RX ファミリ

QSPIX モジュール Firmware Integration Technology

要旨

本書では、RX ファミリ MCU 向けの QSPIX モジュールについて説明します。このモジュールは、Firmware Integration Technology（FIT）を採用しています。QSPIX ドライバのアーキテクチャ、FIT モジュールのユーザアプリケーションへの統合、API の使用方法について詳しく解説します。

このモジュールによってサポートされる RX ファミリ MCU には、単一チャネル用 Quad Serial Peripheral Interface（QSPIX）が内蔵されています。Quad-SPI Memory Interface（QSPIX）モジュールは、SPI 対応インタフェースを持つシリアル ROM の接続用メモリコントローラです。これには、シリアルフラッシュメモリ、シリアル EEPROM、シリアル FeRAM などの不揮発性メモリが含まれます。

動作確認デバイス

この API によって現在サポートされているデバイスは、以下のとおりです。

- RX671 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。

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

- Firmware Integration Technology ユーザーズマニュアル（R01AN1833）
- 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）

目次

1. 概要 4
1.1 QSPIX FIT モジュール 4
1.2 QSPIX FIT モジュールの概要 5
1.3 QSPIX FIT モジュールを使用する 6
1.4 API の概要 6
1.5 ハードウェア設定 7
1.5.1 ハードウェアの構成例 7
1.5.2 端子リスト 7

2. API 情報 8
2.1 ハードウェアの要求 8
2.2 ソフトウェアの要求 8
2.3 制限事項 8
2.3.1 RAM の配置に関する制限事項 8
2.4 サポートされているツールチェーン 8
2.5 使用する割り込みベクタ 9
2.6 ヘッダファイル 9
2.7 整数型 9
2.8 コンパイル時の設定 9
2.9 コードサイズ 10
2.10 引数 10
2.11 戻り値 12
2.12 コールバック関数 13
2.13 FIT モジュールの追加方法 13
2.14 for 文、while 文、do while 文について 14

3. API 関数 15
R_QSPIX_Open() 15
R_QSPIX_Control() 18
R_QSPIX_Write_Indirect() 20
R_QSPIX_Read_Indirect() 22
R_QSPIX_Read_Memory_Map() 24
R_QSPIX_Enter_XIP() 26
R_QSPIX_Exit_XIP() 27
R_QSPIX_BankSet() 28
R_QSPIX_Set_Spi_Protocol() 29
R_QSPIX_Get_Status() 30
R_QSPIX_Close() 31
R_QSPIX_GetVersion() 32

4. 端子設定 33

5. デモプロジェクト 34
5.1 qspix_demo_rskrx671, qspix_demo_rskrx671_gcc 34
5.2 ワークスペースにデモを追加する 34

5.3 デモのダウンロード方法 34

6. 付録 35
6.1 動作確認環境 35
6.2 トラブルシューティング 39

7. 参考ドキュメント 40

テクニカルアップデートの対応について 40

改訂記録 41

1. 概要

QSPIX FIT モジュールを他の FIT モジュールと組み合わせると、ターゲットシステムに容易に統合できます。

QSPIX FIT モジュールの関数は、API としてソフトウェアプログラムに組み込むことができます。

このソフトウェアを使用する前に、RX MCU ユーザーズマニュアル：ハードウェア編の、QSPIX 周辺機能に関する章を確認することを推奨します。

1.1 QSPIX FIT モジュール

QSPIX FIT モジュールは、API としてプロジェクトに組み込むことで使用できます。この FIT モジュールをプロジェクトに組み込む方法に関する詳細は、「2.13 FIT モジュールの追加方法」を参照してください。

1.2 QSPIX FIT モジュールの概要

QSPIX FIT モジュールをプロジェクトに追加したら、インストール用にソフトウェアを構成するために、r_qspix_rx_config.h ファイルを変更する必要があります。

構成オプションに関する詳細は、「2.8 コンパイル時の設定」を参照してください。

QSPIX FIT モジュールには、入出力ポートのレジスタを初期化する関数は含まれていません。入出力ポートの設定は、本モジュール以外で行う必要があります。

入出力ポートに関する詳細は、「4. 端子設定」を参照してください。

下の表に本モジュールの関数を簡単にまとめました。

表 1.1　QSPIX の仕様

パラメータ	仕様
チャネル数	1 チャネル
SPI	• 拡張 SPI、デュアル SPI、クアッド SPI プロトコルをサポート • SPI モード 0 および SPI モード 3 に構成可能 • アドレス幅を 8、16、24、32 ビットに選択可能
タイミング調整関数	幅広いシリアルフラッシュをサポートするよう構成可能
メモリマッピングモード	• 読み取り、高速読み取り、高速読み取りデュアル出力、高速読み取りデュアル入出力、高速読み取りクアッド出力、高速読み取りクアッド入出力命令をサポート • 置き換え可能な命令コード • 調整可能なダミーサイクル数 • プリフェッチ関数 • ポーリング処理 • SPI バスサイクル拡張関数
間接アクセスモード	消去、書き込み、ID 読み取り、パワーダウン制御など、ソフトウェア制御により多種多様なシリアルフラッシュ命令および関数を柔軟にサポート
割り込み要因	エラー割り込み

1.3 QSPIX FIT モジュールを使用する

QSPIX FIT モジュールを C++プロジェクト内で使用する

C++プロジェクトでは、FIT QSPIX モジュールのインタフェースヘッダファイルを extern “C”の宣言に追加してください。

```
Extern “C”
{
      #include “r_smc_entry.h”
      #include “r_qspix_rx_if.h”
}
```

1.4 API の概要

表1.2 API 関数一覧に本モジュールに含まれる API 関数を示します。本モジュールによって使用されるコードセクションのサイズについては、「2.9 コードサイズ」の表も参照してください。

表 1.2　API 関数一覧

関数	関数説明
R_QSPIX_Open()	QSPIX チャネルへの出力適用、関連レジスタの初期化、割り込みの有効化、他の API 関数で使用するためのチャネルハンドルの提供を行う関数です。
R_QSPIX_Control()	設定変更に使用される関数です。
R_QSPIX_Write_Indirect()	QSPIX に生データを直接書き込む関数です。
R_QSPIX_Read_Indirect()	QSPIX から生データを直接読み取る関数です。この API は、read_after_write を True に設定した R_QSPIX_Write_indirect()の後でのみ呼び出すことができます。
R_QSPIX_Read_Memory_Map()	QSPI メモリ空間にアクセスしてデータを読み出します。
R_QSPIX_Enter_XIP()	XIP（execute in place）モードに入る関数です。
R_QSPIX_Exit_XIP()	XIP（execute in place）モードを終了する関数です。
R_QSPIX_BankSet()	アクセスするバンクを選択する関数です。
R_QSPIX_Set_Spi_Protocol()	アクセスするプロトコルを選択する関数です。
R_QSPIX_Get_Status()	プリフェッチのステータス、ビジーフラグ、ROM アクセスエラーフラグを取得する関数です。
R_QSPIX_Cloes()	QSPIX ドライバモジュールを終了する関数です。
R_QSPIX_GetVersion()	API のバージョンを返す関数です。

1.5　ハードウェア設定

1.5.1　ハードウェアの構成例

図1.1 は接続図です。高速処理を実現するためには、ダンピング抵抗またはコンデンサを追加し、各種信号線の回路のマッチングを向上させることを検討してください。

プルアップ処理の実行に失敗しないでください。プルアップ処理を行わないと、スレーブデバイスは書き込み不可状態になったり、データラインが Hi-Z 状態のときに状態を保持したりする可能性があります。

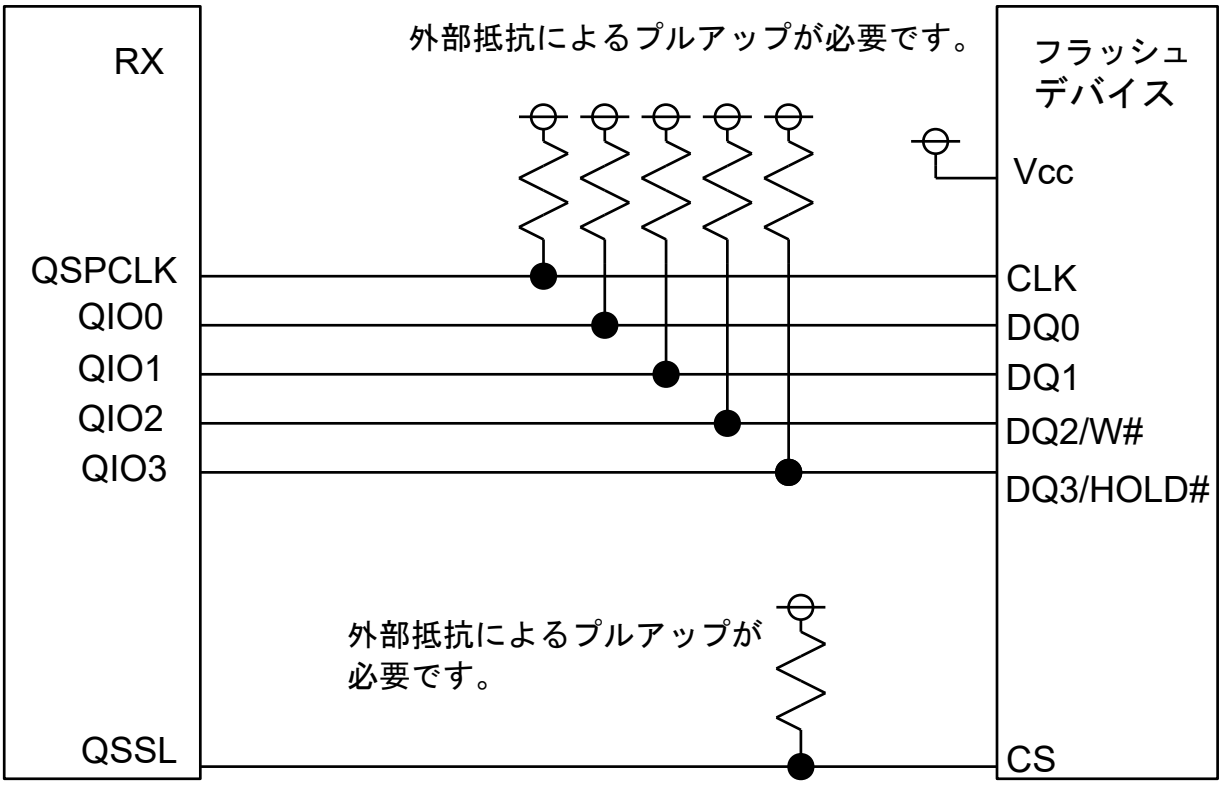

図1.1　RX ファミリ MCU QSPI およびフラッシュドライブ用サンプル配線図

1.5.2　端子リスト

表1.3 は、使用される端子とその用途をまとめたものです。

表 1.3　端子一覧

端子名	入出力	説明
QSPCLK	出力	QSPIX クロック出力端子
QSSL	出力	QSPIX スレーブ選択端子
QIO0	入出力	データ 0 入力/出力
QIO1	入出力	データ 1 入力/出力
QIO2	入出力	データ 2 入力/出力
QIO3	入出力	データ 3 入力/出力

2. API 情報

このドライバ API は、ルネサスの API 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。

- QSPIX

2.2 ソフトウェアの要求

このドライバは、以下のパッケージに依存しています。

- ボードサポートパッケージ (r_bsp) v6.10 以上

2.3 制限事項

2.3.1 RAM の配置に関する制限事項

FIT では、NULL と同じ値を API 関数のポインタ引数として設定すると、パラメータチェックによってエラーが返されることがあります。したがって、NULL と同じ値をポインタ引数として API 関数に受け渡さないでください。

ライブラリ関数仕様により、NULL 値は 0 として定義されます。したがって、上記の現象は、API 関数ポインタ引数に受け渡される変数または関数が、RAM の先頭アドレス（アドレス 0x0）に配置される場合に発生します。この場合、API 関数ポインタ引数に受け渡される変数または関数がアドレス 0x0 に位置付けられないように、セクション設定を変更するか、RAM の先頭に置くダミー変数を準備してください。

CCRX プロジェクト（e^2 studio V21.1.0）の場合、変数がアドレス 0x0 に位置付けられないようにするため、RAM 先頭アドレスは 0x4 として設定されます。GCC プロジェクト（e^2 studio V21.1.0）および IAR プロジェクト（EWRX V4.20.1）の場合、RAM 先頭アドレスは 0x0 であるため、前述の対策が必要です。

セクションのデフォルト設定は、IDE バージョンのアップグレードによって変更される場合があります。最新の IDE を使用する場合は、セクション設定を確認してください。

2.4 サポートされているツールチェーン

QSPIX 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を使用して動作確認を行っています。

2.5　使用する割り込みベクタ

QSPIX（ROM アクセスエラー）を有効にするには、R_QSPIX_Open 関数を実行します（マクロ定義 QSPIX_CFG_ERI_INCLUDED = 1）。

表2.1 に 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覧

デバイス	割り込みベクタ
RX671	GROUPAL0 割り込み（ベクタ番号：30)

2.6　ヘッダファイル

すべての API 呼び出しおよび使用されるインタフェース定義は、r_qspix_rx_if.h に記述されています。

各ビルドのコンフィグレーションオプションは、r_qspix_rx_config.h で選択されます。

2.7　整数型

このドライバは 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。

2.8　コンパイル時の設定

本モジュールのコンフィグレーションオプションの設定は、r_qspix _rx_config.h に記述されています。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。

r_qspix_rx_config.h のコンフィグレーションオプション	
定義	説明
QSPIX_CFG_PARAM_CHECKING_ENABLE (1)	1：パラメータチェック処理がビルドに含まれます。 0：パラメータチェック処理がビルドから省略されます。
QSPIX_CFG_USE_CH0　(1)	QSPIX チャネルを有効にします。 (0)　= 使用しない (1) = 使用する
QSPIX_CFG_ERI_INCLUDED　　(1)	1：ROM アクセスエラーがビルドに含まれます。 0：ROM アクセスエラーがビルドから省略されます。
QSPIX_CFG_ERI_IR_PRIORITY　　(3)	チャネルに ROM アクセスエラー割り込み優先順位を設定します。“1”～“15”で選択した値を割り込み優先レベルに設定します。

2.9 コードサイズ

本モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを下表に示します。

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.8 コンパイル時の設定」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_qspix_rx rev1.40

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00
(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 8.3.0.202202
(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 4.20.3
(統合開発環境のデフォルト設定)

コンフィグ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	
		パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし	パラメータチェックあり	パラメータチェックなし
RX671	ROM	**1859 バイト**	**1452 バイト**	**2416 バイト**	**1980 バイト**	**3201 バイト**	**2607 バイト**
	RAM	**72 バイト**		**128 バイト**		**32 バイト**	
	スタック	**16 バイト**		**-**		**100 バイト**	

2.10 引数

API 関数の引数で使用する構造体を下に示します。この構造体は、API 関数のプロトタイプ宣言と共に r_qspix_ rx_if.h に記述されます。

```
typedef struct st_qspix_cfg
{
    /* 読み取り命令選択 */
    qspix_read_mode_t                       read_mode;

    /* スレーブ選択拡張 */
    qspix_slave_select_extension_t          slave_select_extension;

    /* スレーブ選択拡張 */
    qspix_prefetch_function_t               prefetch_function;

    /* クロックモード選択 */
    qspix_clock_mode_t                      clock_mode;

    /* データ出力期間拡張 */
```

```
    qspix_data_output_select_t              data_output_select;

    /* 特別読み取り命令選択 */
    qspix_special_instruction_select_t      special_instruction_select;

    /* スレーブ選択 High 幅設定 */
    qspix_slave_select_high_width_t         slave_select_high_width;

    /* スレーブ選択保持時間設定 */
    qspix_slave_select_hold_time_t          slave_select_hold_time;

    /* スレーブ選択セットアップ時間設定 */
    qspix_slave_select_setup_time_t         slave_select_setup_time;

    /* クロック分周器選択 */
    qspix_clock_divider_t                   clock_divider;

    /* special_instruction_select = QSPIX_INSTRUCTION_CODE_IN_SPRIR_REGISTER
       のときに使用される特別読み取り命令設定 */
    uint32_t qspix_special_read_instruction;

    /* アドレスサイズ設定 */
    qspix_address_size_t                    address_size;

    /* 4 バイトアドレスの有効化による命令 */
    qspix_instruction_4_Byte_address_t      instruction_4_Byte_address;

    /* ダミーサイクルの数 */
    qspix_dummy_clocks_t                    dummy_clocks;

    /* SPI プロトコル */
    qspix_protocol_t                        protocol;

    /* WP 端子制御 */
    qspix_WP_pin_control_t                  WP_pin_control;

    void (*p_callback)(void *p_cbdat);    /* ユーザコールバック関数へのポインタ */

} qspix_cfg_t;
```

2.11 戻り値

API 関数の戻り値を以下に示します。この列挙型は、API 関数のプロトタイプ宣言と共に r_qspix_rx_if.h に記述されます。

```
typedef enum e_qspix_err        /* QSPIX API エラーコード */
{
    QSPIX_SUCCESS,              /* QSPIX 処理が問題なく完了した。 */
    QSPIX_ERR_OPENED,           /* QSPIX はすでに初期化されていた。 */
    QSPIX_ERR_NOT_OPEN,         /* QSPIX モジュールがまだ初期化されていない。 */
    QSPIX_ERR_INVALID_ARG,      /* 引数が無効。 */
    QSPIX_ERR_INVALID_COMMAND,  /* コマンドパラメータが無効。または強制データ変更が失敗
した。 */
    QSPIX_ERR_NULL_PTR,         /* 引数ポインタが NULL。 */
    QSPIX_ERR_BUSY,     /* QSPIX リソースが別のプロセスによってロックされている。 */
    QSPIX_ERR_HW                /* HW エラー。 */
} qspix_err_t;
```

2.12 コールバック関数

本モジュールでは、ユーザが指定したコールバック関数は、ROM アクセスエラー割り込みが発生すると呼び出されます。

コールバック関数は、“void (*p_callback)(void *p_cbdat)”構造体メンバにユーザ関数のアドレスを保存することで指定されます（「2.10 引数」を参照）。

2.13 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)、(2)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(3)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) 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。

(3) 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。

2.14 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_QSPIX_Open()

QSPIX チャネルへの出力適用、関連レジスタの初期化、割り込みの有効化、他の API 関数で使用するためのチャネルハンドルの提供を行う関数です。

Format

```
qspix_err_t  R_QSPIX_Open (uint8_t channel, qspix_cfg_t * p_cfg)
```

Parameters

channel

初期化する QSPIX チャネルの数

**p_cfg*

QSPIX チャネル構成データ構造体へのポインタ

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_OPENED,	/* QSPIX はすでに初期化されていた。 */
QSPIX_ERR_INVALID_ARG	/* 引数が無効。 */
QSPIX_ERR_NULL_PTR	/* 引数ポインタが NULL。 */
QSPIX_ERR_BUSY	/* QSPIX リソースが別のプロセスによってロックされている。 */

Properties

宣言は r_qspix_rx_if.h に記述されています。

Description

Open 関数は、処理のための QSPIX チャネルの準備を担当します。この関数は、他の QSPIX API 関数（R_QSPIX_GetVersion を除く）を呼び出す前に 1 回呼び出しておく必要があります。

正常に完了すると、選択した QSPIX チャネルのステータスは“open”に設定されます。

その後、最初に R_QSPIX_Close()を呼び出して“close”を実行せずに、同じ QSPIX チャネルに対してこの関数をもう一度呼び出さないでください。

この処理が完了しても、コミュニケーションはまだ利用できません。

ドライバが開くと、QSPIX にはアドレス 0x70000000 で始まる内部フラッシュメモリのようにアクセスすることができます。

Example

条件：チャネルがまだ開いていない

```
void rom_error_access(void *p_cbdat);
void rom_error_access(void *p_cbdat)
{
  /* ハンドル ROM エラーアクセス用コード */

  R_BSP_NOP();
}
qspix_err_t ret = QSPIX_SUCCESS;
qspix_cfg_t p_cfg;
/* 読み取りモードを設定 */
p_cfg.read_mode = QSPIX_READ_MODE_STANDARD;


/* スレーブ選択拡張を設定 */
p_cfg.slave_select_extension = QSPIX_DO_NOT_EXTEND_QSSL;

/* プリフェッチ関数の有効化を設定 */
p_cfg.prefetch_function = QSPIX_PREFETCH_ENABLE;

/* クロックモードを設定 */
p_cfg.clock_mode = QSPIX_SPI_MODE_0;

/* データ出力期間拡張を設定 */
p_cfg.data_output_select = QSPIX_DO_NOT_EXTEND;

/* 特別読み取り命令選択を設定 */
p_cfg.special_instruction_select=QSPIX_DEFAULT_INSTRUCTION_CODE;

/* スレーブ選択 High 幅設定を設定 */
p_cfg.slave_select_high_width=QSPIX_1_CYCLE_QSPCLK;

/* スレーブ選択保持時間設定を設定 */
p_cfg.slave_select_hold_time=QSPIX_RELEASE_QSSL_0_5_QSPCLK;

/* スレーブ選択セットアップ時間設定を設定 */
p_cfg.slave_select_setup_time=QSPIX_OUTPUT_QSSL_0_5_QSPCLK;

/* クロック分周器選択を設定 */
p_cfg.clock_divider=QSPIX_ICLK_DIV_2;

/* アドレスサイズを設定 */
p_cfg.address_size=QSPIX_3_BYTES;

/* 4 バイトアドレスの有効化による命令を設定 */
p_cfg.instruction_4_Byte_address=QSPIX_INSTRUCTION_4_BYTE_ADDRESS_DISABLE;

/* ダミーサイクルの数を設定 */
p_cfg.dummy_clocks=QSPIX_DEFAULT_DUMMY_CYCLES;

/* SPI プロトコルを設定 */
p_cfg.protocol=QSPIX_EXTENDED_SPI_PROTOCOL;

/* WP 端子制御を設定 */
```

```
    p_cfg.WP_pin_control=QSPIX_LOW_LEVEL;

    /* コールバック関数を設定 */
    p_cfg.p_callback = rom_error_access;

    /* R_QSPIX_Open を呼び出す */
    ret = R_QSPIX_Open(QSPIX_CH0,&p_cfg);

    /* エラーが発生した場合、この関数は API 呼び出しのエラー検出をデモンストレーションします。
*/
    if (QSPIX_SUCCESS != ret)
    {
        return error(); // エラー処理コードがここに記述されます。
    }

    /* QSPIX 周辺機能で使用するために入出力ポート端子を初期化。
     * これは、MCU および選択ポートに固有のものです。 */
    R_QSPIX_PinSet_QSPIX0();
```

Special Notes:

なし

R_QSPIX_Control()

設定変更に使用される関数です。

Format

```
qspix_err_t  R_QSPIX_Control(uint8_t channel, qspix_cfg_t *p_cfg);
```

Parameters

channel

制御する QSPIX チャネルの数

**p_cfg*

QSPIX チャネル構成データ構造体へのポインタ

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_NOT_OPEN	/* QSPIX モジュールがまだ初期化されていない。 */
QSPIX_ERR_INVALID_ARG	/* 引数が無効。 */
QSPIX_ERR_NULL_PTR	/* 引数ポインタが NULL。 */

Properties

宣言は r_qspix_rx_if.h に記述されています。

Description

特定モード用に QSPIX チャネルの動作を変更する関数です。この関数は、R_QSPIX_Open の後で呼び出す必要があります。

Example

```
void rom_error_access(void *p_cbdat);
void rom_error_access(void *p_cbdat)
{
  /* ハンドル ROM エラーアクセス用コード */

  R_BSP_NOP();
}
qspix_err_t ret = QSPIX_SUCCESS;
qspix_cfg_t p_cfg;

/* 読み取りモードを設定 */
p_cfg.read_mode = QSPIX_READ_MODE_STANDARD;

/* スレーブ選択拡張を設定 */
p_cfg.slave_select_extension = QSPIX_DO_NOT_EXTEND_QSSL;

/* プリフェッチ関数の有効化を設定 */
p_cfg.prefetch_function = QSPIX_PREFETCH_ENABLE;

/* クロックモードを変更 */
p_cfg.clock_mode = QSPIX_SPI_MODE_3;

/* データ出力期間拡張を設定 */
p_cfg.data_output_select = QSPIX_DO_NOT_EXTEND;
```

```
/* 特別読み取り命令選択を設定 */
p_cfg.special_instruction_select=QSPIX_DEFAULT_INSTRUCTION_CODE;

/* スレーブ選択 High 幅設定を設定 */
p_cfg.slave_select_high_width=QSPIX_1_CYCLE_QSPCLK;

/* スレーブ選択保持時間設定を設定 */
p_cfg.slave_select_hold_time=QSPIX_RELEASE_QSSL_0_5_QSPCLK;

/* スレーブ選択セットアップ時間設定を設定 */
p_cfg.slave_select_setup_time=QSPIX_OUTPUT_QSSL_0_5_QSPCLK;

/* クロック分周器選択を設定 */
p_cfg.clock_divider=QSPIX_ICLK_DIV_2;

/* アドレスサイズを設定 */
p_cfg.address_size=QSPIX_3_BYTES;

/* 4 バイトアドレスの有効化による命令を設定 */
p_cfg.instruction_4_Byte_address=QSPIX_INSTRUCTION_4_BYTE_ADDRESS_DISABLE;

/* ダミーサイクルの数を設定 */
p_cfg.dummy_clocks=QSPIX_DEFAULT_DUMMY_CYCLES;

/* SPI プロトコルを設定 */
p_cfg.protocol=QSPIX_EXTENDED_SPI_PROTOCOL;

/* WP 端子制御を設定 */
p_cfg.WP_pin_control=QSPIX_LOW_LEVEL;

/* コールバック関数を設定 */
p_cfg.p_callback = rom_error_access;

/* R_QSPIX_Control を呼び出す */
ret = R_QSPIX_Control(QSPIX_CH0,&p_cfg);

/* エラーが発生した場合、この関数は API 呼び出しのエラー検出をデモンストレーションします。
*/
if (QSPIX_SUCCESS != ret)
{
    return error(); // エラー処理コードがここに記述されます。
}
```

Special Notes:

なし

R_QSPIX_Write_Indirect()

QSPIX に生データを直接書き込む関数です。

Format

```
qspix_err_t  R_QSPIX_Write_Indirect(uint8_t channel,
                                    uint8_t *p_src_addr,
                                    uint32_t bytes,
                                    bool read_after_write);
```

Parameters

channel

データの書き込みに使用する QSPIX のチャネル

**p_src_addr*

書き込むデータへのポインタ

bytes

書き込み用バイトの数

read_after_write

SPI バスサイクルを閉じるか否か

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_NOT_OPEN	/* QSPIX モジュールがまだ初期化されていない。 */
QSPIX_ERR_INVALID_ARG	/* 引数が不正である場合。 */
QSPIX_ERR_NULL_PTR	/* 引数ポインタが NULL。 */

Properties

プロトタイプ宣言は“r_qspix_rx_if.h”ファイルに記述されています。

Description

データ書き込み用の引数チャネルによって指定されるチャネル数の QSPIX を使用します。
バイト数は、使用されるフラッシュ（最大サイズは 1 ページ）に依存します。
大容量のデータを書き込むとき、コミュニケーションはページ単位に分割されます。

Example

```
#define QSPIX_COMMAND_READ_MODE_STANDARD (0x03)
/* 条件：チャネルが現在開いている。 */
/* 保存するデータへのポインタ */
uint8_t data[4];
/* 書き込むデータへのポインタ */
uint8_t buffer [10];

data[0] = QSPIX_COMMAND_READ_MODE_STANDARD;
data[1] = 0x00; Input the higher-order byte at address 0x000000.
data[2] = 0x00; Input the middle byte at target address 0x000000.
data[3] = 0x00; Input the lower-order byte at target address 0x000000.

/* 標準 0x03 コマンド、その次に 3 アドレスバイトを送信 */
R_QSPIX_Write_Indirect(QSPIX_CH0, &data[0],4,true);

/* フラッシュから 10 バイトを読み取る */
R_QSPIX_Read_Indirect(QSPIX_CH0, &buffer[0],10);
```

Special Notes:

間接アクセスモードで使用できるのは、拡張 SPI プロトコルのみです。また、シリアルフラッシュの内容を参照するためには、Read または Fast Read 命令を使用する必要があります。この構成では、QSPIX は Fast Read Dual Output、Fast Read Dual I/O、Fast Read Quad Output、Fast Read Quad I/O 転送をサポートしていません。これらの高速な読み出し動作が必要な場合は、通常のフラッシュアクセスを使用してください。

R_QSPIX_Read_Indirect()

QSPIX から生データを直接読み取る関数です。この API は、read_after_write を True に設定した R_QSPIX_Write_Indirect()の後でのみ呼び出すことができます。

Format

```
qspix_err_t  R_QSPIX_Read_Indirect(uint8_t channel,
                                   uint8_t *p_des_addr,
                                   uint32_t bytes);
```

Parameters

channel

データの読み取りに使用する QSPIX のチャネル

**p_des_addr*

保存するデータへのポインタ

bytes

読み取り用バイトの数

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_NOT_OPENED	/* QSPIX モジュールがまだ初期化されていない。 */
QSPIX_ERR_INVALID_ARG	/* 引数が不正である場合。 */
QSPIX_ERR_NULL_PTR	/* 引数ポインタが NULL。 */

Properties

宣言は r_qspix_rx_if.h に記述されています。

Description

データ読み取り用の引数チャネルによって指定されるチャネル数の QSPIX を使用します。

バイト数は、使用されるフラッシュに依存します。

Example

```
#define QSPIX_COMMAND_READ_MODE_STANDARD (0x03)
/* 条件：チャネルが現在開いている。 */
/* 保存するデータへのポインタ */
uint8_t data[4];
/* 書き込むデータへのポインタ */
uint8_t buffer [10];

data[0] = QSPIX_COMMAND_READ_MODE_STANDARD;
data[1] = 0x00; Input the higher-order byte at address 0x000000.
data[2] = 0x00; Input the middle byte at target address 0x000000.
```

```
data[3] = 0x00; Input the lower-order byte at target address 0x000000.

/* 標準 0x03 コマンド、その次に 3 アドレスバイトを送信 */
R_QSPIX_Write_Indirect(QSPIX_CH0, &data[0],4,true);

/* フラッシュから 10 バイトを読み取る */
R_QSPIX_Read_Indirect(QSPIX_CH0, &buffer[0],10);
```

Special Notes:

間接アクセスモードで使用できるのは、拡張 SPI プロトコルのみです。また、シリアルフラッシュの内容を参照するためには、Read または Fast Read 命令を使用する必要があります。この構成では、QSPIX は Fast Read Dual Output、Fast Read Dual I/O、Fast Read Quad Output、Fast Read Quad I/O 転送をサポートしていません。これらの高速な読み出し動作が必要な場合は、通常のフラッシュアクセスを使用してください。

R_QSPIX_Read_Memory_Map()

QSPI メモリ空間にアクセスしてデータを読み出します。

Format

```
qspix_err_t  R_QSPIX_Read_Indirect(uint8_t channel,
                                   uint8_t *p_des_addr,
                                   uint32_t p_addr,
                                   qspix_protocol_t protocol_ext,
                                   qspix_address_size_t addr_size,
                                   uint32_t bytes);
```

Parameters

channel

データの読み取りに使用する QSPIX のチャネル

**p_des_addr*

保存するデータへのポインタ

p_addr

開始アドレスを読み出します

protocol_ext

SPI プロトコルモードを選択します

addr_size

アドレスサイズ

bytes

読み取り用バイトの数

Return Values

```
QSPIX_SUCCESS                /* 問題なく処理が完了した。 */
QSPIX_ERR_NOT_OPENED     /* QSPIX モジュールがまだ初期化されていない。 */
QSPIX_ERR_INVALID_ARG        /* 引数が不正である場合。 */
QSPIX_ERR_NULL_PTR           /* 引数ポインタが NULL。 */
```

Properties

宣言は r_qspix_rx_if.h に記述されています。

Description

データ読み取り用の引数チャネルによって指定されるチャネル数の QSPIX を使用します。

Example

```
    /* 条件：チャネルが現在開いている。 */
    /* 保存するデータへのポインタ */
    uint8_t buffer [10];
    /* 開始アドレスを読み出します。*/
    uint32_t addr_start = 0x1000;
    /* SPI プロトコルモードを選択します */
    qspix_protocol_t protocol_used = QSPIX_EXTENDED_SPI_PROTOCOL;
    /* アドレスサイズ */
    qspix_address_size_t addr_size = QSPIX_3_BYTES;

    /* フラッシュから 10 バイトを読み取る */
    R_QSPIX_Read_Memory_Map(QSPIX_CH0, &buffer[0], addr_start, protocol_used,
addr_size, 10);
```

Special Notes:

R_QSPIX_Read_Memory_Map() 関数が呼び出されるたびに、バンク内のアドレスからのみデータを読み取ることができます。指定されたバンク値は、R_QSPIX_BankSet() 関数を呼び出すことによって事前に設定されています。

R_QSPIX_Enter_XIP()

XIP（execute in place）モードに入る関数です。

Format

```
qspix_err_t  R_QSPIX_Enter_XIP(uint8_t channel, uint8_t mode);
```

Parameters

channel

QSPIX のチャネルが XIP（execute in place）モードに入る

mode

XIP モードに入る関数です。

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_NOT_OPEN	/* QSPIX モジュールがまだ初期化されていない。 */
QSPIX_ERR_HW	/* ハードウェアエラー。 */

Properties

プロトタイプ宣言は“r_qspix_rx_if.h”ファイルに記述されています。

Description

XIP モードを選択するには、SPDCR レジスタ内の MODE[7:0]ビットで、シリアルフラッシュデバイスに XIP モード構成を指定します。

Example

```
/* 条件：チャネルが現在開いている。 */
/* XIP に入る */
R_QSPIX_Enter_XIP(QSPIX_CH0,0xA5);
```

Special Notes:

なし

R_QSPIX_Exit_XIP()

XIP（execute in place）モードを終了する関数です。

Format

```
qspix_err_t  R_QSPIX_Exit_XIP(uint8_t channel, uint8_t mode);
```

Parameters

channel

QSPIX のチャネルが XIP（execute in place）モードを終了する

mode

XIP モードを終了する関数です。

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_NOT_OPEN	/* QSPIX モジュールがまだ初期化されていない。 */
QSPIX_ERR_HW	/* ハードウェアエラー。 */

Properties

プロトタイプ宣言は“r_qspix_rx_if.h”ファイルに記述されています。

Description

XIP モードを選択するには、SPDCR レジスタ内の MODE[7:0]ビットで、シリアルフラッシュデバイスに XIP モード構成を指定します。

Example

```
/* 条件：チャネルが現在開いている。 */
/* XIP を終了する */
R_QSPIX_Exit_XIP(QSPIX_CH0, 0xFF);
```

Special Notes:

なし

R_QSPIX_BankSet()

アクセスするバンクを選択する関数です。

Format

```
qspix_err_t  R_QSPIX_BankSet(uint8_t channel, uint8_t bank);
```

Parameters

channel

スイッチバンク用チャネル

bank

設定する必要があるバンク

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_INVALID_ARG	/* 引数が無効。 */
QSPIX_ERR_NOT_OPEN	/* QSPIX モジュールがまだ初期化されていない。 */

Properties

プロトタイプ宣言は“r_qspix_rx_if.h”ファイルに記述されています。

Description

1 つのバンクは、64MB のスライディングアクセスウィンドウで、QSPI デバイスフラッシュメモリ空間へスライドします。チップアドレス 0x4000000 にアクセスするには、バンク 1 を選択してから、内部フラッシュアドレス 0x70000000 から読み取ります。チップアドレス 0x8001000 にアクセスするには、バンク 2 を選択してから、内部フラッシュアドレス 0x70001000 から読み取ります。

この関数は、512Mb（64MB）以下のメモリデバイスには不要です。

Example

```
/* 条件：チャネルが現在開いている。 */
/* バンク 1 を選択 */
R_QSPIX_BankSet(QSPIX_CH0,1);
```

Special Notes:

なし

R_QSPIX_Set_Spi_Protocol()

アクセスするプロトコルを選択する関数です。

Format

```
qspix_err_t  R_QSPIX_Set_Spi_Protocol(uint8_t channel,
                                      qspix_protocol_t protocol);
```

Parameters

channel

設定プロトコル用チャネル

protocol

設定する必要があるプロトコル

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_INVALID_ARG	/* 引数が無効。 */
QSPIX_ERR_NOT_OPEN	/* QSPIX モジュールがまだ初期化されていない。 */

Properties

プロトタイプ宣言は“r_qspix_rx_if.h”ファイルに記述されています。

Description

Example

```
/* 条件：チャネルが現在開いている。 */
/* クアッド SPI プロトコルを選択 */
R_QSPIX_Set_Spi_Protocol(QSPIX_CH0, QSPIX_QUAD_SPI_PROTOCOL);
```

Special Notes:

なし

R_QSPIX_Get_Status()

プリフェッチのステータス、ビジーフラグ、ROM アクセスエラーフラグを取得する関数です。

Format

```
qspix_err_t  R_QSPIX_Get_Status(qspix_cmd_t cmd, uint8_t *return_status);
```

Parameters

cmd

ステータスを取得するコマンド

**return_status*

コマンドに対応するステータスを保存するポインタ

Return Values

QSPIX_SUCCESS	/* 問題なく処理が完了した。 */
QSPIX_ERR_INVALID_COMMAND	/* 引数が無効。 */

Properties

プロトタイプ宣言は“r_qspix_rx_if.h”ファイルに記述されています。

Description

この関数は入力コマンドに対応するステータスを返します。

```
typedef enum e_qspix_cmd
{
    QSPIX_GET_PREFETCH_BUFFER_FILL_LEVEL, /* プリフェッチバッファ格納レベルを取得 */
    QSPIX_GET_PREFETCH_BUFFER_FULL_FLAG, /* プリフェッチフルバッファを取得 */
    QSPIX_GET_PREFETCH_FUNCTION_OPERATING_STATUS_FLAG, /* プリフェッチ機能
                                                動作ステータスを取得 */
    QSPIX_GET_BUS_BUSY_FLAG,         /* バスビジーフラグを取得 */
    QSPIX_GET_ROM_ACCESS_ERROR_FLAG, /* ROM アクセスエラーフラグを取得 */
     QSPIX_GET_XIP_STATUS_FLAG        /*  XIP ステータスフラグを取得 */
 }
qspix_cmd_t;
```

Example

```
/* 条件：チャネルが現在開いている。 */
/* 変数保存先ステータス */
uint8_t status;

/* プリフェッチバッファ充填レベルを取得し、変数ステータスに保存 */
R_QSPIX_Get_Status(QSPIX_GET_PREFETCH_BUFFER_FILL_LEVEL, &status);
```

Special Notes:

なし

R_QSPIX_Close()

QSPIX ドライバモジュールを終了する関数です。

Format

```
qspix_err_t  R_QSPIX_Close(uint8_t channel);
```

Parameters

channel

閉じる必要がある QSPIX のチャネル

Return Values

QSPIX_SUCCESS	/* 成功。チャネルは初期化された。 */
QSPIX_ERR_NOT_OPEN	/* QSPIX モジュールがまだ初期化されていない。 */
QSPIX_ERR_INVALID_ARG	/* 引数が不正である場合。 */

Properties

宣言は r_qspix_rx_if.h に記述されています。

Description

QSPIX チャネルを無効にし、モジュール停止状態に入ります。QSPIX チャネルは、R_QSPIX_Open 関数を実行してもう一度開かれるまで再使用できません。Open 状態ではない QSPIX に対してこの関数が呼び出されると、エラーコードが返されます。

Example

```
/* 条件：チャネルが現在開いている。 */
qspix_err_t ret;
ret = R_QSPIX_Close(QSPIX_CH0);
if (QSPIX_SUCCESS != result)
{
    return result;
}
```

Special Notes:

なし

R_QSPIX_GetVersion()

API のバージョンを返す関数です。

Format

```
uint32_t  R_QSPIX_GetVersion(void);
```

Parameters

なし

Return Values

バージョン番号

Properties

宣言は r_qspix_rx_if.h に記述されています。

Description

この関数は本モジュールのバージョンを返します。バージョン番号は、上位 2 バイトがメジャーバージョン番号、下位 2 バイトがマイナーバージョン番号となるように暗号化されます。

Example

```
/* バージョン番号を取得して文字列に変換。 */
uint32_t version, version_high, version_low;
char version_str[9];

version = R_QSPIX_GetVersion();
version_high = (version >> 16)&0xf;
version_low = version & 0xff;

sprintf(version_str, "QSPIXv%1.1hu.%2.2hu", version_high, version_low);.
```

Special Notes:

なし

4. 端子設定

QSPIX 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_QSPIX_Open 関数を呼び出した後に行ってください。

e2 studio で端子設定を行う場合、スマート・コンフィグレータの端子設定機能を利用できます。端子設定機能を使用する場合、スマート・コンフィグレータの端子設定ウィンドウで選択したオプションに従ってソースファイルが出力されます。端子は、ソースファイルで定義された関数を呼び出すことによって設定されます。詳細については、表 4.1 を参照してください。

表 4.1「スマート・コンフィグレータ」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
全デバイス共通	R_QSPIX_PinSet_QSPIXx	x:チャネル番号

5. デモプロジェクト

デモプロジェクトには、FIT モジュールとそのモジュールが依存するモジュール（例：r_bsp）を使用する main()関数が含まれます。本 FIT モジュールには以下のデモプロジェクトが含まれます。

5.1 qspix_demo_rskrx671, qspix_demo_rskrx671_gcc

サンプルデモでは QSPIX 機能の間接アクセスモードを使用して RSK RX671 上の外部フラッシュメモリにプログラムを書き込みます。

QSPIX は XIP モードに移行した後、外部フラッシュメモリ上で関数を実行します。

外部フラッシュメモリ上の関数は LED0 と LED1 を順に ON/OFF します。

5.2 ワークスペースにデモを追加する

デモプロジェクトは、本アプリケーションノートで提供されるファイルの FITDemos サブディレクトリにあります。ワークスペースにデモプロジェクトを追加するには、「ファイル」 >> 「インポート」を選択し、「インポート」ダイアログから「一般」の「既存プロジェクトをワークスペースへ」を選択して「次へ」ボタンをクリックします。「インポート」ダイアログで「アーカイブ・ファイルの選択」ラジオボタンを選択し、「参照」ボタンをクリックして FITDemos サブディレクトリを開き、使用するデモの zip ファイルを選択して「終了」をクリックします。

5.3 デモのダウンロード方法

デモプロジェクトは、RX Driver Package には同梱されていません。デモプロジェクトを使用する場合は、個別に各 FIT モジュールをダウンロードする必要があります。「スマートブラウザ」の「アプリケーションノート」タブから、本アプリケーションノートを右クリックして「サンプル・コード（ダウンロード）」を選択することにより、ダウンロードできます。

6. 付録

6.1 動作確認環境

本モジュールの動作テスト環境を詳しく説明します。

表 6.1　動作確認環境（Rev.1.42）

項目	内容
統合開発環境	Renesas Electronics e^2 studio Version 2025-01 IAR Embedded Workbench for Renesas RX 5.10.1
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.07.00 コンパイラオプション：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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 5.10.1 コンパイラオプション：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	Rev.1.42
使用ボード	-

表 6.2　動作確認環境（Rev.1.41）

項目	内容
統合開発環境	Renesas Electronics e^2 studio Version 2023-10 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.05.00 コンパイラオプション：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 8.3.0.202305 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：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	Rev.1.41
使用ボード	-

表 6.2　動作確認環境（Rev.1.40）

項目	内容
統合開発環境	Renesas Electronics e^2 studio Version 2023-01 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00 コンパイラオプション：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 8.3.0.202202 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：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	Rev.1.40
使用ボード	Evaluation Kit+ for RX671 (product No.: RTK5EK671xxxxxxxxxx)

表 6.3　動作確認環境（Rev.1.30）

項目	内容
統合開発環境	Renesas Electronics e² studio Version 2022-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.04.00 コンパイラオプション：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 8.3.0.202202 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 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、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 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：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	Rev.1.30
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxx)

表 6.4　動作確認環境（Rev.1.20）

項目	内容
統合開発環境	Renesas Electronics e² studio Version 2021-10 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 コンパイラオプション：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 コンパイラオプション：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 リンカオプション：“Optimize size (-Os)”を使用する場合は、以下のユーザ定義オプションを統合開発環境のデフォルト設定に追加してください。 -Wl,--no-gc-sections リンカが誤って FIT 周辺機能モジュールで宣言された割り込み関数を破棄することによる GCC リンカ問題を解決します。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：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	Rev.1.20
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxx)

表 6.5　動作確認環境（Rev.1.10）

項目	内容
統合開発環境	Renesas Electronics e² studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 コンパイラオプション：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 8.3.0.202004 コンパイラオプション：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 リンカオプション：“Optimize size (-Os)”を使用する場合は、以下のユーザ定義オプションを統合開発環境のデフォルト設定に追加してください。 -Wl,--no-gc-sections リンカが誤って FIT 周辺機能モジュールで宣言された割り込み関数を破棄することによる GCC リンカ問題を解決します。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：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	Rev.1.10
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxxx)

表 6.6　動作確認環境（Rev.1.00）

項目	内容
統合開発環境	Renesas Electronics e² studio Version 2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 コンパイラオプション：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 8.3.0.202004 コンパイラオプション：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 リンカオプション：“Optimize size (-Os)”を使用する場合は、以下のユーザ定義オプションを統合開発環境のデフォルト設定に追加してください。 -Wl,--no-gc-sections リンカが誤って FIT 周辺機能モジュールで宣言された割り込み関数を破棄することによる GCC リンカ問題を解決します。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイラオプション：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	Rev.1.00
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxxx)

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”」エラーが発生しました。
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）」
— e^2 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 e^2 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。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）」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_qspix_rx module」エラーが発生しました。
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、構成設定が間違っている場合のエラーが発生しました。
A：“r_qspix_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_qspix_rx_config.h”ファイルを確認してください。設定が間違っている場合は、その設定に正しい値を設定してください。詳細は「2.8 コンパイル時の設定」を参照してください。

7. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
テクニカルアップデート／テクニカルニュース
ユーザーズマニュアル：開発ツール
　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。

テクニカルアップデートの対応について

本モジュールに該当するテクニカルアップデートはありません。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Mar.31.21	--	初版発行
1.10	Sep.20.21	19, 22	R_QSPIX_Write_Indirect()と R_QSPIX_Read_Indirect()の Special Notes を更新。
		31	「5.デモプロジェクト」の記載内容見直し
		32	6.1 動作確認環境： Rev.1.10 の表を追加
		ﾌﾟﾛｸﾞﾗﾑ	R_QSPIX_Write_Indirect（）を修正しました
1.20	Dec.07.21	32	6.1 動作確認環境： Rev.1.20 の表を追加
		ﾌﾟﾛｸﾞﾗﾑ	R_QSPIX_Write_Indirect()の複数の連続した呼び出しをサポート。 デモプロジェクトを更新。
1.30	Jul.29.22	32	6.1 動作確認環境： Rev.1.30 の表を追加
		プログラム	デモプロジェクトを更新。
1.40	Mar.16.23	6, 23-24	R_QSPIX_Read_Memory_Map()関数を追加。
		10	2.9 コードサイズ、の説明を追加
		34	6.1 動作確認環境： Rev.1.40 の表を追加
		プログラム	R_QSPIX_Read_Memory_Map()関数を追加。
1.41	Dec.13.23	14	2.14「for 文、while 文、do while 文について」を追加。
		13, 33	「2.13 FIT モジュールの追加方法」、「4 端子設定」から FIT Configurator の説明を削除した。
		35	6.1 動作確認環境： Rev.1.41 の表を追加
		プログラム	WAIT_LOOP コメントを追加。
1.42	Mar.15.25	35	6.1 動作確認環境： Rev.1.42 の表を追加
		プログラム	FIT モジュールの免責事項と著作権を更新。

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ーやマガジンケース、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も、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て、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。

© 2025 Renesas Electronics Corporation. All rights reserved.