

RX ファミリ

パラレルデータキャプチャユニット（PDC）モジュール Firmware Integration Technology

要旨

本アプリケーションノートでは、Firmware Integration Technology（以降 FIT と称します）を使用した Parallel Data Capture Unit（以降 PDC と称します）について説明します。本モジュールは、PDC を制御してカメラモジュールなどのイメージセンサから出力されたパラレルデータを取り込みます。以降、本モジュールを PDC FIT モジュールと称します。

本アプリケーションノートは、「RX ファミリ パラレルデータキャプチャユニット（PDC）モジュール Firmware Integration Technology（R01AN2220）」とは互換性がないのでご注意ください。

動作確認デバイス

以下は、この API によってサポートできるデバイスの一覧です。

RX64M

RX71M

RX651、RX65N

RX66N

RX72M

RX72N

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。

対象コンパイラ

Renesas Electronics C/C++ Compiler Package for RX Family

GCC for Renesas RX

IAR C/C++ Compiler for Renesas RX

各コンパイラの動作確認内容については「6.1動作確認環境」を参照してください。

関連ドキュメント

RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology（R01AN1685）

RX ファミリ DMA コントローラ DMACA 制御モジュール Firmware Integration Technology（R01AN2063）

RX Family DTC Module Using Firmware Integration Technology（R01AN1819）

（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。）

目次

1. 概要	3
1.1 PDC FIT モジュールとは	3
1.2 API の概要	4
2. API 情報.....	5
2.1 ハードウェアの要求	5
2.2 ソフトウェアの要求	5
2.3 サポートされているツールチェーン	5
2.4 使用する割り込みベクタ	5
2.5 ヘッドファイル	5
2.6 整数型	5
2.7 コンパイル時の設定	6
2.8 コードサイズ	6
2.9 引数	7
2.10 戻り値	10
2.11 コールバック関数	11
2.12 FIT モジュールの追加方法	13
2.13 for 文、while 文、do while 文について	14
3. API 関数.....	15
R_PDC_Open()	15
R_PDC_Close().....	23
R_PDC_Control()	24
R_PDC_GetFifoAddr().....	34
R_PDC_GetVersion().....	37
4. 端子設定	38
5. 使用方法	39
5.1 API 使用例	39
5.1.1 動作フロー例	39
6. 付録	40
6.1 動作確認環境	40
6.2 トラブルシューティング	43
改訂記録	44

1. 概要

PDC はイメージセンサなどの外部 IO と通信し、外部 IO から出力される画像などのパラレルデータを DTC または DMAC を介して内蔵 RAM、外部アドレス空間（CS 領域、SDRAM 領域）へ転送する機能を備えています。図 1.1 に PDC の概要を示します。

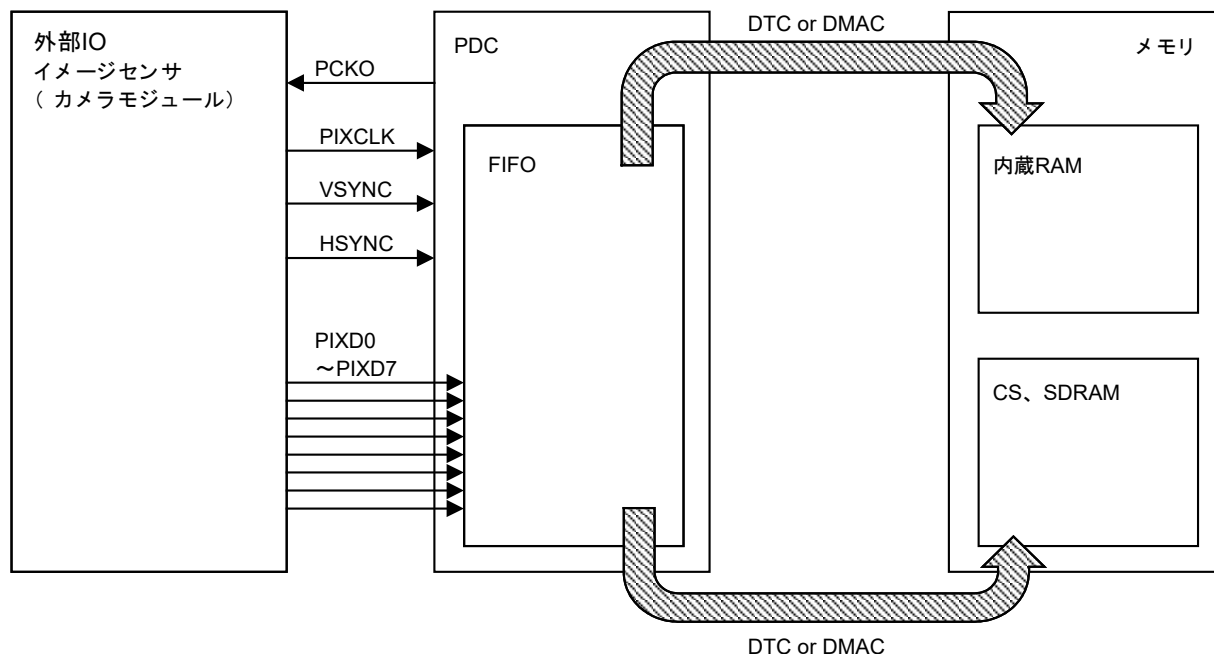


図1.1 PDC の概要

制限事項

本モジュールではr_bspのハードウェアロック機能を使用します。

1.1 PDC FIT モジュールとは

本モジュールはAPIとして、プロジェクトに組み込んで使用します。本モジュールの組み込み方については、「2.12 FIT モジュールの追加方法」を参照してください。

注意事項

PDC FIT モジュールのエンディアンは、コンパイラのエンディアン設定に合わせて、自動で切り替わります。本モジュールのみでは、イメージセンサからの画像データの取得を行うことができません。メモリへの転送はDMACまたはDTCを利用するため、必要に応じてそれぞれのFITモジュールのマニュアルを参照してプロジェクトに組み込んでください。イメージセンサの初期化プログラムと設定はご自身で用意してください。またイメージセンサの設定についてはセンサメーカーにお問い合わせください。

r_bspのハードウェアロック機能については、アプリケーションノート「RXファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)」の「2.17 ロック機能」を参照してください。

1.2 API の概要

表 1.1に PDC FIT モジュールに含まれる API 関数を示します。

表 1.1 API 関数一覧

関数	関数説明
R_PDC_Open	PDC FIT モジュールを初期化する関数です。
R_PDC_Close	PDC の動作を終了し、モジュールストップ状態にします。
R_PDC_Control	コントロールコードに対応した処理を行います。
R_PDC_GetFifoAddr	PDC の FIFO のアドレスを取得する関数です。
R_PDC_GetVersion	API のバージョンを返す関数です。

2. API 情報

本 FIT モジュールの API はルネサスの API の命名基準に従っています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

PDC
DTC
DMAC

2.2 ソフトウェアの要求

本 FIT モジュールは以下の FIT モジュールに依存しています。

Renesas Board Support Package (r_bsp) Rev.5.20 以上

2.3 サポートされているツールチェーン

本 FIT モジュールは「6.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.4 使用する割り込みベクタ

R_PDC_Open 関数を実行すると、引数の値に対応した PCDFI 割り込み、PCFEI 割り込み、PCERI 割り込みが有効になります。表 2.1 に本 FIT モジュールが使用する割り込みベクタを示します。

表 2.1 使用する割り込みベクター一覧

デバイス	割り込みベクタ
RX64M	PCDFI 割り込み (ベクタ番号: 97)
RX65N	GROUPBL0 割り込み (ベクタ番号: 110)
RX66N	● PCFEI 割り込み (グループ割り込み要因番号: 30)
RX71M	● PCERI 割り込み (グループ割り込み要因番号: 31)
RX72M	
RX72N	

2.5 ヘッダファイル

すべての API 呼び出しと使用されるインタフェース定義は r_pdc_rx_if.h に記載しています。

2.6 整数型

このプロジェクトは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.7 コンパイル時の設定

本 FIT モジュールのコンフィギュレーションオプションの設定は、`r_pdc_rx_config.h` で行います。オプション名および設定値に関する説明を下表に示します。

Configuration options in <code>r_pdc_rx_config.h</code>	
PDC_CFG_PCKO_DIV 【注】 デフォルト値は“2”	指定した分周比に応じて PDC 制御レジスタ 0 (PCCR0) の PCKO 分周比選択ビットを設定します。パラレルデータ転送クロック出力 (PCKO) の動作周波数はクロックソースの周辺モジュールクロック B (PCLKB) をこの設定で分周した値となります。設定値は 2, 4, 6, 8, 10, 12, 14, 16 の中から指定してください。その他の値を指定した場合は、ビルド時にエラーになります。 【注】 動作周波数の範囲は 1~30MHz ですが、ご利用のイメージセンサ (カメラモジュール) の仕様に合わせて動作可能な最適値を指定してください。

2.8 コードサイズ

ツールチェーン (セクション2.3記載) でのコードサイズは、最適化レベル 2、およびコードサイズ重視の最適化を前提としたサイズです。ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、本モジュールのコンフィギュレーションヘッダファイルで設定される、ビルド時のコンフィギュレーションオプションによって決まります。

下表の値は下記条件で確認しています。

モジュールリビジョン: `r_pdc_rx` rev2.06

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00

(統合開発環境のデフォルト設定に”-lang = c99”オプションを追加)

GCC for Renesas RX 8.03.00.202102

(統合開発環境のデフォルト設定に”-std=gnu99”オプションを追加)

IAR C/C++ Compiler for Renesas RX version 4.20.1

(統合開発環境のデフォルト設定)

コンフィギュレーションオプション: デフォルト設定

ROM、RAM およびスタックのコードサイズ				
デバイス	分類	使用メモリ		
		Renesas Compiler	GCC	IAR Compiler
RX72N	ROM	2069 バイト	3976 バイト	3236 バイト
	RAM	17 バイト	20 バイト	17 バイト
	スタック (注 1)	152 バイト	-	204 バイト

注 1. 割り込み関数の最大使用スタックサイズを含みます。

2.9 引数

API 関数の引数である構造体、列挙体を示します。これらは API 関数のプロトタイプ宣言とともに `r_pdc_rx_if.h` に記載されています。

```
/* 割り込み優先レベルの制御 */
```

```
typedef struct st_pdc_int_priority_data_cfg
```

```
{  
    uint8_t  pcdfi_level;          /* PCDFI 割り込み優先レベル */  
    uint8_t  groupbl0_level;       /* GROUPBL0 割り込み優先レベル */  
} pdc_ipr_dcfg_t;
```

```
/* 割り込みコントローラ (ICUA) の PDC 割り込みの許可/禁止 */
```

```
typedef struct st_pdc_inticu_data_cfg
```

```
{  
    bool  pcfei_iен;              /* フレームエンド割り込み要求許可 */  
    bool  pceri_iен;              /* エラー割り込み要求許可 */  
    bool  pcdfi_iен;              /* 受信データレディ割り込み要求許可 */  
} pdc_inticu_dcfg_t;
```

```
/* PDC 割り込みの許可/禁止 */
```

```
typedef struct st_pdc_intpdc_data_cfg
```

```
{  
    bool  dfie_iен;              /* 受信データレディ割り込み要求許可 */  
    bool  feie_iен;              /* フレームエンド割り込み要求許可 */  
    bool  ovie_iен;              /* オーバラン割り込み要求許可 */  
    bool  udrie_iен;             /* アンダラン割り込み要求許可 */  
    bool  verie_iен;             /* 垂直方向ライン数設定エラー割り込み要求許可 */  
    bool  herie_iен;             /* 水平方向バイト数設定エラー割り込み要求許可 */  
} pdc_intpdc_dcfg_t;
```

```
/* キャプチャ位置の指定 */
```

```
typedef struct st_pdc_position_data_cfg
```

```
{  
    uint16_t vst_position;        /* 垂直方向キャプチャ開始ライン位置 */  
    uint16_t hst_position;        /* 水平方向キャプチャ開始バイト位置 */  
} pdc_pos_dcfg_t;
```

```
/* キャプチャサイズの指定 */
```

```
typedef struct st_pdc_size_data_cfg
```

```
{  
    uint16_t vsz_size;            /* 垂直方向キャプチャサイズ */  
    uint16_t hsz_size;            /* 水平方向キャプチャサイズ */  
} pdc_size_dcfg_t;
```

```
/* PDC の設定*/
typedef struct st_pdc_data_cfg
{
    uint16_t      iupdc_select;    /* 割り込み設定更新選択 */
    pdc_ipr_dcfg_t priority;        /* 割り込み優先レベル */
    pdc_inticu_dcfg_t inticu_req;   /* ICU 割り込み設定 */
    pdc_intpdc_dcfg_t intpdc_req;   /* PDC 割り込み設定 */
    bool          vps_select;      /* VSYNC 信号極性選択 */
    bool          hps_select;      /* HSYNC 信号極性選択 */
    pdc_pos_dcfg_t capture_pos;     /* キャプチャ位置設定 */
    pdc_size_dcfg_t capture_size;   /* キャプチャサイズ設定 */
    pdc_cb_t       p_callback;      /* コールバック関数へのポインタ */
} pdc_data_cfg_t;
```

```
/* PDC ステータスレジスタ（PCSR）のコピー */
typedef struct st_pdc_data_cfg
{
    bool    frame_busy;    /* PDC の動作状態（FBSY フラグ） */
    bool    fifo_empty;    /* FIFO の状態（FEMPF フラグ） */
    bool    frame_end;     /* フレームエンド（FEF フラグ） */
    bool    overrun;       /* オーバラン（OVRF フラグ） */
    bool    underrun;       /* アンダラン（UDRF フラグ） */
    bool    verf_error;     /* 垂直方向ライン数設定エラー（VERF フラグ） */
    bool    herf_error;     /* 水平方向バイト数設定エラー（HERF フラグ） */
} pdc_pcsr_stat_t;
```

```
/* PDC 端子モニタステータスレジスタ（PCMONR）のコピー*/
typedef struct st_pdc_data_cfg
{
    bool    vsync;         /* VSYNC 信号ステータス（VSYNC フラグ） */
    bool    hsync;         /* HSYNC 信号ステータス（HSYNC フラグ） */
} pdc_pcmnr_stat_t;
```

```
/* PDC ステータス*/
typedef struct st_pdc_data_cfg
{
    pdc_pcsr_stat_t    pcsr_stat;    /* PDC ステータスレジスタ（PCSR）情報 */
    pdc_pcmnr_stat_t   pcmnr_stat;   /* PDC 端子モニタステータス（PCMONR）情報 */
} pdc_stat_t;
```

```
/* R_PDC_Control のコントロールコード */
typedef enum e_pdc_command
{
    PDC_CMD_CAPTURE_START = 0,        /* PDC のキャプチャを開始する */
    PDC_CMD_CHANGE_POS_AND_SIZE,     /* PDC キャプチャ位置&キャプチャサイズ変更 */
    PDC_CMD_STATUS_GET,               /* PDC ステータス取得 */
    PDC_CMD_STATUS_CLR,               /* PDC ステータスクリア */
    PDC_CMD_SET_INTERRUPT,            /* PDC 割り込み設定 */
    PDC_CMD_DISABLE,                  /* PDC 受信動作禁止 */
    PDC_CMD_ENABLE,                   /* PDC 受信動作許可 */
    PDC_CMD_RESET                     /* PDC リセット */
} pdc_command_t;
```



```
/* コールバック関数へのポインタ */
typedef struct
{
    void (*pcb_receive_data_ready)(void *); /* 受信データレディ割り込み発生時の
                                              コールバック関数のポインタ */
    void (*pcb_frame_end)(void *); /* フレームエンド割り込み発生後に PDC の FIFO が
                                    エンプティになった場合のコールバック関数のポインタ */
    void (*pcb_error)(void *); /* オーバランエラー、アンダランエラー、垂直方向ライン数設定エラー、
                                水平方向バイト数設定エラー発生時のコールバック関数のポインタ */
}pdc_cb_t;
```

```
/* コールバック関数呼び出し要因のイベントコード */
typedef enum
{
    PDC_EVT_ID_DATAREADY = 0, /* 受信データレディ割り込みが発生 */
    PDC_EVT_ID_FRAMEEND, /* フレームエンド割り込みが発生 */
    PDC_EVT_ID_TIMEOUT, /* 待機時間経過しても FIFO がエンプティにならなかった */
    PDC_EVT_ID_ERROR, /* エラー割り込みが発生 */
    PDC_EVT_ID_OVERRUN, /* オーバラン割り込みが発生 */
    PDC_EVT_ID_UNDERRUN, /* アンダラン割り込みが発生 */
    PDC_EVT_ID_VERTICALLINE, /* 垂直方向ライン数設定エラー割り込みが発生 */
    PDC_EVT_ID_HORIZONTALBYTE /* 水平方向バイト数設定エラー割り込みが発生 */
}pdc_cb_event_t;
```

```
/* コールバック関数へ渡される引数 */
typedef struct
{
    pdc_cb_event_t event_id; /* コールバック関数呼び出し要因のイベントコード */
}pdc_cb_arg_t;
```

2.10 戻り値

API 関数の戻り値を示します。この列挙型は、API 関数のプロトタイプ宣言とともに `r_pdc_rx_if.h` に記載されています。

```
/* 関数の戻り値 */
typedef enum e_pdc_return
{
    PDC_SUCCESS = 0,
    PDC_ERR_OPENED,

    PDC_ERR_NOT_OPEN,
    PDC_ERR_INVALID_ARG,
    PDC_ERR_INVALID_COMMAND,
    PDC_ERR_NULL_PTR,
    PDC_ERR_LOCK_FUNC,
    PDC_ERR_INTERNAL ,
    PDC_ERR_RST_TIMEOUT
    PDC_ERR_ONGOING
} pdc_return_t;

/* PDC API のエラーコード */

/* 問題なく処理が完了した */
/* PDC モジュールは初期化の状態。初期化関数の R_PDC_Open が
   実行済み。 */
/* PDC モジュールは未初期化の状態。R_PDC_Open が未実行。 */
/* 無効な引数が入力された */
/* コマンドが無効 */
/* 引数のポインタ値が NULL だった */
/* PDC のリソースが他のプロセスで使われている。 */
/* モジュールの内部エラーを検出した。 */
/* 一定時間経過しても PDC のリセットが解除されなかった場合 */
/* PDC が受信動作中の場合 */
```

2.11 コールバック関数

(1) 受信データレディ割り込み (PCDFI)、フレームエンド割り込み (PCFEI) のコールバック関数

PDC FIT モジュールでは、受信データレディ割り込み (PCDFI) が発生したとき、フレームエンド割り込み (PCFEI) が発生して FIFO がエンプティになったときにコールバック関数を呼び出します。

コールバック関数の設定は、R_PDC_Open 関数を用いて設定します。詳細は「R_PDC_Open()」を参照してください。

PDC FIT モジュールは受信データレディ割り込みが発生した場合、受信データレディ割り込みのコールバック関数を呼び出します。ただし、データ転送に DMAC を選択した場合は PCDFI の割り込み優先レベルを'0'に設定してコールバック関数が呼び出されないようにしてください。

PDC FIT モジュールはフレームエンド割り込みが発生した場合、DTC/DMAC が PDC の FIFO に格納されたデータの転送が完了するまで (PDC の FIFO がエンプティになるまで) 待機します。PDC の FIFO がエンプティであることを確認した場合、PDC を動作停止に設定して、フレームエンドフラグを'0'クリアしてから、PDC FIT モジュールはフレームエンド割り込みのコールバック関数を呼び出します。ただし、PDC の FIFO がエンプティになる前にアンダランが発生した場合は、フレームエンドフラグを'0'クリアしてからエラー発生時のコールバック関数を呼び出します。また一定時間経過しても PDC の FIFO がエンプティにならない場合はフレームエンドフラグを'0'クリアしてからタイムアウトのコールバック関数を呼び出します。

コールバック関数が呼び出されるとき、表 2.2 に示す定数を格納した変数を引数として渡します。引数の値をコールバック関数外で使用する場合は、グローバル変数などの変数にコピーしてください。

上記のタイミングでコールバック関数を呼び出す場合は、グループ割り込み (GROUPBL0) の割り込み要求を許可にしたうえ、R_PDC_Open 関数実行時に渡される引数で PCDFI 割り込み要求および PCFEI 割り込み要求、受信データレディ割り込み要求およびフレームエンド割り込み要求を許可に設定してください。詳細は「0 R_PDC_Open()」を参照してください。

表 2.2 受信データレディ割り込み、フレームエンド割り込み発生時のコールバック関数の引数一覧

変数定義	意味
PDC_EVT_ID_DATAREADY	受信データレディ割り込みが発生した
PDC_EVT_ID_FRAMEEND	フレームエンド割り込みが発生した
PDC_EVT_ID_TIMEOUT	待機時間を経過しても FIFO がエンプティにならなかった。

(2) エラー発生時のコールバック関数

PDC FIT モジュールでは、オーバラン、アンダラン、垂直方向ライン数設定エラー、水平方向バイト数設定エラーが発生したときコールバック関数を呼び出します。

コールバック関数の設定は、R_PDC_Open 関数を用いて設定します。詳細は「R_PDC_Open()」を参照してください。

PDC FIT モジュールはエラー割り込みが発生した場合、PDC の動作を停止させた後に“PDC_EVT_ID_ERROR”を引数とするコールバック関数を呼び出します。その後オーバラン、アンダラン、垂直方向ライン数設定エラー、水平方向バイト数設定エラーの順にエラー発生を確認します。エラーが発生していた場合はコールバック関数を呼び出します。コールバック関数の処理が完了すると発生していたエラーフラグを'0'クリアしてから、次のエラー発生を確認します。

“PDC_EVT_ID_ERROR”を引数とするコールバック関数が呼び出された場合は、処理の先頭で DTC/DMAC の転送処理を禁止に設定してください。

コールバック関数が呼び出されるとき、表 2.3 に示す定数を格納した変数を引数として渡します。引数の値をコールバック関数外で使用する場合は、グローバル変数などの変数にコピーしてください。

上記のタイミングでコールバック関数を呼び出す場合は、グループ割り込み (GROUPBL0) の割り込み要求を許可にしたうえ、R_PDC_Open 関数実行時に渡される引数で PCERI 割り込み要求およびオーバラン割り込み要求、アンダラン割り込み要求、垂直方向ライン数設定エラー割り込み要求、水平方向バイト数設定エラー割り込み要求を許可に設定してください。詳細は「R_PDC_Open()」を参照してください。

表 2.3 エラー発生時のコールバック関数の引数一覧

変数定義	意味
PDC_EVT_ID_ERROR	エラー割り込みが発生した
PDC_EVT_ID_OVERRUN	オーバランエラーが発生した
PDC_EVT_ID_UNDERRUN	アンダランエラーが発生した
PDC_EVT_ID_VERTICALLINE	垂直方向ライン数設定エラーが発生した
PDC_EVT_ID_HORIZONTALBYTE	水平方向バイト数設定エラーが発生した

2.12 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。ただし、Smart Configurator は、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

- (1) e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。
- (2) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。
- (3) CS+上で Smart Configurator を使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。
- (4) CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

2.13 for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理等で for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合は、「WAIT_LOOP」で該当の処理を検索できます。

以下に記述例を示します。

while 文の例：

```
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}
```

for 文の例：

```
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}
```

do while 文の例：

```
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API 関数

R_PDC_Open()

この関数は PDC FIT モジュールを初期化する関数です。この関数は他の API 関数を使用する前に実行される必要があります。

Format

```

pdc_return_t R_PDC_Open (
    pdc_data_cfg_t *p_data_cfg
)

```

Parameters

*p_data_cfg
PDC 設定データ構造体のポインタ

参照する pdc_data_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、API 呼び出し時に設定する必要はありません。

構造体メンバ	概略	設定値	設定対象となるレジスタ	設定内容
priority.pcdfi_level	PCDFI 割り込み優先レベル	8bit data 00h to 0Fh	ICU.IPR097.IPR	受信データレディ割り込み（PCDFI）優先レベル設定
priority.groupbl0_level	GROUPBL0 割り込み優先レベル	8bit data 00h to 0Fh	ICU.IPR110.IPR	フレームエンド割り込みおよびエラー割り込み優先レベル設定
inticu_req.pcdfi_ien	PCDFI 割り込み許可	false	ICU.IER0C.IEN1	受信データレディ割り込み（PCDFI）の割り込み要求を禁止
		true		受信データレディ割り込み（PCDFI）の割り込み要求を許可
inticu_req.pcfei_ien	PCFEI 割り込み許可	false	ICU.GRPBL0.EN30	フレームエンド割り込み（PCFEI）の割り込み要求を禁止
		true		フレームエンド割り込み（PCFEI）の割り込み要求を許可
inticu_req.pceri_ien	PCERI 割り込み許可	false	ICU.GRPBL0.EN31	エラー割り込み（PCERI）の割り込み要求を禁止
		true		エラー割り込み（PCERI）の割り込み要求を許可
intpdc_req.dfie_ien	受信データレディ割り込み要求	false	PCCR0.DFIE	受信データレディ割り込み要求の発生を禁止
		true		受信データレディ割り込み要求の発生を許可
intpdc_req.feie_ien	フレームエンド割り込み要求	false	PCCR0.FEIE	フレームエンド割り込み要求の発生を禁止
		true		フレームエンド割り込み要求の発生を許可
intpdc_req.ovie_ien	オーバラン割り込み要求	false	PCCR0.OVIE	オーバラン割り込み要求の発生を禁止
		true		オーバラン割り込み要求の発生を許可
intpdc_req.udrie_ien	アンダラン割り込み要求	false	PCCR0.UDRIE	アンダラン割り込み要求の発生を禁止
		true		アンダラン割り込み要求の発生を許可

intpdc_req.verie_ien	垂直方向ライン数設定エラー割り込み要求.	false	PCCR0.VERIE	垂直方向ライン数設定エラー割り込み要求の発生を禁止
		true		垂直方向ライン数設定エラー割り込み要求の発生を許可
intpdc_req.herie_ien	水平方向バイト数設定エラー割り込み要求	false	PCCR0.HERIE	水平方向バイト数設定エラー割り込み要求の発生を禁止
		true		水平方向バイト数設定エラー割り込み要求の発生を許可
vps_select	VSYNC 信号極性選択	PDC_VSYNC_SIGNAL_POLARITY_HIGH	PCCR0.VPS	VSYNC 信号は High アクティブ
		PDC_VSYNC_SIGNAL_POLARITY_LOW		VSYNC 信号は Low アクティブ
hps_select	HSYNC 信号極性選択	PDC_HSYNC_SIGNAL_POLARITY_HIGH	PCCR0.HPS	HSYNC 信号は High アクティブ
		PDC_HSYNC_SIGNAL_POLARITY_LOW		HSYNC 信号は Low アクティブ
capture_pos.vst_position	垂直方向キャプチャ開始ライン位置	12bit data 0000h to 0FFh	VCR.VST	垂直方向のキャプチャ開始ライン位置
capture_pos.hst_position	水平方向キャプチャ開始バイト位置	12bit data 0000h to 0FFh	HCR.HST	水平方向のキャプチャ開始バイト位置
capture_size.vsz_size	垂直方向キャプチャサイズ	12bit data 0001h to 0FFh	VCR.VSZ	垂直方向のキャプチャライン数
capture_size.hsz_size	水平方向キャプチャサイズ	12bit data 0004h to 0FFh	HCR.HSZ	水平方向のキャプチャバイト数
p_callback.pcb_receive_data_ready	PCDFI 割り込み発生時のコールバック関数へのポインタ	NULL / FIT_NO_FUNC 以外	なし	受信データレディ割り込み発生時にポインタが示すアドレスのコールバック関数を実行します
		NULL / FIT_NO_FUNC		要因が発生してもコールバック関数は実行されません
p_callback.pcb_frame_end	PCFEI 割り込み発生時のコールバック関数へのポインタ	NULL / FIT_NO_FUNC 以外	なし	フレームエンド割り込み発生後にFIFOがエンプティになったときにポインタが示すアドレスのコールバック関数を実行します
		NULL / FIT_NO_FUNC		要因が発生してもコールバック関数は実行されません

p_callback.pcb_error	PCERI 割り込み発生時のコールバック関数へのポインタ	NULL / FIT_NO_F UNC 以外	なし	エラー割り込み発生時およびオーバーラン、アンダラン、垂直方向ライン数設定エラー、水平方向バイト数設定エラー発生時にポインタが示すアドレスのコールバック関数を実行します
		NULL / FIT_NO_F UNC		要因が発生してもコールバック関数は実行されません

Return Values

PDC_SUCCESS /* 問題なく処理が完了した場合 */
PDC_ERR_OPENED /* R_PDC_Open が既の実行されている場合 */
PDC_ERR_INVALID_ARG /* PDC 設定情報のパラメータの値が不正な場合 */
PDC_ERR_NULL_PTR /* 引数 p_data_cfg が NULL ポインタの場合 */
PDC_ERR_LOCK_FUNC /* PDC が既に他のプロセスにロックされている場合 */
PDC_ERR_INTERNAL /* モジュールの内部エラーが検出された場合 */
PDC_ERR_RST_TIMEOUT /* 一定時間経過しても PDC のリセットが解除されなかった場合 */

Properties

r_pdc_rx_if.h にプロトタイプ宣言されています。

Description

PDC を使用するための初期設定として、以下の処理を行います。

r_bsp のハードウェアロック機能を使用した PDC のハードウェアリソースロック

PDC のモジュールストップ解除

PDC で使用する割り込み発生時にコールバックされる関数の登録

PDC で使用する割り込み設定

受信データレディ割り込み (PCDFI)、フレームエンド割り込み (PCFEI)、エラー割り込み (PCERI) の割り込み設定を行います。

PDC 受信動作停止

PDC 制御レジスタ 1 (PCCR1) PCE を“受信動作を禁止”に設定します。

パラレルデータ転送クロック出力 (PCKO) のクロック設定

PDC 制御レジスタ 0 (PCCR0) PCKDIV を設定します。

パラレルデータ転送クロック出力 (PCKO) の設定は、r_pdc_rx_config.h の PDC_CFG_PCKO_DIV に応じた設定値を設定します。

パラレルデータ転送クロック出力 (PCKO) のクロック供給開始

PDC 制御レジスタ 0 (PCCR0) PCKOE を“PCKO 出力を許可”に設定します。

PIXCLK の入力許可 (PCCR0.PCKE)

PDC 制御レジスタ 0 (PCCR0) PCKE を“PIXCLK 入力を許可”に設定します。

PDC リセット (PCCR0.PRST)

PDC の内部状態および PDC リセット対象レジスタの初期化を開始します。

垂直及び水平方向のキャプチャ範囲設定 (VCR、HCR の設定)

VSYN、HSYN 信号の極性設定 (VPS、HPS)

割り込みの許可/禁止設定 (DFIE、FEIE、OVIE、UDRIE、VERIE、HERIE)

エンディアン設定 (EDS)

Example

サンプルコードの例ではイメージセンサの出力が1ドット当たり2バイトのため、水平方向の取り込み位置とサイズに水平ドット数を2倍した値を設定しています。ご使用のイメージセンサの出力に合わせて値を見直してください。

Case 1: VGA (640x480) サイズで画像を取得する場合の設定

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Setting values of PDC operation */
pdc_data_cfg_t data_pdc;

/*
Set the value 0 to PCDFI interrupt priority when using DMAC
Set the value 1-15 to PCDFI interrupt priority level when using DTC
*/
data_pdc.priority.pcdfi_level = 0;
/* Set the values 1-15 to GROUPBL0 interrupt priority level */
data_pdc.priority.groupbl0_level = 2;
/* PCDFI interrupt request in ICU is enabled */
data_pdc.inticu_req.pcdfi_iен = true;
/* PCFEI interrupt request in ICU is enabled */
data_pdc.inticu_req.pcfei_iен = true;
/* PCERI interrupt request in ICU is enabled */
data_pdc.inticu_req.pceri_iен = true;
/* Generation of receive data ready interrupt requests is enabled */
data_pdc.intpdc_req.dfie_iен = true;
/* Generation of frame end interrupt requests is enabled */
data_pdc.intpdc_req.feie_iен = true;
/* Generation of overrun interrupt requests is enabled */
data_pdc.intpdc_req.ovie_iен = true;
/* Generation of underrun interrupt requests is enabled */
data_pdc.intpdc_req.udrie_iен = true;
/* Generation of vertical line number setting error interrupt requests is enabled */
data_pdc.intpdc_req.verie_iен = true;
/* Generation of horizontal byte number setting error interrupt requests is enabled */
data_pdc.intpdc_req.herie_iен = true;
/* VSYNC signal is active LOW */
data_pdc.vps_select = PDC_VSYNC_SIGNAL_POLARITY_LOW;
/* HSYNC signal is active HIGH */
data_pdc.hps_select = PDC_HSYNC_SIGNAL_POLARITY_HIGH;
/* Capture from 0 pixel of vertical direction */
data_pdc.capture_pos.vst_position = 0;
/* Capture from 0 pixel of horizontal direction */
data_pdc.capture_pos.hst_position = 0;
/* Capture 480 pixels in vertical direction */
data_pdc.capture_size.vsz_size = 480;
/* Capture 640 pixels in horizontal direction */
data_pdc.capture_size.hsz_size = (640 * 2);
/* Pointer to PCDFI interrupt callback function */
data_pdc.p_callback.pcb_receive_data_ready = (void (*)(void *)) pcdfi_callback;
/* Pointer to PCFEI interrupt callback function */
data_pdc.p_callback.pcb_frame_end = (void (*)(void *)) pcfei_callback;
/* Pointer to PCERI interrupt callback function */
data_pdc.p_callback.pcb_error = (void (*)(void *)) pceri_callback;

ret_pdc = R_PDC_Open(&data_pdc);
if (PDC_SUCCESS != ret_pdc)
{
/* Error processing */
}
```

Case 2: VGA (640x480) の画像に対して QVGA (320x240) で中心から右下を取得する場合の設定

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t    ret_pdc;
/* Setting values of PDC operation */
pdc_data_cfg_t          data_pdc;

/*
  Set the value 0 to PCDFI interrupt priority when using DMAC
  Set the value 1-15 to PCDFI interrupt priority level when using DTC
*/
data_pdc.priority.pcdfi_level = 0;
/* Set the values 1-15 to GROUPBL0 interrupt priority level */
data_pdc.priority.groupbl0_level = 2;
/* PCDFI interrupt request in ICU is enabled */
data_pdc.inticu_req.pcdfi_iен = true;
/* PCFEI interrupt request in ICU is enabled */
data_pdc.inticu_req.pcfei_iен = true;
/* PCERI interrupt request in ICU is enabled */
data_pdc.inticu_req.pceri_iен = true;
/* Generation of receive data ready interrupt requests is enabled */
data_pdc.intpdc_req.dfie_iен = true;
/* Generation of frame end interrupt requests is enabled */
data_pdc.intpdc_req.feie_iен = true;
/* Generation of overrun interrupt requests is enabled */
data_pdc.intpdc_req.ovie_iен = true;
/* Generation of underrun interrupt requests is enabled */
data_pdc.intpdc_req.udrie_iен = true;
/* Generation of vertical line number setting error interrupt requests is enabled */
data_pdc.intpdc_req.verie_iен = true;
/* Generation of horizontal byte number setting error interrupt requests is enabled */
data_pdc.intpdc_req.herie_iен = true;
/* VSYNC signal is active LOW */
data_pdc.vps_select = PDC_VSYNC_SIGNAL_POLARITY_LOW;
/* HSYNC signal is active HIGH */
data_pdc.hps_select = PDC_HSYNC_SIGNAL_POLARITY_HIGH;
/* Capture from 240 pixel of vertical direction */
data_pdc.capture_pos.vst_position = 240;
/* Capture from 320 pixel of horizontal direction */
data_pdc.capture_pos.hst_position = (320 * 2);
/* Capture 240 pixels in vertical direction */
data_pdc.capture_size.vsz_size = 240;
/* Capture 320 pixels in horizontal direction */
data_pdc.capture_size.hsz_size = (320 * 2);
/* Pointer to PCDFI interrupt callback function */
data_pdc.p_callback.pcb_receive_data_ready = (void (*)(void *)) pcdfi_callback;
/* Pointer to PCFEI interrupt callback function */
data_pdc.p_callback.pcb_frame_end = (void (*)(void *)) pcfei_callback;
/* Pointer to PCERI interrupt callback function */
data_pdc.p_callback.pcb_error = (void (*)(void *)) pceri_callback;

ret_pdc = R_PDC_Open(&data_pdc);
if (PDC_SUCCESS != ret_pdc)
{
  /* Error processing */
}
```

受信データレディ割り込みが発生したときにコールバックされる関数

```
#include "platform.h"
#include "r_pdc_rx_if.h"

void pcdfi_callback(void * pdata)
{
    /* Stores the argument for callback function */
    pdc_cb_arg_t * pdecode;
    pdecode = (pdc_cb_arg_t *)pdata;

    switch(pdecode->event_id)
    {
        case PDC_EVT_ID_DATAREADY:
            /* do something */
            break;

        default:
            break;
    }
}
```

フレームエンド割り込みが発生して PDC の FIFO がエンプティになったときにコールバックされる関数

```
#include "platform.h"
#include "r_pdc_rx_if.h"

void pcfei_callback(void * pdata)
{
    /* Stores the argument for callback function */
    pdc_cb_arg_t * pdecode;
    pdecode = (pdc_cb_arg_t *)pdata;

    switch(pdecode->event_id)
    {
        case PDC_EVT_ID_FRAMEEND:
            /* do something */
            break;

        case PDC_EVT_ID_TIMEOUT:
            /* do something */
            break;

        default:
            break;
    }
}
```

エラー割り込み、オーバランエラー、アンダランエラー、垂直方向ライン数設定エラー、水平方向バイト数設定エラーが発生したときにコールバックされる関数

```
#include "platform.h"
#include "r_pdc_rx_if.h"

void pceri_callback(void *pdata)
{
    /* Stores the argument for callback function */
    pdc_cb_arg_t *pdecode;
    pdecode = (pdc_cb_arg_t *)pdata;

    switch(pdecode->event_id)
    {
        case PDC_EVT_ID_ERROR:
            /* Disable the DTC or DMAC transfer */
            /* Error interrupt processing */
            break;

        case PDC_EVT_ID_OVERRUN:
            /* Overrun error processing */
            break;

        case PDC_EVT_ID_UNDERRUN:
            /* Underrun error processing */
            break;

        case PDC_EVT_ID_VERTICALLINE:
            /* Vertical Line Number Setting Error processing */
            break;

        case PDC_EVT_ID_HORIZONTALBYTE:
            /* Horizontal Byte Number Setting Error processing */
            break;

        default:
            break;
    }
}
```

Special Notes:

本 API はデバイスとカメラモジュールを接続した状態で実行してください。本 API を実行すると PIXCLK の入力許可後に PDC をリセットしますが、カメラモジュールが出力する PIXCLK がデバイスに入力されていない状態ではリセットが完了しないためです。戻り値が PDC_ERR_RST_TIMEOUT であることを確認した場合は、使用するカメラモジュールの設定およびハードウェアの構成を確認してください。

本 API 内で PDC のエンディアンを設定します。エンディアンはコンパイラの設定に合わせて選択されるようになっています。コンパイラのエンディアン設定がリトル・エンディアンであれば PDC のエンディアン設定もリトル・エンディアンとなり、コンパイラのエンディアン設定がビッグ・エンディアンであれば、PDC のエンディアン設定もビッグ・エンディアンとなります。

登録するコールバック関数は引数、戻り値のどちらも void 型にしてください。

R_PDC_Close()

PDC の動作を終了し、モジュールストップ状態にします。

Format

pdcc_return_t R_PDC_Close (void)

Parameters

なし

Return Values

PDC_SUCCESS	<i>/* 問題なく処理が完了した場合 */</i>
PDC_ERR_NOT_OPEN	<i>/* R_PDC_Open が実行されていない場合 */</i>
PDC_ERR_ONGOING	<i>/* PDC が受信動作中の場合 */</i>

Properties

r_pdc_rx_if.h にプロトタイプ宣言されています。

Description

PDC を終了するための処理を行います。

PDC で使用する割り込み（PCFEI、PCERI、PCDFI）を禁止にします。

PDC の動作禁止

PDC 制御レジスタ 1（PCCR1）PCE を“受信動作を禁止”に設定します。

パラレルデータ転送クロック出力（PCKO）のクロック供給停止

PDC 制御レジスタ 0（PCCR0）PCKOE を“PCKO 出力を禁止”に設定します。

イメージセンサからのピクセルクロックの入力禁止

PDC 制御レジスタ 0（PCCR0）PCKE を“PIXCLK 入力を禁止”に設定します。

PDC のモジュールストップ

r_bsp のハードウェアロック機能を使用した PDC のハードウェアリソースロックの解除

Example

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdcc_return_t ret_pdc;

ret_pdc = R_PDC_Close();

if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}
```

Special Notes:

本 API は、R_PDC_Open を実行して戻り値が PDC_SUCCESS であることを確認してから実行してください。

また、フレームエンド発生後やエラー検出後など、受信動作が停止していることを確認してから実行してください。

R_PDC_Control()

コントロールコードに対応した処理を行う関数です。

Format

```
pdrc_return_t R_PDC_Control (  
    pdrc_command_t    command,  
    pdrc_data_cfg_t   *p_data_cfg,  
    pdrc_stat_t       *p_stat  
)
```

Parameters

command

コントロールコード

*p_data_cfg

PDC 設定データ構造体へのポインタ

*p_stat

PDC ステータス構造体へのポインタ

The Command Values:

/ イメージセンサ (カメラモジュール) からのデータキャプチャを開始する場合 */*

PDC_CMD_CAPTURE_START

/ イメージセンサ (カメラモジュール) からのデータキャプチャ範囲を変更する場合 */*

PDC_CMD_CHANGE_POS_AND_SIZE

/ PDC のステータス情報を取得する場合 */*

PDC_CMD_STATUS_GET

/ PDC のステータス情報をクリアする場合 */*

PDC_CMD_STATUS_CLR

/ PDC の割り込み設定を再設定する場合 */*

PDC_CMD_SET_INTERRUPT

/ PDC の受信動作を禁止する場合 */*

PDC_CMD_DISABLE

/ PDC の受信動作を許可する場合 */*

PDC_CMD_ENABLE

/ PDC をリセットする場合 */*

PDC_CMD_RESET

指定するコマンドに応じて参照する引数が異なります。

PDC_CMD_CAPTURE_START の場合

— 参照する pdrc_data_cfg_t 構造体メンバと設定値
なし

— 参照する pdrc_stat_t 構造体メンバと設定値
なし

PDC_CMD_CHANGE_POS_AND_SIZE の場合

— 参照する pdc_data_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、API 呼び出し時に設定する必要はありません。

構造体メンバ	概略	設定値	設定対象となるレジスタ	設定内容
vst_position	垂直方向キャプチャ開始ライン位置	12bit data 0000h to 0FFEh	VCR.VST	垂直方向のキャプチャ開始ライン位置
hst_position	水平方向キャプチャ開始バイト位置	12bit data 0000h to 0FFBh	HCR.HST	水平方向のキャプチャ開始バイト位置
vsz_size	垂直方向キャプチャサイズ	12bit data 0001h to 0FFFh	VCR.VSZ	垂直方向のキャプチャライン数
hsz_size	水平方向キャプチャサイズ	12bit data 0004h to 0FFFh	HCR.HSZ	水平方向のキャプチャバイト数

— 参照する pdc_stat_t 構造体メンバと設定値

なし

PDC_CMD_STATUS_GET の場合

— 参照する pdc_data_cfg_t 構造体メンバと設定値

なし

— 参照する pdc_stat_t 構造体メンバと設定値

構造体メンバ	概略	設定値	参照対象となるレジスタ	設定内容
pcsr_stat.frame_busy	フレームビジーフラグ	false	PCSR.FBSY	受信動作停止
		true		受信動作中
pcsr_stat.fifo_empty	FIFO エンプティフラグ	false	PCSR.FEMPF	FIFO はエンプティではない
		true		FIFO はエンプティ
pcsr_stat.frame_end	フレームエンドフラグ	false	PCSR.FEF	フレームエンドなし
		true		フレームエンド発生
pcsr_stat.overrun	オーバランフラグ	false	PCSR.OVRF	FIFO オーバランなし
		true		FIOF オーバラン発生
pcsr_stat.underrun	アンダランフラグ	false	PCSR.UDRF	アンダランなし
		true		アンダラン発生
pcsr_stat.verf_error	垂直方向ライン数設定エラーフラグ	false	PCSR.VERF	垂直方向ライン数設定エラーなし
		true		垂直方向ライン数設定エラー発生
pcsr_stat.herf_error	水平方向バイト数設定エラーフラグ	false	PCSR.HERF	水平方向バイト数設定エラーなし
		true		水平方向バイト数設定エラー発生
pcmonr_stat.vsync	VSYNC 信号ステータスフラグ	false	PCMONR.VSYNC	VSYNC 信号は"LOW"
		true		VSYNC 信号は"HIGH"
pcmonr_stat.hsync	HSYNC 信号ステータスフラグ	false	PCMONR.HSYNC	HSYNC 信号は"LOW"
		true		HSYNC 信号は"HIGH"

PDC_CMD_STATUS_CLR の場合

— 参照する pdc_data_cfg_t 構造体メンバと設定値
なし

— 参照する pdc_stat_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、API 呼び出し時に設定する必要はありません。

構造体メンバ	概略	設定値	設定対象となるレジスタ	設定内容
pcsr_stat.frame_end	フレームエンドフラグ	false	PCSR.FEF	なにもしない
		true		フレームエンドフラグをクリアする
pcsr_stat.overrun	オーバランフラグ	false	PCSR.OVRF	なにもしない
		true		オーバランフラグをクリアする
pcsr_stat.underrun	アンダランフラグ	false	PCSR.UDRF	なにもしない
		true		アンダランフラグをクリアする
pcsr_stat.verf_error	垂直方向ライン数設定エラーフラグ	false	PCSR.VERF	なにもしない
		true		垂直方向ライン数設定エラーフラグをクリアする
pcsr_stat.herf_error	水平方向バイト数設定エラーフラグ	false	PCSR.HERF	なにもしない
		true		水平方向バイト数設定エラーフラグをクリアする

PDC_CMD_SET_INTERRUPT の場合

— 参照する pdc_data_cfg_t 構造体メンバと設定値

以下に記載したパラメータ以外は参照しませんので、API 呼び出し時に設定する必要はありません。

構造体メンバ	概略	設定値	設定対象となるレジスタ	設定内容
iupd_select	更新対象選択	10bit data 0000h to 03FFh	なし	下記パラメータのどの割り込み設定を更新するかを選択します。 Bit 0 : PCDFI 割り込み優先レベル Bit 1 : GROUPBL0 割り込み優先レベル Bit 2 : PCDFI 割り込み許可 Bit 3 : PCFEI 割り込み許可 Bit 4 : PCERI 割り込み許可 Bit 5 : 受信データレディ割り込み要求 Bit 6 : フレームエンド割り込み要求 Bit 7 : オーバラン割り込み要求 Bit 8 : アンダラン割り込み要求 Bit 9 : 垂直方向ライン数設定エラー割り込み要求 Bit 10 : 水平方向バイト数設定エラー割り込み要求 Bit 11-15 : 使用しない “0”の場合、設定の更新をしません。 “1”の場合、設定を更新します。
priority.pcdfi_level	PCDFI 割り込み優先レベル	8bit data 00h to 0Fh	ICU.IPR097.IPR	受信データレディ割り込み（PCDFI）優先レベル設定 【注】 iupd_select の Bit0 を“1”に設定してください。

構造体メンバ	概略	設定値	設定対象となるレジスタ	設定内容
priority.groupbl0_level	GROUPBL0 割り込み優先レベル	8bit data 00h to 0Fh	ICU.IPR110.IPR	フレームエンド割り込みおよびエラー割り込み優先レベル設定 【注】 iupd_select の Bit1 を "1" に設定してください。 現在の設定値より小さい値への変更は無効です。
inticu_req.pcfei_ien	PCFEI 割り込み許可	false	ICU.GRPBL0.EN30	フレームエンド割り込み (PCFEI) の割り込み要求を禁止 【注】 iupd_select の Bit2 を "1" に設定してください。
		true		フレームエンド割り込み (PCFEI) の割り込み要求を許可 【注】 iupd_select の Bit2 を "1" に設定してください。
inticu_req.pceri_ien	PCERI 割り込み許可	false	ICU.GRPBL0.EN31	エラー割り込み (PCERI) の割り込み要求を禁止 【注】 iupd_select の Bit3 を "1" に設定してください。
		true		エラー割り込み (PCERI) の割り込み要求を許可 【注】 iupd_select の Bit3 を "1" に設定してください。
inticu_req.pcdfi_ien	PCDFI 割り込み許可	false	ICU.IER0C.IEN1	受信データレディ割り込み (PCDFI) の割り込み要求を禁止 【注】 iupd_select の Bit4 を "1" に設定してください。
		true		受信データレディ割り込み (PCDFI) の割り込み要求を許可 【注】 iupd_select の Bit4 を "1" に設定してください。
intpdc_req.dfie_ien	受信データレディ割り込み要求	false	PCCR0.DFIE	受信データレディ割り込み要求の発生を禁止 【注】 iupd_select の Bit5 を "1" に設定してください。
		true		受信データレディ割り込み要求の発生を許可 【注】 iupd_select の Bit5 を "1" に設定してください。
intpdc_req.feie_ien	フレームエンド割り込み要求	false	PCCR0.FEIE	フレームエンド割り込み要求の発生を禁止 【注】 iupd_select の Bit6 を "1" に設定してください。
		true		フレームエンド割り込み要求の発生を許可 【注】 iupd_select の Bit6 を "1" に設定してください。
intpdc_req.ovie_ien	オーバラン割り込み要求	false	PCCR0.OVIE	オーバラン割り込み要求の発生を禁止 【注】 iupd_select の Bit7 を "1" に設定してください。
		true		オーバラン割り込み要求の発生を許可。 【注】 iupd_select の Bit7 を "1" に設定してください。

構造体メンバ	概略	設定値	設定対象となるレジスタ	設定内容
intpdc_req. udrie_ien	アンダラン 割り込み要求	false	PCCR0.UDRIE	アンダラン割り込み要求の発生を禁止 【注】 iupd_select の Bit8 を"1"に設定してください。
		true		アンダラン割り込み要求の発生を許可 【注】 iupd_select の Bit8 を"1"に設定してください。
intpdc_req. verie_ien	垂直方向ライン 数設定エラー 割り込み要求	false	PCCR0.VERIE	垂直方向ライン数設定エラー割り込み要求の発生を禁止 【注】 iupd_select の Bit9 を"1"に設定してください。
		true		垂直方向ライン数設定エラー割り込み要求の発生を許可 【注】 iupd_select の Bit9 を"1"に設定してください。
intpdc_req. herie_ien	水平方向バイト 数設定エラー 割り込み要求	false	PCCR0.HERIE	水平方向バイト数設定エラー割り込み要求の発生を禁止 【注】 iupd_select の Bit10 を"1"に設定してください。
		true		水平方向バイト数設定エラー割り込み要求の発生を許可 【注】 iupd_select の Bit10 を"1"に設定してください。

— 参照する pdc_stat_t 構造体メンバと設定値
なし

PDC_CMD_DISABLE/PDC_CMD_ENABLE の場合

— 参照する pdc_data_cfg_t 構造体メンバと設定値
なし

— 参照する pdc_stat_t 構造体メンバと設定値
なし

PDC_CMD_RESET の場合

— 参照する pdc_data_cfg_t 構造体メンバと設定値
なし

— 参照する pdc_stat_t 構造体メンバと設定値
なし

Return Values

PDC_SUCCESS

/* 問題なく処理が完了した場合 */

PDC_ERR_NOT_OPEN

/* R_PDC_Open が実行されていない場合 */

PDC_ERR_INVALID_ARG

/* PDC レジスタへの設定値が不正な場合 */

PDC_ERR_INVALID_COMMAND

/* 引数のコマンドが不正な場合 */

PDC_ERR_NULL_PTR

/* 引数 p_data_cfg または p_stat が NULL ポインタの場合 */

PDC_ERR_RST_TIMEOUT

/* 一定時間経過しても PDC のリセットが解除されなかった場合 */

PDC_ERR_ONGOING

/* PDC が受信動作中の場合 */

Properties

r_pdc_rx_if.h にプロトタイプ宣言されています。

Description

<PDC_CMD_CAPTURE_START コマンド処理>

割り込みの再設定と PDC のリセットを行ってから PDC の受信動作を許可することでデータキャプチャを開始します。

<PDC_CMD_CHANGE_POS_AND_SIZE コマンド処理>

PDC の受信動作を禁止に設定してからキャプチャ開始位置及びキャプチャサイズの再設定を行います。

— 水平方向の取り込み位置とサイズは、ご使用のイメージセンサの出力に合わせて設定してください。

<PDC_CMD_STATUS_GET コマンド処理>

PDC のステータス情報を引数の p_stat が示すポインタ位置に書き込みます。

<PDC_CMD_STATUS_CLR コマンド処理>

引数の p_stat で指定した PDC のステータス情報をクリアします。

<PDC_CMD_SET_INTERRUPT コマンド>

PDC の受信動作を禁止に設定してから PDC の割り込みを再設定します。

<PDC_CMD_DISABLE コマンド>

PDC の受信動作を禁止します。

<PDC_CMD_ENABLE コマンド>

PDC の受信動作を許可します。

<PDC_CMD_RESET コマンド処理>

PDC の受信動作を禁止に設定してから PDC をリセットします。

Example

サンプルコードの例では、イメージセンサの出力が 1 ドット当たり 2 バイトのため、水平方向の取り込み位置とサイズに水平ドット数を 2 倍した値を設定しています。ご使用のイメージセンサの出力に合わせて値を見直してください。

Case 1: キャプチャを開始する場合

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Unused */
pdc_data_cfg_t dummy_data;
/* Unused */
pdc_stat_t dummy_stat;

ret_pdc = R_PDC_Control(PDC_CMD_CAPTURE_START, &dummy_data, &dummy_stat);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error Processing */
}
```

Case 2: キャプチャ位置とサイズを再設定する場合

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Setting values of PDC operation */
pdc_data_cfg_t data_pdc;
/* Unused */
pdc_stat_t dummy_stat;

/* Capture from 0 pixel of vertical direction */
data_pdc.capture_pos.vst_position = 0;
/* Capture from 0 pixel of horizontal direction */
data_pdc.capture_pos.hst_position = 0;
/* Capture 480 pixels in vertical direction */
data_pdc.capture_pos.vsz_size = 480;
/* Capture 640 pixels in horizontal direction */
data_pdc.capture_pos.hsz_size = (640 * 2);

ret_pdc = R_PDC_Control(PDC_CMD_CHANGE_POS_AND_SIZE, &data_pdc, &dummy_stat);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}
```

Case 3: ステータスを取得する場合

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Unused */
pdc_data_cfg_t dummy_data;
/* Status values of PDC operation */
pdc_stat_t stat_pdc;

ret_pdc = R_PDC_Control(PDC_CMD_STATUS_GET, &dummy_data, &stat_pdc);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}
```

Case 4: ステータスをクリアする場合

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Unused */
pdc_data_cfg_t dummy_data;
/* Status values of PDC operation */
pdc_stat_t stat_pdc;

/* Clear Frame Busy Flag */
stat_pdc.pcsr_stat.frame_busy = true;
/* Clear FIFO Empty Flag */
```

```

stat_pdc.pcsr_stat.fifo_empty = true;
/* Clear Frame End Flag */
stat_pdc.pcsr_stat.frame_end = true;
/* Clear Overrun Flag */
stat_pdc.pcsr_stat.overrun = true;
/* Clear Underrun Flag */
stat_pdc.pcsr_stat.underrun = true;
/* Clear Vertical Line Number Setting Error Flag */
stat_pdc.pcsr_stat.verf_error = true;
/* Clear Horizontal Byte Number Setting Error Flag */
stat_pdc.pcsr_stat.herf_error = true;

ret_pdc = R_PDC_Control(PDC_CMD_STATUS_CLR, &dummy_data, &stat_pdc);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}

```

Case 5: 割り込みの再設定を行う場合

```

#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Setting values of PDC operation */
pdc_data_cfg_t p_data_pdc;
/* Unused */
pdc_stat_t dummy_stat;

/* Update all of interrupt setting values with the contents of following */
data_pdc.iupd_select = PDC_ALL_INT_UPDATE;
/* PCDFI interrupt priority level is 8 */
data_pdc.priority.pcdfi_level = 8;
/* GROUPBL0 interrupt priority level is 2 */
data_pdc.priority.groupbl0_level = 2;
/* PCDFI interrupt request in ICU is enabled */
data_pdc.inticu_req.pcdfi_iен = true;
/* PCFEI interrupt request in ICU is enabled */
data_pdc.inticu_req.pcfei_iен = true;
/* PCERI interrupt request in ICU is enabled */
data_pdc.inticu_req.pceri_iен = true;
/* Generation of receive data ready interrupt requests is enabled */
data_pdc.intpdc_req.dfie_iен = true;
/* Generation of frame end interrupt requests is enabled */
data_pdc.intpdc_req.feie_iен = true;
/* Generation of overrun interrupt requests is enabled */
data_pdc.intpdc_req.ovie_iен = true;
/* Generation of underrun interrupt requests is enabled */
data_pdc.intpdc_req.udrie_iен = true;
/* Generation of vertical line number setting error interrupt requests is enabled */
data_pdc.intpdc_req.verie_iен = true;
/* Generation of horizontal byte number setting error interrupt requests is enabled */
data_pdc.intpdc_req.herie_iен = true;

ret_pdc = R_PDC_Control(PDC_CMD_SET_INTERRUPT, &data_pdc, &dummy_stat);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}

```

```
}
```

Case 6: PDC の受信動作のみを停止する場合

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Unused */
pdc_data_cfg_t dummy_data;
/* Unused */
pdc_stat_t dummy_stat;

ret_pdc = R_PDC_Control(PDC_CMD_DISABLE, &dummy_data, &dummy_stat);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}
```

Case 7: PDC の受信動作のみを許可する場合

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Unused */
pdc_data_cfg_t dummy_data;
/* Unused */
pdc_stat_t dummy_stat;

ret_pdc = R_PDC_Control(PDC_CMD_ENABLE, &dummy_data, &dummy_stat);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}
```

Case 8: PDC のリセットを行う場合

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Error code of PDC FIT API */
volatile pdc_return_t ret_pdc;
/* Unused */
pdc_data_cfg_t dummy_data;
/* Unused */
pdc_stat_t dummy_stat;

ret_pdc = R_PDC_Control(PDC_CMD_RESET, &dummy_data, &dummy_stat);
if (PDC_SUCCESS != ret_pdc)
{
    /* Error processing */
}
```


Special Notes:

受信動作中に本 API を実行した場合は PDC レジスタを書き換えるため、受信動作が停止します。フレームエンド割り込みの発生前に本 API を実行すると受信動作が停止するため、画像データの取り込みが途中で停止します。画像の取り込みを再開する場合は転送先のメモリのポインタを DMAC/DTC に再設定してから、R_PDC_Control のキャプチャ開始コマンドで画像データの再取得を行ってください。

コマンド“PDC_CMD_STATUS_CLR”を引数にして R_PDC_Control を実行する場合は、クリアするステータス情報を“true”に、クリアしないステータス情報を“false”に設定してください。設定しないで R_PDC_Control を実行すると、意図しないステータス情報がクリアされる場合があります。

コマンド“PDC_CMD_STATUS_GET”以外を引数にして R_PDC_Control を実行する場合、フレームエンド発生後やエラー検出後など、受信動作が停止していることを確認してから実行してください。

R_PDC_GetFifoAddr()

PDC の FIFO のアドレスを取得する関数です。

Format

```
    pdc_return_t R_PDC_GetFifoAddr (  
        uint32_t      *p_fifo_addr  
    )
```

Parameters

**p_fifo_addr*
PDC の FIFO のアドレスへのポインタ

Return Values

<i>PDC_SUCCESS</i>	<i>/* 問題なく処理が完了した場合 */</i>
<i>PDC_ERR_NOT_OPEN</i>	<i>/* R_PDC_Open が実行されていない場合 */</i>
<i>PDC_ERR_NULL_PTR</i>	<i>/* 引数 p_fifo_addr が NULL ポインタの場合 */</i>

Properties

r_pdc_rx_if.h にプロトタイプ宣言されています。

Description

引数 p_fifo_addr に PDC 受信データレジスタ（PCDR）のアドレスを格納します。

Example

Case 1: DMAC (RX ファミリ DMA コントローラ DMCA 制御モジュール Firmware Integration Technology) を使用している場合の設定例

```
#include "platform.h"
#include "r_pdc_rx_if.h"
#include "r_dmaca_rx_if.h"

/* Error code of PDC API */
volatile pdc_return_t      ret_pdc;
/* Error code of DMACA FIT API */
volatile dmaca_return_t    ret_dmaca;
/* Setting values of dmaca_transfer information structure */
dmaca_transfer_data_cfg_t  td_cfg;
/* Pointer to FIFO address of PDC */
uint32_t                  pdc_fifo_address;

/* Set PDC FIFO to DMACA transfer source address */
ret_pdc = R_PDC_GetFifoAddr(&pdc_fifo_address);
if (PDC_SUCCESS == ret_pdc)
{
    td_cfg.p_src_addr = pdc_fifo_address;
}
/* Set PCDFI to DMACA activation source */
td_cfg.act_source = IR_PDC_PCDFI;

ret_dmaca = R_DMACA_Create (DMACA_CH0, &td_cfg);
if (DMACA_SUCCESS != ret_dmaca)
{
    /* Error processing */
}
```

Case 2: DTC（RX ファミリ DTC モジュール Firmware Integration Technology）を使用している場合の設定例

```
#include "platform.h"
#include "r_pdc_rx_if.h"
#include "r_dtc_rx_if.h"

/* Error code of PDC API */
volatile pdc_return_t      ret_pdc;
/* Error code of DTC FIT API */
volatile dtc_err_t         ret_dtc;
/* Activation source of DTC */
dtc_activation_source_t    act_source;
/* Pointer to start address of Transfer data area on RAM */
dtc_transfer_data_t        *p_transdata_dtc;
/* Pointer to setting values for transfer data */
dtc_transfer_data_cfg_t    *p_data_dtc;
/* Pointer to FIFO address of PDC */
uint32_t                   pdc_fifo_address;
/* Number of chain transfer */
uint32_t                   chain_trans_nr;

/* Set PCDFI to DTC Activation source */
act_source = (dtc_activation_source_t)VECT_PDC_PCDFI;
/* Set PDC FIFO to DTC transfer source address */
ret_pdc = R_PDC_GetFifoAddr(&pdc_fifo_address);
if (PDC_SUCCESS == ret_pdc)
{
    p_data_dtc->source_addr = pdc_fifo_address;
}
/* Set 0 to number of chain transfer */
chain_trans_nr = 0;

ret_dtc = R_DTC_Create(act_source, p_transdata_dtc, p_data_dtc, chain_trans_nr);
if(DTC_SUCCESS != ret_dtc)
{
    /* Error processing */
}
```

Special Notes:

なし

R_PDC_GetVersion()

API のバージョンを返す関数です。

Format

uint32_t R_PDC_GetVersion (void)

Parameters

なし

Return Values

バージョン番号

Properties

r_pdc_rx_if.h にプロトタイプ宣言されています。

Description

現在インストールされている PDC FIT モジュールのバージョンを返します。バージョン番号はコード化されています。最初の 2 バイトがメジャーバージョン番号で、後の 2 バイトがマイナーバージョン番号です。例えば、バージョンが 4.25 の場合、戻り値は'0x00040019'となります。

Example

```
#include "platform.h"
#include "r_pdc_rx_if.h"

/* Version number */
uint32_t version;

version = R_PDC_GetVersion();
```

Special Notes:

なし

4. 端子設定

PDC FIT モジュールを使用するためには、マルチファンクションピンコントローラ（MPC）で周辺機能の入出力信号を端子に割り付ける（以下、端子設定と称す）必要があります。端子設定は、R_PDC_Open 関数を呼び出す前に行ってください。

e² studio の場合は「FIT Configurator」または「Smart Configurator」の端子設定機能を使用することができます。FIT Configurator、Smart Configurator の端子設定機能を使用すると、端子設定画面で選択したオプションに応じて、ソースファイルが出力されます。そのソースファイルで定義された関数を呼び出すことにより端子を設定できます。詳細は表 4.1を参照してください。

表 4.1 FIT Configurator、Smart Configurator が出力する関数一覧

使用マイコン	出力される関数名	備考
RX64M RX65N RX66N RX71M RX72M RX72N	R_PDC_PinSet()	-

5. 使用方法

5.1 API 使用例

以下に API の使用例として、イメージセンサからの入力画像を DMAC 起動で SDRAM へ転送する場合の動作フロー例とサンプルコードを掲載します。

5.1.1 動作フロー例



6. 付録

6.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 6.1 動作確認環境 (Rev.2.01)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e ² studio V6.00.000
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V2.07.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.01
使用ボード	Renesas Starter Kit+ for RX65N-2MB (型名：RTK50565N2SxxxxxBE)

表 6.2 動作確認環境 (Rev.2.02)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e ² studio V7.03.000
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.02

表 6.3 動作確認環境 (Rev.2.03)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e ² studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99 GCC for Renesas RX 4.08.04.201803 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 IAR C/C++ Compiler for Renesas RX version 4.10.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.03
使用ボード	Renesas Starter Kit+ for RX64M (型名：RTK500564Mxxxxxx)

表 6.4 動作確認環境 (Rev.2.04)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e ² studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99 GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.04
使用ボード	Renesas Starter Kit+ for RX72M (型名：RTK5572Mxxxxxxxxxx)

表 6.5 動作確認環境 (Rev.2.05)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e ² studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99 GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.05
使用ボード	Renesas Starter Kit+ for RX72N (型名：RTK5572Nxxxxxxxxxx)

表 6.6 動作確認環境 (Rev.2.06)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e ² studio Version 2021-10 IAR Embedded Workbench for Renesas RX 4.20.1
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.03.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99 GCC for Renesas RX 8.3.0.202102 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 IAR C/C++ Compiler for Renesas RX version 4.20.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.06
使用ボード	Renesas Starter Kit+ for RX65N-2MB (型名：RTK50565Nxxxxxxxxxx)

表 6.7 動作確認環境（Rev.2.07）

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e ² studio Version 2025-01 IAR Embedded Workbench for Renesas RX 5.10.1
C コンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.07.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0.202411 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 5.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev2.07
使用ボード	-

6.2 トラブルシューティング

- (1) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A : FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e² studio を使用している場合
アプリケーションノート RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

- (2) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_pdc_rx module.」エラーが発生します。

A : 追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

- (3) Q : 本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「ERROR - PDC_CFG_xxx_xxx」エラーが発生します。

A : “r_pdc_rx_config.h”ファイルの設定値が間違っている可能性があります。“r_pdc_rx_config.h”ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

- (4) Q : 一定時間経過しても PDC のリセットが解除されません。

A : 正しく端子設定が行われていない可能性があります。本 FIT モジュールを使用する場合は端子設定が必要です。詳細は「4 端子設定」を参照してください。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
2.00	2016.10.01	—	初版発行
2.01	2017.10.02	—	RX65N-2MB 版に対応
		5	「2.4 使用する割り込みベクタ」を追加
		13	「2.12 FIT モジュールの追加方法」の内容を変更
		21	「3.1 R_PDC_Open()」 Special Notes の内容を変更
		37	「4. 端子設定」の内容を変更
		39	「6.1 動作確認環境」を追加
		39	「6.2 トラブルシューティング」を追加
2.02	2019.02.01	39	「表 6.2 動作確認環境 (Rev.2.02)」を追加
		—	機能関連 Smart Configurator での GUI によるコンフィグオプション設定機能に対応 ■内容 GUI によるコンフィグオプション設定機能に対応するため、設定ファイルを追加。
2.03	2019.05.20	—	以下のコンパイラに対応 ・ GCC for Renesas RX ・ IAR C/C++ Compiler for Renesas RX
		1	「関連ドキュメント」に R01AN1723、R01AN1826、R20AN0451 を削除
		1	「対象コンパイラ」を追加
		5	「2.2 ソフトウェアの要求」 依存する r_bsp モジュールのリビジョンを追加
		6	「2.8 コードサイズ」を更新
		39	「6.1 動作確認環境」に、「表 6.3 動作確認環境 (Ver.2.03)」を追加
2.04	2019.07.30	—	RX72M 版に対応
		1	「関連ドキュメント」に R01AN1833 を削除
		5	「表 2.1 使用する割り込みベクター一覧」を更新
		6	「2.8 コードサイズ」を更新
		14	「2.13 for 文、while 文、do while 文について」を追加。
		15-37	API 説明ページの「Reentrant」項目を削除
		38	「表 4.1 FIT Configurator、Smart Configurator が出力する関数一覧」を更新
		41	「表 6.4 動作確認環境 (Rev.2.04)」を追加

Rev.	発行日	改訂内容	
		ページ	ポイント
2.05	2019.11.22	—	RX66N、RX72N 版に対応
		5	「表 2.1 使用する割り込みベクター一覧」を更新
		6	「2.8 コードサイズ」を更新
		38	「表 4.1 FIT Configurator、Smart Configurator が出力する関数一覧」を更新
		41	「表 6.5 動作確認環境 (Rev.2.05)」を追加
2.06	2022.01.07	プログラム	ソフトウェア不具合のため、パラレルデータキャプチャユニット FIT モジュールを改修 ■内容 受信動作および継続受信動作中に” R_PDC_Close” 関数を呼び出したときに、PCCR0 レジスタを書き換えることができないため ” R_PDC_Close” 関数の処理が終了しない場合がある。 ■発生条件 ”R_PDC_Control”関数を呼び出してデータキャプチャを開始し、フレームエンド割り込みもしくはエラー割り込みでデータキャプチャを終了するまでの受信動作および継続受信動作中に” R_PDC_Close” 関数を呼び出した場合。 ■対策 パラレルデータキャプチャユニット FIT モジュール Rev2.06 を使用してください。本修正により、以下の関数を変更しています。 R_PDC_Close 関数 対応ツールニュース番号 :R20TS0307
		6	「2.8 コードサイズ」を更新
		10	「2.10 戻り値」に「PDC_ERR_ONGOING」を追加。
		23	「3.2 R_PDC_Close」の Return Values 及び Special Notes の内容を変更。
		24	「3.3 R_PDC_Control」の Return Values 及び Special Notes の内容を変更。
		41	「表 6.6 動作確認環境 (Rev.2.06)」を追加
2.07	2025.03.15	プログラム	FIT モジュールの免責事項と著作権を更新。
		42	「表 6.7 動作確認環境 (Rev.2.07)」を追加

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. 静電気対策

CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレイやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

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. 電源オフ時における入力信号

当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

未使用端子は、「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。

5. クロックについて

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。

6. 入力端子の印加波形

入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、 V_{IL} (Max.) から V_{IH} (Min.) までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、 V_{IL} (Max.) から V_{IH} (Min.) までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

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、将来の拡張機能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。

8. 製品間の相違について

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、フラッシュメモリ、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、ノイズ幅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。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、お客様の責任において、お客様の機器・システムを設計ください。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. 当社製品または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. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
 5. 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。
 6. 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、当社は一切その責任を負いません。
 7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
 8. 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。
 9. 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。
 10. 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、当社は、一切その責任を負いません。
 11. 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。
 12. 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。
 13. 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。
 14. 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.5.0-1 2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。