0	PFV のチャネル番号の最小値
PFV_CHANNEL_MAX	1	PFV のチャネル番号の最大値
PFV_INPUT_FIFO_FULL_BYTE	32	PFV の入力 FIFO のバイト数
PFV_OUTPUT_FIFO_FULL_BYTE	32	PFV の出力 FIFO のバイト数

6.7 構造体/共用体

章	型名	概要
6.7.1	pfv_dmac_config_t	R_PFV_DMAC_Initialize 関数のパラメーター
6.7.2	pfv_config_t	R_PFV_Initialize 関数のパラメーター
6.7.3	pfv_io_format_t	R_PFV_DMAC_SetIOFormat 関数のパラメーター
6.7.4	pfv_transfer_config_t	R_PFV_DMAC_Transfer 関数のパラメーター
6.7.5	pfv_color_matrix_t	R_PFV_DMAC_SetImageColorMatrix 関数のパラメーター
6.7.6	pfv_reset_color_matrix_t	カラーマトリックスのリセット値
6.7.7	pfv_dma_bit_count_t	DMA トランスファーサイズと、PFV の FIFO のフルとエンプティのレベル
*2	r_ospl_async_t	通知設定
*2	pfv_async_status_t	状態と割込みステータス。r_ospl_async_status_t 型と同じ

6.7.1 pfv_dmac_config_t

概　要　R_PFV_DMAC_Initialize 関数のパラメーター
ヘッダ　devdrv_pfv.h
説　明
メンバー変数

bit_flags_fast32_t　flags	フラグド構造体パラメーター。参考：(6.10.1) F_PFV_DMAC_INPUT_DMAC_CHANNEL F_PFV_DMAC_OUTPUT_DMAC_CHANNEL F_PFV_DMAC_RESET_COLOR_MATRIXES F_PFV_DMAC_GET_DMA_BIT_COUNT
int_fast32_t input_DMAC_channel	入力画像データをメモリから PFV に転送する DMAC のチャネル番号。 省略時は、DMAC を使わない。
int_fast32_t output_DMAC_channel	出力画像データを PFV からメモリに転送する DMAC のチャネル番号。 省略時は、DMAC を使わない。
pfv_color_matrix_sub_t* reset_color_matrixes	カラーマトリックスのリセット値。参照：(6.7.6)
pfv_get_dma_bit_count_func_t get_dma_bit_count_func	R_Userdef_PFV_GetDMA_BitCount 関数の代わりに使用する関数

6.7.2 pfv_config_t

概　要　R_PFV_Initialize 関数のパラメーター
ヘッダ　devdrv_pfv.h
説　明
メンバー変数

bit_flags_fast32_t　flags	フラグド構造体パラメーター。参考：(6.10.1) F_PFV_RESET_COLOR_MATRIXES
pfv_color_matrix_sub_t* reset_color_matrixes	カラーマトリックスのリセット値。参照：(6.7.6)

6.7.3 pfv_io_format_t

概　要　R_PFV_DMAC_SetIOFormat 関数などのパラメーター

*2 RZ/A1H グループ OS 移植層 OSPL （R01AN1887JJ）を参照。

ヘッダ　devdrv_pfv.h

説　明

メンバー変数

bit_flags_fast32_t flags	フラグド構造体パラメーター。参考：(6.10.1) F_PFV_INPUT_FORMAT F_PFV_INPUT_SWAP F_PFV_OUTPUT_FORMAT F_PFV_OUTPUT_SWAP F_PFV_OUTPUT_ALPHA F_PFV_IS_DITHER F_PFV_IMAGE_WIDTH F_PFV_IMAGE_HEIGHT
pfv_format_t input_format	入力する画像のピクセルフォーマット。 省略時は、設定を変えない。 初期値は、PFV_RGB888。
pfv_swap_t input_swap	入力する画像のピクセルフォーマットのバイトオーダーの入れ替え。 省略時は、input_format メンバー変数に応じて、以下のデフォルトのオーダーになる。PFV_SWAP_ARGB8888、PFV_SWAP_RGB565_PIXEL10、PFV_SWAP_YCbCr422_Cb_Y0_Cr_Y1。
pfv_format_t output_format	出力する画像のピクセルフォーマット。 省略時は、設定を変えない。 初期値は、PFV_ARGB8888。
pfv_swap_t output_swap	出力する画像のピクセルフォーマットのバイトオーダーの入れ替え。 省略時は、output_format メンバー変数に応じて、以下のデフォルトのオーダーになる。PFV_SWAP_ARGB8888、PFV_SWAP_RGB565_PIXEL10、PFV_SWAP_YCbCr422_Cb_Y0_Cr_Y1。
int_t　output_alpha	出力する画像のアルファ成分の値。output_format が PFV_ARGB8888 のときのみ有効。 値の範囲は、0～255。 省略時は、255。
bool_t　is_dither	ディザをかけるかどうか。output_format が PFV_RGB565 のときのみ有効。ただし、出力データの色のビット数より、入力データの色のビット数が大きくないときは、有効にしてもディザの効果は表れません。 省略時は、設定を変えない。 初期値は、true。
int_t　image_width	画像の幅（ピクセル） 本変数はディザを使用するときのみ必要。 省略時は、設定を変えない。 初期値は、0。 R_PFV_DMAC_Transfer 関数の pfv_transfer_config_t::buffer_width を指定したときは、その値に本設定が上書きされる。
int_t　image_height	画像の高さ（ピクセル） 本変数はディザを使用するときのみ必要。 省略時は、設定を変えない。 初期値は、0。

	R_PFV_DMAC_Transfer 関数の pfv_transfer_config_t::buffer_height を指定したときは、その値に本設定が上書きされる。

6.7.4 pfv_transfer_config_t

概　要　R_PFV_DMAC_Transfer 関数のパラメーター

ヘッダ　devdrv_pfv.h

説　明

メンバー変数

bit_flags_fast32_t　flags	フラグド構造体パラメーター。参考：(6.10.1) F_PFV_TRANSFER_INPUT_BUFFER_ADDRESS F_PFV_TRANSFER_INPUT_BUFFER_SIZE F_PFV_TRANSFER_OUTPUT_BUFFER_ADDRESS F_PFV_TRANSFER_OUTPUT_BUFFER_SIZE F_PFV_TRANSFER_BUFFER_WIDTH F_PFV_TRANSFER_BUFFER_HEIGHT
uintptr_t input_buffer_address	入力画像が入ったメモリの先頭物理アドレス。 省略時は、DMAC を使用しません。省略時は、R_PFV_WritePixelDataViaPIO 関数を使用して、PFV に直接ライトしてください。
size_t input_buffer_size	入力画像が入ったメモリのサイズ（バイト）。 input_buffer_address を指定したときは必須。 buffer_width, buffer_height メンバー変数が指定されたときは、本変数はサイズが十分にあるかをチェックするために使われ、変換するサイズは buffer_width, buffer_height メンバー変数から計算されたサイズになります。
uintptr_t output_buffer_address	出力画像が入ったメモリの先頭物理アドレス。 省略時は、DMAC を使用しません。省略時は、R_PFV_ReadPixelDataViaPIO 関数を使用して、PFV から直接リードしてください。
size_t output_buffer_size	出力画像が入るメモリのサイズ（バイト） ouput_buffer_address を指定したときは必須。 buffer_width, buffer_height メンバー変数を省略したときに、本変数に設定したサイズが出力画像サイズより大きくすると、転送が完了したと判定しません。 buffer_width, buffer_height メンバー変数が指定されたときは、本変数はサイズが十分にあるかをチェックするために使われ、変換するサイズは buffer_width, buffer_height メンバー変数から計算されたサイズになります。 R_PFV_DMAC_Transfer 関数から返った後は、本変数の値は、出力画像のサイズになります。
int_t　buffer_width	画像の幅（ピクセル） 省略時は、画像の幅と高さから計算されたサイズではなく、input_buffer_size に指定したサイズだけ変換する。
int_t　buffer_height	画像の高さ（ピクセル） buffer_width を指定したときは必須。
int_fast32_t interval_count_of_DMA	DMAC の CHITVL_n レジスタの ITVL ビットに設定する値。 インターバル･カウント。１回のリードまたはライトが完了してから次のリードまたはライトまで待つカウント数。単位は、Bφクロック。省略時は 0。

r_ospl_axi_cache_attribute_t source_AXI_cache_attribute	DMAC の CHEXT_n レジスタの SCA ビットに設定する値。 DMAC がリードするときの AXI バスの先にある L2 キャッシュのキャッシュ属性。省略時は、RZ/A1H の内部バス用の R_OSPL_AXI_CACHE_ZERO。
r_ospl_axi_protection_t source_AXI_protection	DMAC の CHEXT_n レジスタの SPR ビットに設定する値。 DMAC がリードするときの AXI バスの先にある L2 キャッシュの保護属性。省略時は、RZ/A1H の内部バス用の R_OSPL_AXI_PROTECTION_ZERO。
r_ospl_axi_cache_attribute_t destination_AXI_cache_attribute	DMAC の CHEXT_n レジスタの DCA ビットに設定する値。 DMAC がライトするときの AXI バスの先にある L2 キャッシュのキャッシュ属性。省略時は、RZ/A1H の内部バス用の R_OSPL_AXI_CACHE_ZERO。
r_ospl_axi_protection_t destination_AXI_protection	DMAC の CHEXT_n レジスタの DPR ビットに設定する値。 DMAC がライトするときの AXI バスの先にある L2 キャッシュの保護属性。省略時は、RZ/A1H の内部バス用の R_OSPL_AXI_PROTECTION_ZERO。

6.7.5 pfv_color_matrix_t

概　要　R_PFV_DMAC_SetImageColorMatrix 関数などのパラメーター

ヘッダ　devdrv_pfv.h

説　明　VDC5 ドライバの vdc5_color_matrix_t 型と同じ構成です。vdc5_color_matrix_t 型の変数を pfv_color_matrix_t 型にキャストして使用することができます。

メンバー変数

int32_t　dummy	未使用。vdc5_color_matrix_t 型と互換を持たせるため。
pfv_color_matrix_mode_t mode	画像データのピクセルフォーマットの変換の種類。参考：(6.6.5)
uint16_t dc_offset_plus_128[ ]	カラーマトリックスのオフセット成分に 128 を加算した値の配列。 各成分が加算されます。ブライト調整に使用します。 配列番号は、(6.6.6)pfv_dc_offset_index_t 型。
int16_t matrix_multiply_256[ ]	カラーマトリックスのゲイン成分に 256 を乗算した値の配列。 各成分が積算されます。ゲイン調整に使用します。 配列番号は、(6.6.7)pfv_matrix_multiply_index_t 型。

サンプル：

```
pfv_color_matrix_t  unit_matrix = /* RGB→RGB のオフセット 0、ゲイン 1 倍 */
{
  0,  /* .dummy */
  PFV_GBR_TO_GBR,  /* .mode */
  { /* .dc_offset_plus_128[ ] */
    128,  /* [ PFV_DC_OFFSET_Y_OR_GREEN ] */
    128,  /* [ PFV_DC_OFFSET_BLUE ] */
    128   /* [ PFV_DC_OFFSET_RED ] */
  },
  { /* .matrix_multiply_256[ ] */
    256,  /* [ PFV_COLOR_MATRIX_GG ] */
      0,  /* [ PFV_COLOR_MATRIX_GB ] */
      0,  /* [ PFV_COLOR_MATRIX_GR ] */
      0,  /* [ PFV_COLOR_MATRIX_BG ] */
    256,  /* [ PFV_COLOR_MATRIX_BB ] */
```

```
      0,  /* [ PFV_COLOR_MATRIX_BR ] */
      0,  /* [ PFV_COLOR_MATRIX_RG ] */
      0,  /* [ PFV_COLOR_MATRIX_RB ] */
    256,  /* [ PFV_COLOR_MATRIX_RR ] */
  }
};
```

6.7.6 pfv_reset_color_matrix_t

概　要　カラーマトリックスのリセット値

ヘッダ　r_pfv.h

説　明　要素数が PFV_MATRIX_MODE_COUNT の pfv_reset_color_matrix_t 型の配列として使います。配列の要素番号は、pfv_color_matrix_mode_t 型です。

メンバー変数

uint16_t dc_offset_plus_128[ ]	カラーマトリックスのオフセット成分に 128 を加算した値の配列。 各成分が加算されます。ブライト調整に使用します。 配列番号は、(6.6.6)pfv_dc_offset_index_t 型。
int16_t matrix_multiply_256[ ]	カラーマトリックスのゲイン成分に 256 を乗算した値の配列。 各成分が積算されます。ゲイン調整に使用します。 配列番号は、(6.6.7)pfv_matrix_multiply_index_t 型。

サンプル：

```
#define  NUM_2048  2048  /* if ( matrix_multiply_256 < 0 )  matrix_multiply_256
= register_value - 2048 */
static const pfv_reset_color_matrix_t
gs_ResetColorMatrixes[ PFV_MATRIX_MODE_COUNT ] =
{
    {   /* GBR => GBR */
        { 128, 128, 128 },
        { 256, 0, 0, 0, 256, 0, 0, 0, 256 }
    },
    {   /* GBR => YCBCR, SMPTE 293M */
        { 128, 128, 128 },
        { 150, 29, 77, 1963-NUM_2048, 128, 2005-NUM_2048, 1941-NUM_2048,
          2027-NUM_2048, 128 }
    },
    {   /* YCBCR => GBR, SMPTE 293M */
        { 128, 128, 128 },
        { 256, 1960-NUM_2048, 1865-NUM_2048, 256, 454, 0, 256, 0, 359 }
    },
    {   /* YCBCR => YCBCR */
        { 128, 128, 128 },
        { 256, 0, 0, 0, 256, 0, 0, 0, 256 }
    }
};
#undef  NUM_2048
```

6.7.7 pfv_dma_bit_count_t

概　要　DMA トランスファーサイズと、PFV の FIFO のフルとエンプティのレベル。

ヘッダ　r_pfv.h

説　明　R_Userdef_PFV_GetDMA_BitCount 関数の引数です。(6.9.5(11))

メンバー変数

dmac_bit_count_t input_buffer_DMA_bit_count	入力バッファーに対する DMA トランスファーサイズ。参考：RZ/A1H グループ DMAC_TEMPORARY サンプルプログラム(PFV 付属)（R01AN1888JJ）

dmac_bit_count_t input_PFV_DMA_bit_count	PFV の入力 FIFO に対する DMA トランスファーサイズ
pfv_idtrg_t input_trigger_byte_count	PFV の入力 FIFO がエンプティになるバイト数。参考：(6.6.8)
dmac_bit_count_t output_PFV_DMA_bit_count	PFV の出力 FIFO に対する DMA トランスファーサイズ
dmac_bit_count_t output_buffer_DMA_bit_count	出力バッファーに対する DMA トランスファーサイズ
pfv_odtrg_t output_trigger_byte_count	PFV の出力 FIFO がフルになるバイト数。参考：(6.6.9)

6.8 変数

表 6.6 に static 型変数を、表 6.7 に const 型変数を示します。

表 6.6 static 型変数

型	変数名	内容	使用関数
pfv_dmac_contexts_t	gs_pfv_dmac_contexts	PFV-DMAC	各種
pfv_contexts_t	gs_pfv_contexts	PFV	各種
r_dmac_temporary_channel_t	gs_dmac_temporary_channel	DMAC	各種

表 6.7 const 型変数

型	変数名	内容	使用関数
なし			

6.9 関数

6.9.1 一覧

章	分類
(1)	PFV と DMAC が連携する関数
(2)	PFV 単体に関する関数
(3)	初期化前でも使える PFV 単体に関する関数
(4)	ドライバ移植層の関数

(1) PFV と DMAC が連携する関数の一覧

章	関数名	概要
6.9.2(1)	R_PFV_DMAC_Initialize	PFV と入力用と出力用の DMAC を初期化します
6.9.2(2)	R_PFV_DMAC_Terminate	PFV と入力用と出力用の DMAC の終了処理をします
6.9.2(3)	R_PFV_DMAC_SetIOFormat	画像データのピクセルフォーマットを変換することに関する設定をします
6.9.2(4)	R_PFV_DMAC_GetIOFormat	画像データのピクセルフォーマットを変換することに関する設定を取得します
6.9.2(5)	R_PFV_DMAC_ResetImageColorMatrix	カラーマトリックスをリセットします
6.9.2(6)	R_PFV_DMAC_SetImageColorMatrix	カラーマトリックスを設定します
6.9.2(7)	R_PFV_DMAC_GetImageColorMatrix	カラーマトリックスの設定を取得します
6.9.2(8)	R_PFV_DMAC_Transfer	画像データの変換をします。（同期関数）
6.9.2(9)	R_PFV_DMAC_TransferAsync	画像データの変換をします。（非同期関数）
6.9.2(10)	R_PFV_DMAC_OnInterrupting	割込みを受信します。
6.9.2(11)	R_PFV_DMAC_OnInterrupted	割込み応答処理を行います。
6.9.2(12)	R_PFV_DMAC_StopTransfer	画像データの変換を中止します

(2) PFV 単体に関する関数の一覧

章	関数名	概要
6.9.3(1)	R_PFV_Initialize	PFV を初期化します
6.9.3(2)	R_PFV_Terminate	PFV の終了処理をします
6.9.3(3)	R_PFV_SetIOFormat	画像データのピクセルフォーマットを変換することに関する設定をします
6.9.3(4)	R_PFV_GetIOFormat	画像データのピクセルフォーマットを変換することに関する設定を取得します
6.9.3(5)	R_PFV_ResetImageColorMatrix	カラーマトリックスをリセットします
6.9.3(6)	R_PFV_SetImageColorMatrix	カラーマトリックスを設定します
6.9.3(7)	R_PFV_GetImageColorMatrix	カラーマトリックスの設定を取得します
6.9.3(8)	R_PFV_GetFilledByteInInputFIFO	PFV の入力 FIFO に埋まっているデータのバイト数を取得します。
6.9.3(9)	R_PFV_WritePixelDataViaPIO	PFV に画像データをライトします
6.9.3(10)	R_PFV_GetFilledByteInOutputFIFO	PFV の出力 FIFO に埋まっているデータのバイト数を取得します。
6.9.3(11)	R_PFV_ReadPixelDataViaPIO	PFV から画像データをリードします

6.9.3(12)	R_PFV_OnInterrupting	割込みを受信します。
6.9.3(13)	R_PFV_OnInterrupted	割込み応答処理を行います。
6.9.3(14)	R_PFV_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ーを取得します。

(3) 初期化前でも使える PFV 単体に関する関数の一覧

章	関数名	概要
6.9.4(1)	R_PFV_STATIC_GetColorMatrixMode	適切な pfv_color_matrix_mode_t 型の値を取得します

(4) ドライバ移植層の関数の一覧

章	関数名	概要
6.9.5(1)	R_Userdef_PFV_SetDefaultAsync	r_ospl_async_t 型の構造体のデフォルト値を設定します
6.9.5(2)	R_Userdef_PFV_OnInitialize	ドライバ移植層の初期化をします
6.9.5(3)	R_Userdef_PFV_OnFinalize	ドライバ移植層の終了処理をします
6.9.5(4)	R_Userdef_PFV_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
6.9.5(5)	R_Userdef_PFV_OnEnableInterrupt	割込みを許可します
6.9.5(6)	R_Userdef_PFV_OnDisableInterrupt	割込みを禁止します
6.9.5(7)	R_Userdef_PFV_OnInterruptDefault	デフォルトの割込みコールバック関数
6.9.5(8)	R_Userdef_PFV_DMAC_SetDefaultConfig	pfv_dmac_config_t 型の構造体のデフォルト値を設定します
6.9.5(9)	R_Userdef_PFV_SetDefaultConfig	pfv_config_t 型の構造体のデフォルト値を設定します
6.9.5(10)	R_Userdef_PFV_DMAC_GetInterruptUnit	割込みが発生した周辺機能の種類を取得します
6.9.5(11)	R_Userdef_PFV_GetDMA_BitCount	DMA トランスファーサイズと、PFV の FIFO のフルとエンプティのレベルを取得します

6.9.2　PFV と DMAC が連携する関数

(1)　R_PFV_DMAC_Initialize

概　要	PFV と入力用と出力用の DMAC を初期化します。
ヘッダ	r_pfv.h
宣　言	errnum_t　R_PFV_DMAC_Initialize(int_fast32_t　pfv_channel, pfv_dmac_config_t* in_out_config);
説　明	内部変数を初期化します。 入力用と出力用の DMAC とは、入力画像データをメモリから PFV に転送する DMAC と、出力画像データを PFV からメモリに転送する DMAC です。 PFV から出力される割込み（IFEI、OFFI）は、DMAC に通知する設定になり、CPU には入りません。 カラーマトリックスの要素をすべて 0 にします。

引　数		
	int_fast32_t pfv_channel	PFV のチャネル番号。 PFV_CHANNEL_MIN～PFV_CHANNEL_MAX。
	pfv_dmac_config_t* in_out_config	その他の設定値。参照：(6.7.1) NULL 不可。
リターン値	エラーコード。エラーなし＝0	

(2)　R_PFV_DMAC_Terminate

概　要	PFV と入力用と出力用の DMAC の終了処理をします。
ヘッダ	r_pfv.h
宣　言	errnum_t　R_PFV_DMAC_Terminate(int_fast32_t　pfv_channel);
説　明	DMAC が動いているときは、中断します。

引　数		
	int_fast32_t　pfv_channel	PFV のチャネル番号。
リターン値	エラーコード。エラーなし＝0	

(3)　R_PFV_DMAC_SetIOFormat

概　要	画像データのピクセルフォーマットを変換することに関する設定をします。
ヘッダ	r_pfv.h
宣　言	errnum_t　R_PFV_DMAC_SetIOFormat(int_fast32_t　pfv_channel, pfv_io_format_t* in_out_io_format);
説　明	カラーマトリックスのリセットも行います。 参考：(5)R_PFV_DMAC_ResetImageColorMatrix

引　数		
	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_io_format_t* in_out_io_format	設定。参考：(6.7.3) NULL 不可。
リターン値	エラーコード。エラーなし＝0	

(4)　R_PFV_DMAC_GetIOFormat

概　要	画像データのピクセルフォーマットを変換することに関する設定を取得します。
ヘッダ	r_pfv.h
宣　言	errnum_t　R_PFV_DMAC_GetIOFormat(int_fast32_t　pfv_channel, pfv_io_format_t* out_io_format);
説　明	

引　数		
	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_io_format_t* out_io_format	（出力）設定。参考：(6.7.3)
リターン値	エラーコード。エラーなし＝0	

(5)　R_PFV_DMAC_ResetImageColorMatrix

概　要	カラーマトリックスをリセットします。
ヘッダ	r_pfv.h

宣　言　errnum_t　R_PFV_DMAC_ResetImageColorMatrix(int_fast32_t　pfv_channel, pfv_color_matrix_mode_t　mode);

説　明　オフセットは０、ゲインは１倍になります。
RGB 系と YCbCr 系の変換をするときは、カラーマトリックスの値が、SMPTE 293M に定義されている変換式になります。

引　数

int_fast32_t　pfv_channel	PFV のチャネル番号。
pfv_color_matrix_mode_t　mode	画像データのピクセルフォーマットの変換の種類。参考：(6.6.5)

リターン値　エラーコード。エラーなし＝0

(6)　R_PFV_DMAC_SetImageColorMatrix

概　要　カラーマトリックスを設定します。

ヘッダ　r_pfv.h

宣　言　errnum_t　R_PFV_DMAC_SetImageColorMatrix(int_fast32_t　pfv_channel, pfv_color_matrix_t* offset_and_matrix);

説　明

引　数

int_fast32_t　pfv_channel	PFV のチャネル番号。
pfv_color_matrix_t* offset_and_matrix	カラーマトリックス。参考：(6.7.5)

リターン値　エラーコード。エラーなし＝0

(7)　R_PFV_DMAC_GetImageColorMatrix

概　要　カラーマトリックスの設定を取得します。

ヘッダ　r_pfv.h

宣　言　errnum_t　R_PFV_DMAC_GetImageColorMatrix(int_fast32_t　pfv_channel, pfv_color_matrix_t* out_offset_and_matrix);

説　明

引　数

int_fast32_t　pfv_channel	PFV のチャネル番号。
pfv_color_matrix_t* out_offset_and_matrix	（出力）カラーマトリックス。参考：(6.7.5)

リターン値　エラーコード。エラーなし＝0

(8)　R_PFV_DMAC_Transfer

概　要　画像データの変換をします。（同期関数）

ヘッダ　r_pfv.h

宣　言　errnum_t　R_PFV_DMAC_Transfer(int_fast32_t　pfv_channel, pfv_transfer_config_t* in_out_config);

説　明　DMAC を使用して PFV を使用した変換をします。
変換が完了するまでリターンしない同期関数です。

引　数

int_fast32_t　pfv_channel	PFV のチャネル番号。
pfv_transfer_config_t* in_out_config	設定。参考：(6.7.4)

リターン値　エラーコード。エラーなし＝0

(9)　R_PFV_DMAC_TransferAsync

概　要　画像データの変換をします。（非同期関数）

ヘッダ　r_pfv.h

宣　言　errnum_t　R_PFV_DMAC_TransferAsync(int_fast32_t　pfv_channel, pfv_transfer_config_t* in_out_config,　r_ospl_async_t* async);

説　明　DMAC を使用して PFV を使用した変換を開始します。

変換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_TransferStart 関数の説明を参照

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_transfer_config_t* in_out_config	設定。参考：(6.7.4)
	r_ospl_async_t* async	通知設定
リターン値	エラーコード。エラーなし＝0	

(10) R_PFV_DMAC_OnInterrupting

概　要　割込みを受信します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_DMAC_OnInterrupting(const r_ospl_interrupt_t* 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。
本関数の中から、発生した割込みを受信する R_PFV_OnInterrupting 関数、または、R_DMAC_TEMPORARY_OnInterrupting 関数を呼び出します。

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	

(11) R_PFV_DMAC_OnInterrupted

概　要　割込み応答処理を行います。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_DMAC_OnInterrupted(int_fast32_t　pfv_channel);
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。
本関数の中から、発生した割込みの応答処理をする R_PFV_OnInterrupted 関数、または、R_DMAC_TEMPORARY_OnInterrupted 関数を呼び出します。

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号
リターン値	エラーコード。エラーなし＝0	

(12) R_PFV_DMAC_StopTransfer

概　要　画像データの変換を中止します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_DMAC_StopTransfer(int_fast32_t　pfv_channel);
説　明

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
リターン値	エラーコード。エラーなし＝0	

6.9.3　PFV 単体に関する関数

(1) R_PFV_Initialize

概　要　PFV を初期化します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_Initialize(int_fast32_t　pfv_channel,　pfv_config_t* in_out_config);
説　明　内部変数と PFV を初期化します。
内部からドライバ移植層の R_Userdef_PFV_OnInitialize を呼び出します。
カラーマトリックスの要素をすべて 0 にします。

引　数	int_fast32_t pfv_channel	PFV のチャネル番号。 PFV_CHANNEL_MIN～PFV_CHANNEL_MAX。

	pfv_config_t* in_out_config	設定。参照：(6.7.2) NULL 可能。
リターン値	エラーコード。エラーなし＝0	

(2) R_PFV_Terminate

概　要	PFV の終了処理をします。	
ヘッダ	r_pfv.h	
宣　言	errnum_t　R_PFV_Terminate(int_fast32_t　pfv_channel);	
説　明	内部からドライバ移植層の R_Userdef_PFV_OnFinalize を呼び出します。	
引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
リターン値	エラーコード。エラーなし＝0	

(3) R_PFV_SetIOFormat

概　要	画像データのピクセルフォーマットを変換することに関する設定をします。	
ヘッダ	r_pfv.h	
宣　言	errnum_t　R_PFV_SetIOFormat(int_fast32_t　pfv_channel,　pfv_io_format_t* in_out_io_format);	
説　明	カラーマトリックスのリセットも行います。 参考：(5)R_PFV_ResetImageColorMatrix	
引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_io_format_t* in_out_io_format	設定。参考：(6.7.3) NULL 不可。
リターン値	エラーコード。エラーなし＝0	

(4) R_PFV_GetIOFormat

概　要	画像データのピクセルフォーマットを変換することに関する設定を取得します。	
ヘッダ	r_pfv.h	
宣　言	errnum_t　R_PFV_GetIOFormat(int_fast32_t　pfv_channel,　pfv_io_format_t* out_io_format);	
説　明		
引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_io_format_t* out_io_format	（出力）設定。参考：(6.7.3)
リターン値	エラーコード。エラーなし＝0	

(5) R_PFV_ResetImageColorMatrix

概　要	カラーマトリックスをリセットします。	
ヘッダ	r_pfv.h	
宣　言	errnum_t　R_PFV_ResetImageColorMatrix(int_fast32_t　pfv_channel, pfv_color_matrix_mode_t　mode);	
説　明	オフセットは０、ゲインは１倍になります。 RGB 系と YCbCr 系の変換をするときは、カラーマトリックスの値が、SMPTE 293M に定義されている変換式になります。	
引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_color_matrix_mode_t　mode	画像データのピクセルフォーマットの変換の種類。参考：(6.6.5)
リターン値	エラーコード。エラーなし＝0	

(6) R_PFV_SetImageColorMatrix

概　要	カラーマトリックスを設定します。
ヘッダ	r_pfv.h
宣　言	errnum_t　R_PFV_SetImageColorMatrix(int_fast32_t　pfv_channel, pfv_color_matrix_t* offset_and_matrix);
説　明	

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_color_matrix_t* offset_and_matrix	カラーマトリックス。参考：(6.7.5)
リターン値	エラーコード。エラーなし＝0	

(7)　R_PFV_GetImageColorMatrix

概　要　カラーマトリックスの設定を取得します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_GetImageColorMatrix(int_fast32_t　pfv_channel, pfv_color_matrix_t* out_offset_and_matrix);
説　明

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_color_matrix_t* out_offset_and_matrix	（出力）カラーマトリックス。参考：(6.7.5)
リターン値	エラーコード。エラーなし＝0	

(8)　R_PFV_GetFilledByteInInputFIFO

概　要　PFV の入力 FIFO に埋まっているデータのバイト数を取得します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_GetFilledByteInInputFIFO(int_fast32_t　pfv_channel,　int_t* out_filled_byte);
説　明　PFV の入力 FIFO の最大バイト数は、PFV_INPUT_FIFO_FULL_BYTE です。
PFV にエラーが発生していたら、本関数のリターン値はエラーになります。

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
	int_t* out_filled_byte	（出力）PFV の入力 FIFO に埋まっているバイト数
リターン値	エラーコード。エラーなし＝0	

(9)　R_PFV_WritePixelDataViaPIO

概　要　PFV に変換前の画像データをライトします。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_WritePixelDataViaPIO(int_fast32_t　pfv_channel,　uint32_t pixel_data,　int_t　bit_count);
説　明　入力 FIFO が満杯のときは、エラーになります。R_PFV_GetFilledByteInInputFIFO 関数を使用して FIFO が空いているか確認してください。

引　数	int_fast32_t pfv_channel	PFV のチャネル番号。
	uint32_t　pixel_data	１ピクセル分の画像データ ただし、PFV_YCbCr422 のときは２ピクセル分。
	int_t　bit_count	pixel_data に渡した画像データのビット数。 PFV_RGB565 のときは 16。その他のときは 32。
リターン値	エラーコード。エラーなし＝0	

(10)　R_PFV_GetFilledByteInOutputFIFO

概　要　PFV の出力 FIFO に埋まっているデータのバイト数を取得します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_GetFilledByteInOutputFIFO(int_fast32_t　pfv_channel,　int_t* out_filled_byte);
説　明　PFV の出力 FIFO の最大バイト数は、PFV_OUTPUT_FIFO_FULL_BYTE です。
PFV にエラーが発生していたら、本関数のリターン値はエラーになります。

引　数	int_fast32_t pfv_channel	PFV のチャネル番号。
	int_t* out_filled_byte	（出力）PFV の出力 FIFO に埋まっているバイト数
リターン値	エラーコード。エラーなし＝0	

(11) R_PFV_ReadPixelDataViaPIO

概　要　PFV から変換後の画像データをリードします。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_ReadPixelDataViaPIO(int_fast32_t　pfv_channel,　uint32_t* out_pixel_data,　int_t　bit_count);
説　明　出力 FIFO が空のときは、エラーになります。R_PFV_GetFilledByteInOutputFIFO 関数を使用して FIFO にデータが入っているか確認してください。

引　数	int_fast32_t pfv_channel	PFV のチャネル番号。
	uint32_t* out_pixel_data	（出力）１ピクセル分の画像データ ただし、PFV_YCbCr422 のときは２ピクセル分。
	int_t　bit_count	pixel_data に渡した画像データのビット数。 PFV_RGB565 のときは 16。その他のときは 32。
リターン値	エラーコード。エラーなし＝0	

(12) R_PFV_OnInterrupting

概　要　割込みを受信します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_OnInterrupting(const r_ospl_interrupt_t* 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、pfv_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 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	

(13) R_PFV_OnInterrupted

概　要　割込み応答処理を行います。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_OnInterrupted(int_fast32_t const　pfv_channel);
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_PFV_OnInterrupting 関数によって 1 に設定された pfv_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 OSPL （R01AN1887JJ）にある R_DRIVER_OnInterrupted 関数の説明を参照。

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号。
リターン値	エラーコード。エラーなし＝0	

(14) R_PFV_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ーを取得します。
ヘッダ　r_pfv.h
宣　言　errnum_t　R_PFV_GetAsyncStatus(int_fast32_t　pfv_channel,　const pfv_async_status_t** out_Status);
説　明　out_Status 引数に指定するポインター変数には、const 修飾子が必要です。

引　数	int_fast32_t　pfv_channel	PFV のチャネル番号

	pfv_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ー
リターン値	エラーコード。エラーなし＝0	

6.9.4 初期化前でも使える PFV 単体に関する関数

(1) R_PFV_STATIC_GetColorMatrixMode

概　要　適切な pfv_color_matrix_mode_t 型の値を取得します。

ヘッダ　r_pfv.h

宣　言　errnum_t　R_PFV_STATIC_GetColorMatrixMode(pfv_format_t　input_format, pfv_format_t　output_format,　pfv_color_matrix_mode_t* out_color_matrix_mode);

説　明

引　数	pfv_format_t　input_format	PFV に入力するピクセルフォーマット
	pfv_format_t　output_format	PFV から出力するピクセルフォーマット
	pfv_color_matrix_mode_t* out_color_matrix_mode	参考：(6.6.5)pfv_color_matrix_mode_t
リターン値	エラーコード。エラーなし＝0	

6.9.5 ドライバ移植層の関数

(1) R_Userdef_PFV_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_pfv_pl.h

宣　言　void　R_Userdef_PFV_SetDefaultAsync(r_ospl_async_t* Async);

説　明　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します。
ReturnValue メンバー変数は、コールバック元のドライバで初期化します。

引　数	r_ospl_async_t* Async	通知設定、NULL 不可
リターン値	なし	

(2) R_Userdef_PFV_OnInitialize

概　要　ドライバ移植層の初期化をします。

ヘッダ　r_pfv_pl.h

宣　言　errnum_t　R_Userdef_PFV_OnInitialize(int_fast32_t　pfv_channel);

説　明　必要ならクロックを供給する設定をしてください。
R_PFV_DMAC_Initialize、R_PFV_Initialize 関数から呼ばれます。

引　数	int_fast32_t pfv_channel	PFV のチャネル番号。
リターン値	エラーコード。エラーなし＝0	

(3) R_Userdef_PFV_OnFinalize

概　要　ドライバ移植層の終了処理をします。

ヘッダ　r_pfv_pl.h

宣　言　errnum_t　R_Userdef_PFV_OnFinalize (int_fast32_t　pfv_channel, errnum_t e);

説　明　必要ならクロック供給を停止する設定をしてください。
R_PFV_DMAC_Terminate、R_PFV_Terminate 関数から呼ばれます。

引　数	int_fast32_t pfv_channel	PFV のチャネル番号。
	errnum_t e	これまでに発生したエラーコード。 エラー無し=0
リターン値	エラーコードまたは e。 0=「成功かつ e=0」	

(4) R_Userdef_PFV_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_pfv_pl.h
宣　言　errnum_t　R_Userdef_PFV_SetInterruptCallbackCaller(int_fast32_t pfv_channel, const r_ospl_caller_t* caller);
説　明　割込みがある非同期処理を開始するたびに呼ばれます。
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にしてください。詳細は、RZ/A1H グループ OS 移植層 OSPL（R01AN1887JJ）にある R_OSPL_CallInterruptCallback 関数の説明を参照。
ドライバドライバ本関数の内部で、チャンネル番号はチェックしなくて構いません。

引　数		
	int_fast32_t　pfv_channel	PFV のチャネル番号。
	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	

(5)　R_Userdef_PFV_OnEnableInterrupt

概　要　割込みを許可します。
ヘッダ　r_pfv_pl.h
宣　言　void　R_Userdef_PFV_OnEnableInterrupt(int_fast32_t pfv_channel, pfv_interrupt_lines_t enables);
説　明　本関数の内部で、チャンネル番号をチェックしなくて構いません。

引　数		
	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_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
リターン値	エラーコード。エラーなし＝0	

(6)　R_Userdef_PFV_OnDisableInterrupt

概　要　割込みを禁止します。
ヘッダ　r_pfv_pl.h
宣　言　void　R_Userdef_PFV_OnDisableInterrupt(int_fast32_t pfv_channel, pfv_interrupt_lines_t disables1);
説　明　本関数の内部で、チャンネル番号をチェックしなくて構いません。
R_Userdef_PFV_OnInitialize 関数が呼ばれる前に、本関数が呼ばれることがあります。

引　数		
	int_fast32_t　pfv_channel	PFV のチャネル番号。
	pfv_interrupt_lines_t disables1	禁止する割込み線の種類を 1 にしたビットフラグ
リターン値	エラーコード。エラーなし＝0	

(7)　R_Userdef_PFV_OnInterruptDefault

概　要　デフォルトの割込みコールバック関数。
ヘッダ　r_pfv_pl.h
宣　言　errnum_t　R_Userdef_PFV_OnInterruptDefault(const r_ospl_interrupt_t* interrupt_source, const r_ospl_caller_t* caller);
説　明　R_PFV_DMAC_TransferAsync 関数の 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登録されます。
本関数は、r_ospl_callback_t 型です。

引　数		
	r_ospl_interrupt_t* interrupt_source	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	

(8) R_Userdef_PFV_DMAC_SetDefaultConfig

概　要	pfv_dmac_config_t 型の構造体のデフォルト値を設定します。
ヘッダ	r_pfv_pl.h
宣　言	errnum_t　R_Userdef_PFV_DMAC_SetDefaultConfig(pfv_dmac_config_t* in_out_config);
説　明	pfv_dmac_config_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します。

引　数		
	pfv_dmac_config_t* in_out_config	設定、NULL 不可
リターン値	エラーコード。エラーなし＝0	

(9) R_Userdef_PFV_SetDefaultConfig

概　要	pfv_config_t 型の構造体のデフォルト値を設定します。
ヘッダ	r_pfv_pl.h
宣　言	errnum_t　R_Userdef_PFV_SetDefaultConfig(pfv_config_t* in_out_config);
説　明	pfv_config_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します。

引　数		
	pfv_config_t* in_out_config	設定、NULL 不可
リターン値	エラーコード。エラーなし＝0	

(10) R_Userdef_PFV_DMAC_GetInterruptUnit

概　要	割込みが発生した周辺機能の種類を取得します。
ヘッダ	r_pfv_pl.h
宣　言	errnum_t　R_Userdef_PFV_DMAC_GetInterruptUnit(r_ospl_interrupt_t* InterruptSource, r_pfv_dmac_interrupt_unit_t* out_Unit);
説　明	割込みコールバック関数から呼び出してください。 管理下に置いている周辺機能のうち、どの周辺機能の R_DRIVER_OnInterrupting 関数を呼び出すかを判定します。 内部では、InterruptSource 引数の IRQ_Num メンバー変数を大小比較することで、どの周辺機能から来た割込みであるかを判定します。

引　数		
	r_ospl_interrupt_t* InterruptSource	割込み発信元
	r_pfv_dmac_interrupt_unit_t* out_Unit	（出力）　周辺機能の種類
リターン値	エラーコード。エラーなし＝0	

(11) R_Userdef_PFV_GetDMA_BitCount

概　要	DMA トランスファーサイズと、PFV の FIFO のフルとエンプティのレベルを取得します。
ヘッダ	r_pfv_pl.h
宣　言	errnum_t　R_Userdef_PFV_GetDMA_BitCount(uintptr_t　input_buffer_start_address,　size_t　input_buffer_size, uintptr_t　output_buffer_start_address,　size_t　output_buffer_size, pfv_dma_bit_count_t*　out);
説　明	input_buffer_size=0 または output_buffer_size=0 のときは、PFV への入力または出力に DMAC を使いません。out 引数に出力する入力 DMAC または出力 DMAC に関する内容は、無視されます。 本関数は、pfv_get_dma_bit_count_func_t 型です。

引　数		
	uintptr_t input_buffer_start_address	入力バッファーの先頭の物理アドレス
	size_t　input_buffer_size	入力バッファーから転送するバイト数
	uintptr_t output_buffer_start_address	出力バッファーの先頭の物理アドレス
	size_t　output_buffer_size	出力バッファーへ転送するバイト数

	pfv_dma_bit_count_t* out	（出力）DMA トランスファーサイズと、PFV の FIFO のフルとエンプティのレベル。参考：(6.7.7)
リターン値	エラーコード。エラーなし＝0	

6.10 補足

6.10.1 フラグド構造体パラメーター

構造体の中の Flags メンバー変数をビットフラグとして使い、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、デフォルトの値が設定されたものと同じとします。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
1.03	2017.08.24	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
1.02	2016.02.29	L2 キャッシュ対応を追加。 初期設定例 1.01 に更新。 内部で使用する OSPL をバージョン 0.96 に更新。 内部で使用する DMAC ドライバをバージョン 1.02 に更新。
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（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、その他の不適切に使用しないでください。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、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、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。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。

   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. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）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、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
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. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

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。

(Rev.3.0-1 2016.11)

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/