RX ファミリ

IRQ モジュール Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT) を使用した割り込み要求 (IRQ) モジュールについて説明します。本モジュールは IRQ を使用して、外部端子割り込みからのイベントを処理するため、統一化されたインタフェースを提供します。以降、本モジュールを IRQ FIT モジュールと称します。

対象デバイス

- RX110、RX111、RX113 グループ
- RX130 グループ、RX13T グループ
- RX140 グループ
- RX230 グループ RX231 グループ RX23T グループ RX23W グループ RX23E-A グループ
- RX24T グループ
- RX24U グループ
- RX64M グループ
- RX651、RX65N グループ
- RX66T グループ
- RX66N グループ
- RX671 グループ
- RX71M グループ
- RX72T グループ
- RX72M グループ
- RX72N 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。

ターゲットコンパイラ

- ルネサスエレクトロニクス製 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環境に関する詳細な内容は、セクション「6.1 動作確認環境 」を参照してください。

目次

1. 概要

1.1 IRQ 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、2.13 FIT モジュールの追加方法 を参照してください。

1.2 IRQ FIT モジュールの概要

本ソフトウェアでは、外部端子割り込みからのイベントを処理するため、統一化されたインタフェースが提供されます。各イベントは IRQ ベクタにマッピングされます。IRQ で割り込みに対応するために必要な処理が R_IRQ_Open() API 関数で行われます。各 MCU に搭載された個数の IRQ ベクタが使用でき、各 API 関数において、個別の IRQ ベクタを認識するための手段となります。本モジュールでは、IRQ API 関数を実行するために必要なベクタ固有の情報を持つデータ構造体が使用されます。データ構造体は使用する各 IRQ に割り当てられます。各 IRQ を使用するときに、1 つのデータ構造体が割り当てられます。各データ構造体は IRQ ハンドルと呼ばれ、それぞれにハンドルポインタがあります。

R_IRQ_Open() 関数によって、IRQ が初期化されるとき、その IRQ のハンドルポインタが呼び出し元に返されます。以降、その他の IRQ API 関数を呼び出すときは、アプリケーションはこのとき選択された IRQ のハンドルポインタを提供する必要があります。API 関数が呼び出されると、その関数はハンドルから IRQ 番号、その IRQ に関連する情報、およびハンドル構造体に含まれる情報を取り出します。

IRQ イベントがトリガされると、ユーザが定義したコールバック関数に制御を渡す割り込み処理が実行されます。コールバック関数は割り込みステートで実行されるため、そのコールバック関数の処理が完了し、プログラムが ISR から復帰するまで、その他の割り込みは禁止されます。IRQ ベクタの初期化後、割り込み処理はユーザアプリケーションによっていつでも許可または禁止にできます。

図 1 : プロジェクト層の例

1.3 IRQ FIT モジュールの使用方法

IRQ モジュールの本来の使用目的は、MCU の GPIO 入力端子の状態変化によってトリガされる割り込みイベントの生成を容易にすることです。ユーザアプリケーションは、イベント検出によって実行されるイベントにコールバック関数を任意で割り当てることができます。

IRQ FIT モジュールをプロジェクトに追加後、そのインストールに合わせてソフトウェアを設定するために、*r_irq_rx_config.h* ファイルを変更する必要があります。設定オプションに関する詳細は 2.8、コンパイル時の設定をご覧ください。

Smart Configurator を使用する場合、少なくとも 1 本の IRQ 端子を選択する必要があります。選択しない場合、ビルドエラーが発生します。r_irq_rx_config.h で、以下の行がコメントアウトされていないことを確認してください。

```
#include "r_irq_rx_pinset.h"
#ifndef R_IRQ_RX_H
#error "Smart Configurator で「IRQ 端子」設定を追加してください。または、FIT コンフィギュレータを使用している場合、以下のマクロのコメントを削除（コメントアウトを解除）してください"
#endif
```

また、以下の行がコメントアウトされていることも確認してください。

```
#define IRQ_PORT_IRQ0_PORT      ('m')
#define IRQ_PORT_IRQ0_BIT       ('n')
…
```

一方、FIT コンフィギュレータを使用する場合、r_irq_rx_config.h でコメントアウト/コメント解除を行うセクションが、Smart Configurator を使用する場合の逆になります

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

IRQ のソースコードで使用される入力端子関連の制御レジスタは、事前に正しく設定されなければなりません。IRQ FIT モジュールでは、これらのレジスタは初期化されませんので、IRQ の API 関数を呼び出す前に、外部で初期化しておく必要があります。通常、これらのレジスタの初期化はシステムの起動時に汎用端子の初期化処理で行われます。

IRQ で使用する割り込みベクタを設定する必要はありません。ビルド時に設定オプションで有効に設定された IRQ について、IRQ FIT モジュールが自動的に設定します。

IRQ を使用するために、実行時にはまず R_IRQ_Open()関数を呼び出して要求される IRQ 番号とその他の必要な設定情報を渡します。これらの処理が完了すると、IRQ はアクティブとなり、入力端子に応答できる状態になります。IRQ イベントの発生で、R_IRQ_Open()呼び出しの引数として指定したコールバック関数が呼び出されます。

R_IRQ_Control()関数には、割り込みのトリガモードと優先レベルの変更に使用できる便利なコマンドが 2 つ用意されています。これらのコマンドによって、必要に応じてそのときの状態に合わせて割り込みイベントの調整が可能になります。R_IRQ_Control()関数が変更を行っている間は、選択された IRQ 番号に対応する割り込みは禁止にされます。

IRQ のその他の設定は、R_IRQ_Open()関数でのみ設定されます。これらの設定は、頻繁に変更することは想定されていません。動作開始後に変更が必要な場合、R_IRQ_Close()関数を呼び出した後に、新規の設定で R_IRQ_Open()関数を再度呼び出さなければなりません。

一般的に IRQ の API 関数の多くがハンドル引数を必要とします。ハンドルは、そのときの動作に対して選択された IRQ 番号を識別するために使用されます。ハンドルは R_IRQ_Open()関数を最初に呼び出したときに取得されます。ユーザはハンドルの格納先アドレスを R_IRQ_Open()に提供する必要があります。以降、その他の IRQ 関数が呼び出されたときに、取得された IRQ 番号のハンドル値を提供します。各 IRQ には個別のハンドルが割り当てられますので、ユーザアプリケーションではそれらを把握しておかなければなりません。

1.3.1 IRQ 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 IRQ 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_irq_rx_if.h”
}
```

1.4 API の概要

表 1.1 に本モジュールに含まれる API 関数を示します。

表 1.1API 関数一覧

関数	関数説明
R_IRQ_Open()	指定された IRQ を使用可能にするために必要なレジスタを初期化します。割り込みを有効にし、その他の API 関数で使用するハンドルを提供します。割り込みイベントに応答して、コールバック関数のポインタを取得します。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。
R_ IRQ_Close()	指定された IRQ および関連する割り込みを禁止します。
R_IRQ_Control()	IRQ に関連する特殊なハードウェア、またはソフトウェアの動作を制御します。
R_IRQ_ReadInput()	指定された IRQ に割り当てられた端子の現在のレベルを読み出します。
R_IRQ_InterruptEnable()	指定された IRQ の ICU 割り込みを許可または禁止します。
R_IRQ_GetVersion()	本モジュールのバージョン番号を返します。
R_IRQ_IRClear()	この関数は指定された IRQ の IR フラグをクリアします。

1.5 制限事項

このドライバは外部端子割り込み（IRQ）にのみ使用できます。

2. API 情報

本 FIT モジュールは、下記の条件で動作を確認しています。

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。

- IRQ
- 割り込み要因として設定できる 1 つ、または複数の GPIO 入力端子

2.2 ソフトウェアの要求

このドライバは以下の FIT モジュールに依存しています。

- ボードサポートパッケージ（r_bsp）v5.20 以上。本モジュールの API を呼び出すときには、事前に関連する I/O ポートが正しく初期化されているものとして動作します。
- デジタルフィルタ機能を使用するには、本モジュールの API を呼び出す前に、外部で周辺クロック（PCLK）を初期化しておく必要があります。

2.3 制限事項

2.3.1 RAM の配置に関する制限事項

FIT では、API 関数のポインタ引数に NULL と同じ値を設定すると、パラメータチェックにより戻り値がエラーとなる場合があります。そのため、API 関数に渡すポインタ引数の値は NULL と同じ値にしないでください。

ライブラリ関数の仕様で NULL の値は 0 と定義されています。そのため、API 関数のポインタ引数に渡す変数や関数が RAM の先頭番地(0x0 番地)に配置されていると上記現象が発生します。この場合、セクションの設定変更をするか、API 関数のポインタ引数に渡す変数や関数が 0x0 番地に配置されないように RAM の先頭にダミーの変数を用意してください。

なお、CCRX プロジェクト(e2 studio V7.5.0)の場合、変数が 0x0 番地に配置されることを防ぐために RAM の先頭番地が 0x4 になっています。GCC プロジェクト(e2 studio V7.5.0)、IAR プロジェクト(EWRX V4.12.1)の場合は RAM の先頭番地が 0x0 になっていますので、上記対策が必要となります。

IDE のバージョンアップによりセクションのデフォルト設定が変更されることがあります。最新の IDE を使用される際は、セクション設定をご確認の上、ご対応ください。

2.4 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。

2.5 使用する割り込みベクタ

トリガモードの設定に一致する IRQ 入力端子の状態変化が発生すると、割り込み要求が生成されます。割り込みが許可されていれば、割り当てたコールバック関数を呼び出す割り込み ISR が実行されます。コールバック関数には、ISR に対して即座に実行したいコードが配置されます。割り込みに関連してコールバックが処理されるため、この間は割り込みが禁止されます。システムで発生するその他の割り込みを落とさないために、コールバック関数の処理はできる限り速く完了するようにしてください。

IRQ 割り込みは、R_IRQ_Open 関数を実行すると有効化されます。

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覧

デバイス	割り込みベクタ
RX110 *1 RX111 *1 RX113 *1 RX130 *1 RX13T *2 RX140*4 RX230 *1 RX231 *1 RX23T *2 RX23W*3 RX23E-A *1 RX24T *1 RX24U *1 RX64M RX651 RX65N RX66T RX66N RX671 RX71M RX72T RX72M RX72N	IRQ0 割り込み（ベクタ番号：64） IRQ1 割り込み（ベクタ番号：65） IRQ2 割り込み（ベクタ番号：66） IRQ3 割り込み（ベクタ番号：67） IRQ4 割り込み（ベクタ番号：68） IRQ5 割り込み（ベクタ番号：69） IRQ6 割り込み（ベクタ番号：70） IRQ7 割り込み（ベクタ番号：71） IRQ8 割り込み（ベクタ番号：72） IRQ9 割り込み（ベクタ番号：73） IRQ10 割り込み（ベクタ番号：74） IRQ11 割り込み（ベクタ番号：75） IRQ12 割り込み（ベクタ番号：76） IRQ13 割り込み（ベクタ番号：77） IRQ14 割り込み (ベクタ番号： 78) IRQ15 割り込み（ベクタ番号：79）

注 1：IRQ0 から IRQ7 までしかありません。
注 2：IRQ0 から IRQ5 までしかありません。
注 3：IRQ0、IRQ1、IRQ4 ～ IRQ7 しかありません。
注 4：IRQ0 ～ IRQ2、IRQ4 ～ IRQ7 しかありません。

2.6 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_ irq_rx_if.h に記載しています。

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ギュレーションオプションの設定は、r_irq_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。

コンフィギュレーションオプション（r_irq_rx_config.h）	
IRQ_CFG_FILT_EN_IRQn ※デフォルト値は“0”	このオプションを 1 にすると、IRQn に対して、デジタルフィルタ機能を有効にします。
IRQ_CFG_FILT_PCLK_IRQn	IRQ 番号 n に対する PCLK デジタルフィルタクロック分周器の設定。定義済みの定数“IRQ_CFG_PCLK_DIVxx”のいずれか 1 つを選択してください。 例： /* Filter sample clock divisor for IRQ 0 = PCLK/64.*/ #define IRQ_CFG_FILT_PLCK_IRQ0　(IRQ_CFG_PCLK_DIV64)
IRQ_CFG_REQUIRE_LOCK ※デフォルト値は“1”	このオプションを 1 にすると、R_IRQ_Open()関数で、関数実行期間に対して BSP ロックが取得されます。これは、内部の状態を再入アクセスから保護するためです。復帰時にロックは解除されます。BSP ロック機能が必要ない場合、あるいはシステムで対応可能な場合はこのオプションは 0 に設定して構いません。
IRQ_CFG_PARAM_CHECKING ※デフォルト値は“BSP_CFG_PARAM_CHECKING_ENABLE”	IRQ API 関数に渡す引数のチェックを有効または無効にできます。システムの動きを速くしたり、コードの容量を小さく抑える必要があるときのために、引数チェックを無効にするオプションが用意されています。 初期設定では、本モジュールは BSP_CFG_PARAM_CHECKING_ENABLE マクロをシステム全体で使用するように設定されています。IRQ_CFG_PARAM_CHECKING を再定義することによって、IRQ モジュールでこの設定は上書きできます。 IRQ モジュールでパラメータチェックを有効するには、IRQ_CFG_PARAM_CHECKING を 1 に、無効にするには 0 に設定します。
IRQ_PORT_IRQn_PORT IRQ_PORT_IRQn_BIT	IRQ FIT モジュールが、R_IRQ_ReadInput()など、ポートに特定の動作を実行できるように、各 IRQ に対してポートおよびポートビットを割り当てる必要があります。 必要に応じて、以下のフォーマットに従って、それらを設定してください。 #define IRQ_PORT_IRQ*_PORT　（‘m’） （m はポート番号） #define IRQ_PORT_IRQ*_BIT（‘n’） （n はビット番号） 注： ここで割り当てたポートは、外部で BSP によって行われたポート設定と一致する必要があります。

2.9 コードサイズ

本モジュールのコード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。掲載した値は、「2.4 サポートされているツールチェーン」の C コンパイラでコンパイルオプションがデフォルト時の参考値です。コンパイルオプションのデフォルトは最適化レベル：2、最適化のタイプ：サイズ優先、データ・エンディアン：リトルエンディアンです。コードサイズは C コンパイラのバージョンやコンパイルオプションにより異なります。

ROM, RAM およびスタックのコードサイズ (1/4)

デバイス	分類		使用メモリ ルネサス製コンパイラ パラメータチェック処理あり	使用メモリ ルネサス製コンパイラ パラメータチェック処理なし	備考
RX110, RX111	ROM	1 IRQ	777 バイト	527 バイト	
		8 IRQs	1377 バイト	1127 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		48 バイト	46 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX130, RX230, RX24T, RX24U	ROM	1 IRQ	777 バイト	527 バイト	
		8 IRQs	1366 バイト	1116 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		48 バイト	36 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。

ROM、RAM およびスタックのコードサイズ (2/4)					
デバイス	カテゴリ		使用メモリ ルネサス製コンパイラ		備考
			パラメータチェック処理あり	パラメータチェック処理なし	
RX23T	ROM	1 IRQ	767 バイト	521 バイト	
		6 IRQ	1207 バイト	961 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	10 バイト	4 バイト	
		6 IRQ	30 バイト	24 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		48 バイト	36 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX23W	ROM	1 IRQ	696 バイト	448 バイト	
		6 IRQs	986 バイト	738 バイト	ROM サイズは、プログラム、定数、および初期化データの合計です。
	RAM	1 IRQ	12 バイト	4 バイト	
		6 IRQs	32 バイト	24 バイト	RAM サイズは、未初期化データ、データ、スタック、その他の合計です。
	最大のスタック使用量		84 バイト	76 バイト	最大スタック使用量は、周辺 API の最大サイズと周辺割り込みの最大スタックサイズの合計です。
RX64M、RX65N、RX66T、RX71M	ROM	1 IRQ	892 バイト	640 バイト	
		16 IRQ	2180 バイト	1928 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		48 バイト	40 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。

ROM、RAM およびスタックのコードサイズ (3/4)					
デバイス	カテゴリ		使用メモリ ルネサス製コンパイラ パラメータチェック処理あり	使用メモリ ルネサス製コンパイラ パラメータチェック処理なし	備考
RX113、RX231	ROM	1 IRQ	777 バイト	529 バイト	
		8 IRQ	1366 バイト	1118 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		48 バイト	36 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX72T	ROM	1 IRQ	892 バイト	640 バイト	
		16 IRQ	2181 バイト	1929 bytes	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		48 バイト	40 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX72M	ROM	1 IRQ	879 バイト	620 バイト	
		16 IRQ	2160 バイト	1901 bytes	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	
	最大のスタック使用量		84 バイト	80 バイト	

ROM、RAM およびスタックのコードサイズ (4/4)					
デバイス	カテゴリ		使用メモリ ルネサス製コンパイラ パラメータチェック処理あり	使用メモリ ルネサス製コンパイラ パラメータチェック処理なし	備考
RX13T	ROM	1 IRQ	695 バイト	440 バイト	
		6 IRQ	985 バイト	730 バイト	
	RAM	1 IRQ	10 バイト	4 バイト	
		6 IRQ	30 バイト	24 バイト	
	最大のスタック使用量		84 バイト	80 バイト	
RX66N	ROM	1 IRQ	819 バイト	560 バイト	
		16 IRQs	2100 バイト	1841 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQs	80 バイト	64 バイト	
	最大のスタック使用量		72 バイト	72 バイト	
RX72N	ROM	1 IRQ	819 バイト	560 バイト	
		16 IRQs	2100 バイト	1841 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQs	80 バイト	64 バイト	
	最大のスタック使用量		72 バイト	72 バイト	
RX23E-A	ROM	1 IRQ	786 バイト	531 バイト	
		8 IRQs	1385 バイト	1130 バイト	
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	40 バイト	32 バイト	
	最大のスタック使用量		76 バイト	76 バイト	
RX671	ROM	1 IRQ	871 バイト	599 バイト	
		16 IRQs	2141 バイト	1869 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQs	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)

ROM、RAM およびスタックのコードサイズ (4/4)					
デバイス	カテゴリ		使用メモリ ルネサス製コンパイラ パラメータチェック処理あり	使用メモリ ルネサス製コンパイラ パラメータチェック処理なし	備考
	最大のスタック使用量		72 バイト	68 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX140	ROM	1 IRQ	855 バイト	571 バイト	
		8 IRQs	1362 バイト	1078 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	36 バイト	28 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		72 バイト	64 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。

ROM、RAM およびスタックのコードサイズ (1/4)					
デバイス	カテゴリ		使用メモリ GCC パラメータチェック処理あり	使用メモリ GCC パラメータチェック処理なし	備考
RX110、RX111	ROM	1 IRQ	1532 バイト	1140 バイト	
		8 IRQ	2592 バイト	2200 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX130、RX230、RX24T、RX24U	ROM	1 IRQ	1532 バイト	1140 バイト	
		8 IRQ	2592 バイト	2200 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX13T	ROM	1 IRQ	1604 バイト	1204 バイト	
		6 IRQ	2360 バイト	1960 バイト	
	RAM	1 IRQ	12 バイト	4 バイト	
		6 IRQ	32 バイト	24 バイト	
	最大のスタック使用量		-	-	

ROM、RAM およびスタックのコードサイズ (2/4)					
デバイス	カテゴリ		使用メモリ GCC パラメータチェック処理あり	使用メモリ GCC パラメータチェック処理なし	備考
RX23T	ROM	1 IRQ	1532 バイト	1140 バイト	
		6 IRQ	4048 バイト	3656 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		6 IRQ	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX64M、RX65N、RX66T、RX71M	ROM	1 IRQ	1772 バイト	1380 バイト	
		16 IRQ	2288 バイト	1904 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	8 バイト	4 バイト	
		16 IRQ	28 バイト	24 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX66N	ROM	1 IRQ	596 バイト	164 バイト	
		16 IRQ	2872 バイト	2440 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	
	最大のスタック使用量		-	-	

ROM、RAM およびスタックのコードサイズ (3/4)					
デバイス	カテゴリ		使用メモリ GCC パラメータチェック処理あり	使用メモリ GCC パラメータチェック処理なし	備考
RX113、RX231	ROM	1 IRQ	1540 バイト	1148 バイト	
		8 IRQ	2600 バイト	2216 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX72T	ROM	1 IRQ	1772 バイト	1380 バイト	
		16 IRQ	4048 バイト	3656 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX72M	ROM	1 IRQ	1892 バイト	1484 バイト	
		16 IRQ	4168 バイト	3760 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	
	最大のスタック使用量		-	-	
RX72N	ROM	1 IRQ	596 バイト	164 バイト	
		16 IRQ	2872 バイト	2440 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	
	最大のスタック使用量		-	-	

ROM、RAM およびスタックのコードサイズ (4/4)					
デバイス	カテゴリ		使用メモリ GCC パラメータチェック処理あり	使用メモリ GCC パラメータチェック処理なし	備考
RX23E-A	ROM	1 IRQ	1652 バイト	1236 バイト	
		8 IRQ	2704 バイト	2296 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX671	ROM	1 IRQ	1912 バイト	1480 バイト	
		16 IRQ	4172 バイト	3748 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。

ROM、RAM およびスタックのコードサイズ (4/4)					
デバイス	カテゴリ		使用メモリ GCC パラメータチェック処理あり	使用メモリ GCC パラメータチェック処理なし	備考
RX140	ROM	1 IRQ	1784 バイト	1328 バイト	
		8 IRQ	2836 バイト	2388 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		-	-	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。

ROM、RAM およびスタックのコードサイズ (1/3)					
デバイス	カテゴリ		使用メモリ IAR コンパイラ パラメータチェック処理あり	使用メモリ IAR コンパイラ パラメータチェック処理なし	備考
RX110、RX111	ROM	1 IRQ	1120 バイト	792 バイト	
		8 IRQ	1676 バイト	1348 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		152 バイト	152 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX130、RX230、RX24T、RX24U	ROM	1 IRQ	1120 バイト	792 バイト	
		8 IRQ	1672 バイト	1344 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		152 バイト	152 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX13T	ROM	1 IRQ	1112 バイト	784 バイト	
		6 IRQ	1509 バイト	1184 バイト	
	RAM	1 IRQ	12 バイト	4 バイト	
		6 IRQ	32 バイト	24 バイト	
	最大のスタック使用量		104 バイト	104 バイト	

ROM、RAM およびスタックのコードサイズ (2/3)					
デバイス	カテゴリ		使用メモリ IAR コンパイラ パラメータチェック処理あり	使用メモリ IAR コンパイラ パラメータチェック処理なし	備考
RX23T	ROM	1 IRQ	1112 バイト	784 バイト	
		6 IRQ	1504 バイト	1176 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	8 バイト	4 バイト	
		6 IRQ	28 バイト	24 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		152 バイト	152 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX64M、RX65N、RX66T、RX71M	ROM	1 IRQ	1338 バイト	1039 バイト	
		16 IRQ	2523 バイト	2289 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		168 バイト	168 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX66N	ROM	1 IRQ	1278 バイト	937 バイト	
		16 IRQ	2463 バイト	2128 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	
	最大のスタック使用量		88 バイト	88 バイト	

ROM、RAM およびスタックのコードサイズ (3/3)					
デバイス	カテゴリ		使用メモリ IAR コンパイラ パラメータチェック処理あり	使用メモリ IAR コンパイラ パラメータチェック処理なし	備考
RX113、RX231	ROM	1 IRQ	1120 バイト	792 バイト	
		8 IRQ	1674 バイト	1346 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		152 バイト	152 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX72T	ROM	1 IRQ	1319 バイト	984 バイト	
		16 IRQ	2504 バイト	2167 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		168 バイト	168 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX72M	ROM	1 IRQ	1320 バイト	992 バイト	
		16 IRQ	2505 バイト	2174 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	
	最大のスタック使用量		172 バイト	172 バイト	
RX72N	ROM	1 IRQ	1278 バイト	937 バイト	
		16 IRQ	2463 バイト	2128 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQ	80 バイト	64 バイト	
	最大のスタック使用量		88 バイト	88 バイト	

ROM、RAM およびスタックのコードサイズ (3/3)					
デバイス	カテゴリ		使用メモリ IAR コンパイラ パラメータチェック処理あり	使用メモリ IAR コンパイラ パラメータチェック処理なし	備考
RX23E-A	ROM	1 IRQ	1054 バイト	718 バイト	
		8 IRQ	1606 バイト	1272 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	40 バイト	32 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		76 バイト	76 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。
RX671	ROM	1 IRQ	1301 バイト	969 バイト	
		16 IRQs	2486 バイト	2155 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQs	80 バイト	64 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		76 バイト	76 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。

ROM、RAM およびスタックのコードサイズ (3/3)					
デバイス	カテゴリ		使用メモリ IAR コンパイラ パラメータチェック処理あり	使用メモリ IAR コンパイラ パラメータチェック処理なし	備考
RX140	ROM	1 IRQ	1168 バイト	801 バイト	
		8 IRQ	1646 バイト	1279 バイト	ROM サイズは、次の式で計算できます。 1 チャネルの場合の ROM サイズ + (58 バイト x 追加チャネル数)
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQ	36 バイト	28 バイト	RAM サイズは、次の式で計算できます。 1 チャネルの場合の RAM サイズ + (4 バイト x 追加チャネル数)
	最大のスタック使用量		76 バイト	76 バイト	ネストした割り込みは禁止されているので、1 個のチャネルを使用する場合の最大値を掲載します。

2.10 引数

API 関数の引数である構造体を示します。この構造体は、API 関数のプロトタイプ宣言とともに r_irq_rx_if.h に記載されています。

2.10.1 特殊なデータ型

強力な型チェック機能を提供しエラーを減少させるため、API 関数で使用されるパラメータの多くが、提供された型定義の引数を要求します。使用可能な値は、*r_irq_rx_if.h* ファイルに定義されます。

IRQ 番号の列挙型

型: irq_number_t

マクロ: IRQ_NUM_n

値(n): 0～7（全 MCU）、8～15（IRQ を 16 個備えた MCU）

例: IRQ_NUM_2

IRQ 制御コマンドコード

型: irq_cmd_t

値: IRQ_CMD_SET_PRIO、IRQ_CMD_SET_TRIG

IRQ 割り込み優先レベルの設定

型: irq_prio_t

マクロ: IRQ_PRI_n

値(n): 0～15

例: IRQ_PRI_3

IRQ トリガモードの設定

型: irq_trigger_t

値: IRQ_TRIG_LOWLEV、IRQ_TRIG_FALLING、IRQ_TRIG_RISING、IRQ_TRIG_BOTH_EDGE

ハンドル

型: irq_handle_t

値: ユーザによってこのハンドル型を格納するためのポインタが提供されます。ハンドル値は、R_IRQ_Open 関数によって自動的に割り当てられます。

2.11 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_irq_rx_if.h で記載されています。

戻り値の型: irq_err_t

値:	説明
IRQ_SUCCESS	関数の処理を、エラーを出さずに完了しました。
IRQ_ERR_BAD_NUM	無効な IRQ 番号が渡されました。
IRQ_ERR_NOT_OPENED	IRQ が起動していません。関数を完了できません。
IRQ_ERR_NOT_CLOSED	IRQ は前回の処理から起動したままです。
IRQ_ERR_UNKNOWN_CMD	制御コマンドが認識されません。
IRQ_ERR_INVALID_ARG	パラメータに対して無効な引数です。
IRQ_ERR_INVALID_PTR	null ポインタ受信; 要求される引数がありません。
IRQ_ERR_LOCK	ロックの手順に失敗しました。

2.12 コールバック関数

本モジュールでは、IRQ 割り込みが発生したタイミングで。ユーザが設定したコールバック関数を呼び出します。

コールバック関数は、「2.10 引数」に記載された構造体メンバ “void (*const pcallback)(void *pargs))” に、ユーザの関数のアドレスを格納することで設定されます。コールバック関数が呼び出されるとき、す定数が格納された変数が、引数として渡されます。

引数の型は void ポインタ型で渡されるため、コールバック関数の引数は下記の例を参考に void 型のポインタ変数としてください。

コールバック関数内部で値を使うときはキャストして値を使用してください。

```
void my_irq_callback(void * pdata)
{
    irq_number_t  my_triggered_irq_number;

    my_triggered_irq_number = *((irq_number_t *)pdata);
    ...
}
```

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)、(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。

2.14 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_IRQ_Open()

この関数は関連する IRQ レジスタの初期化を行い、割り込みを有効にし、その他の API 関数で使用するハンドルを提供します。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

```
irq_err_t       R_IRQ_Open (
        irq_number_t    irq_number,
          irq_trigger_t   trigger,
          irq_prio_t      priority,
                  irq_handle_t    *phandle,
        void (*const pcallback)(void *pargs)
)
```

Parameters

irq_number_t irq_number
初期化対象の IRQ 番号

irq_trigger_t trigger
トリガの型に使用する列挙型: Low レベル、立ち上がりエッジ、立ち下がりエッジ、両エッジ

irq_prio_t priority
IRQ の割り込み優先レベルに使用する列挙型

irq_handle_t phandle
IRQ ハンドルのロケーションへのポインタ。 ハンドル値はこの関数で設定されます。

pcallback
割り込みから呼び出される関数へのポインタ。

Return Values

```
[IRQ_SUCCESS]          /*成功; IRQ が初期化されました。                          */
[IRQ_ERR_BAD_NUM]      /* IRQ 番号が無効または使用不可です。                     */
[IRQ_ERR_NOT_CLOSED]   /* IRQ は現在動作中です。R_IRQ_Close() を実行してください。 */
[IRQ_ERR_INVALID_PTR]  /* phandle ポインタが null です。                          */
[IRQ_ERR_INVALID_ARG]  /*無効な引数値が渡されました。                            */
[IRQ_ERR_LOCK]         /*ロック機能が取得できませんでした。                      */
```

Properties

ファイル r_irq_rx_if.h にプロトタイプ宣言されています。

Description

本 Open 関数は IRQ が動作できるように準備します。Open 関数の完了後、IRQ が有効になり、割り込みへの対応が可能となります。他の IRQ API 関数を呼び出す前に、本関数を呼び出す必要があります。本関数が問題なく完了すると、選択された IRQ は「動作中」に設定されます。以降は、R_IRQ_Close()を呼び出して IRQ を終了しない限りは、同一 IRQ に対して本関数を呼び出さないでください。

Example

```
/* 他の IRQ API 関数へのアクセスに使用されるハンドルを割り当てる。*/
irq_handle_tmy_handle;
irq_err_t   result;

/* IRQ0 を使用するための処理を行う。割り込みトリガは立ち下がりエッジ、割り込み優先レベルは
3。*/
result = R_IRQ_Open (IRQ_NUM_0,
         IRQ_TRIG_FALLING,
                     IRQ_PRI_3,
                     &my_handle,
                     &my_callback);

if(IRQ_SUCCESS != result)
{
    // エラー処理
}
```

Special Notes:

なし

R_IRQ_Control()

この関数は IRQ 関連の特殊なハードウェア、またはソフトウェアの動作を制御します。

Format

```
irq_err_t       R_IRQ_Control (
        irq_handle_t  const     handle,
      irq_cmd_t                 const cmd,
        void            *pcmd_data
)
```

Parameters

irq_handle_t const handle

IRQ のハンドル

irq_cmd_t cmd

コマンドコードの列挙型

IRQ_CMD_SET_PRIO - 割り込み優先レベルを変更します。

IRQ_CMD_SET_TRIG - 割り込みのトリガモードを変更します。

pcmd_data

コマンドデータ構造体のパラメータへのポインタです。これは、処理を完了させるために必要なコマンドに特定のデータの位置を参照するのに使用される void 型のポインタです。

Return Values

[IRQ_SUCCESS]	*/*成功;コマンドが完了しました。*	**/*
[IRQ_ERR_NOT_OPENED]	*/* IRQ が起動していません。R_IRQ_Open() を実行してください。*	**/*
[IRQ_ERR_BAD_NUM]	*/* IRQ 番号が無効または使用不可です。*	**/*
[IRQ_ERR_UNKNOWN_CMD]	*/* IRQ 番号が無効または使用不可です。*	**/*
[IRQ_ERR_INVALID_PTR]	*/* Control コマンドが認識されません。*	**/*
[IRQ_ERR_INVALID_ARG]	*/* pcmd_data ポインタ、またはハンドルが NULL です。*	**/*
[IRQ_ERR_LOCK]	*/* ロック機能が取得できませんでした。*	**/*

Properties

ファイル r_irq_rx_if.h にプロトタイプ宣言されています。

Description

本関数は IRQ 関連の特殊なハードウェア、またはソフトウェアの動作を制御します。

この関数は IRQ ハンドルを取得して、以下のものを識別します。

- 選択された IRQ
- 実行する動作を選択するための列挙型のコマンド値
- 動作を完了するために必要な情報やデータを含むロケーションへの void 型ポインタ。

本ポインタは、r_irq_rx_if.h で提供される適切な型を使って、コマンドに合うように呼び出し元によって型変換された格納先を示す必要があります。

Example

```
/* トリガモードを立ち上がりエッジに変更する。 */
irq_trigger_t  my_trig_mode = IRQ_TRIG_RISING;

result = R_IRQ_Control(my_handle, IRQ_CMD_SET_TRIG, &my_trig_mode);

/* 割り込み優先レベルを変更する。 */
irq_prio_t my_priority = IRQ_PRI_10;

result = R_IRQ_Control(my_handle, IRQ_CMD_SET_PRIO, &my_priority);
```

Special Notes:

なし

R_IRQ_Close()

この関数はハンドルによって指定された IRQ を無効にします。

Format

```
irq_err_t       R_IRQ_Close (
        irq_handle_t      handle
)
```

Parameters

irq_handle_t handle
IRQ のハンドル

Return Values

[IRQ_SUCCESS]	*/* 成功; IRQ を終了しました。 */*	*/*
[IRQ_ERR_NOT_OPENED]	*/* IRQ は起動していないため、終了処理は行われません。 */*	*/*
[IRQ_ERR_BAD_NUM]	*/* IRQ 番号が無効または使用不可です。 */*	*/*
[IRQ_ERR_INVALID_PTR]	*/* 要求されたポインタの引数が NULL です。 */*	*/*

Properties

ファイル r_irq_rx_if.h にプロトタイプ宣言されています。

Description

本関数は、ポートへの割り当てをクリアして IRQ を開放し、関連する割り込みを禁止にします。IRQ のハンドルは、その IRQ が動作中でないことを示すステートに変更され、その IRQ は R_IRQ_Open 関数で再度有効にされるまで使用できません。ステートが動作中でない IRQ に対して本関数が呼び出された場合、エラーコードが返されます。

Example

```
/* */
irq_err_t result;

result = R_IRQ_Close(my_handle);
```

Special Notes:

なし

R_IRQ_ReadInput()

この関数は指定された IRQ に割り当てられた端子のレベルを読み出します。

Format

```
irq_err_t       R_IRQ_ReadInput (
        irq_handle_t const      handle,
        uint8_t                 *plevel
)
```

Parameters

irq_handle_t const handle
IRQ のハンドル

uint8_t plevel
入力端子の状態を返すロケーションへのポインタ

Return Values

```
[IRQ_SUCCESS]           /* 成功; 動作が完了しました。                                   */
[IRQ_ERR_NOT_OPENED]    /* IRQ が起動していません。R_IRQ_Open() を実行してください。    */
[IRQ_ERR_BAD_NUM]       /* IRQ 番号が無効または使用不可です。                           */
[IRQ_ERR_INVALID_PTR]   /* plevel のデータポインタ、またはハンドルが NULL です。        */
```

Properties

ファイル r_irq_rx_if.h にプロトタイプ宣言されています。

Description

本関数は指定された IRQ に割り当てられた端子のレベルを読み出します。このレベルは読み出しを行った時点のレベルで、割り込みがトリガされた時点のレベルとは異なる場合があります。この関数は、スイッチによって割り込みがトリガされ、デバウンスを待つ必要がある場合などに使用されます。

Example

```
/* 現在の入力の論理レベルを確認する */
uint8_t irq_pin_level;

result = R_IRQ_ReadInput(my_handle, (uint8_t*)&irq_pin_level);
```

Special Notes:

なし

R_IRQ_InterruptEnable()

この関数は指定された IRQ に対して ICU 割り込みを許可、または禁止します。

Format

```
irq_err_t       R_IRQ_InterruptEnable (
        irq_handle_t const      handle,
        bool                    enable
)
```

Parameters

irq_handle_t const handle
IRQ のハンドル

bool enable
true＝割り込み許可
false＝割り込み禁止

Return Values

```
[IRQ_SUCCESS]           /* 成功; 動作が完了しました。                                   */
[IRQ_ERR_NOT_OPENED]    /* IRQ が起動していません。R_IRQ_Open() を実行してください。    */
[IRQ_ERR_BAD_NUM]       /* IRQ 番号が無効または使用不可です。                           */
[IRQ_ERR_INVALID_PTR]   /*ハンドルがNULL です。                                        */
```

Properties

ファイル r_irq_rx_if.h にプロトタイプ宣言されています。

Description

本関数はハンドルの引数を使って、指定された IRQ の ICU 割り込みを許可、または禁止にします。この関数は頻繁に呼び出される可能性があり、素早く実行されることが想定されます。

Example

```
irq_err_t result;

/* 割り込み許可 */
result = R_IRQ_InterruptEnable (my_handle, true);

/* 割り込み禁止 */
result = R_IRQ_InterruptEnable (my_handle, false);
```

Special Notes:

なし

R_IRQ_GetVersion()

この関数は実行時に本モジュールのバージョンを返します。

Format

uint32_t R_IRQ_GetVersion (void)

Parameters

なし

Return Values

メジャーバージョンとマイナーバージョンからなる 32 ビット値で示されるバージョン番号

Properties

ファイル r_irq_rx_if.h にプロトタイプ宣言されています。

Description

この関数は本モジュールのバージョンを返します。バージョン番号は符号化され、最上位の 2 バイトがメジャーバージョン番号を、最下位の 2 バイトがマイナーバージョン番号を示しています。

Example

```
/* バージョン番号を取り出し、取り出した番号を文字列に変換する*/

uint32_t   version, version_high, version_low;
char       version_str[9];

version = R_IRQ_GetVersion();

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

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

Special Notes:

なし

R_IRQ_IRClear ()

この関数は指定された IRQ の IR フラグをクリアします。

Format

```
irq_err_t       R_IRQ_IRClear (
        irq_handle_t const          handle,
)
```

Parameters

irq_handle_t const handle
IRQ のハンドル

Return Values

```
[IRQ_SUCCESS]           /* 成功; 動作が完了しました。                                  */
[IRQ_ERR_NOT_OPENED]    /* IRQ が起動していません。R_IRQ_Open() を実行してください。   */
[IRQ_ERR_BAD_NUM]       /* IRQ 番号が無効または使用不可です。                          */
[IRQ_ERR_INVALID_PTR]   /*ハンドルがNULL です。                                       */
```

Properties

ファイル r_irq_rx_if.h にプロトタイプ宣言されています。

Description

この関数は引数で指定された IRQ の IR フラグをクリアします。

Example

```
/* IR フラグをクリア */

irq_err_t result;

result = R_IRQ_IRClear (my_handle);
```

Special Notes:

IR フラグはエッジ検出が使用されるときのみクリアされます。

割り込み要求先が DTC または DMAC のとき、IR フラグに 0 を書かないでください。

4. 端子設定

IRQ 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。

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。

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 irq_demo_rskrx113、irq_demo_rskrx231、irq_demo_rskrx64m、irq_demo_rskrx71m、irq_demo_rskrx65n、irq_demo_rskrx65n_2m、irq_demo_rskrx72m、irq_demo_rskrx671、irq_demo_rskrx113_gcc、irq_demo_rskrx231_gcc、irq_demo_rskrx64m_gcc、irq_demo_rskrx71m_gcc、irq_demo_rskrx65n_gcc、irq_demo_rskrx65n_2m_gcc、irq_demo_rskrx72m_gcc、irq_demo_rskrx671_gcc

irq_demo_rskrx113、irq_demo_rskrx231、irq_demo_rskrx64m、irq_demo_rskrx71m、irq_demo_rskrx65n および irq_demo_rskrx65n_2m は、それぞれ RSKRX113、RSKRX231、RSKRX64M、RSKRX71M、RSKRX65N、RSKRX65N-2MB および RSKRX72M のデモボードに対応した IRQ FIT モジュールのデモプログラムです。これらのプログラムで、R_IRQ_Open API 呼び出しを使って、割り込みの入力としてポートビットを設定する方法、および割り込みを使用するためのコールバック関数の設定方法をデモします。また、R_IRQ_Control API 呼び出しを使って、割り込みのトリガ条件を再設定する方法、R_IRQ_ReadInput API 呼び出しの使用方法、コールバックの引数を逆参照して、割り込み番号を取得する方法についてもデモします。デモでは割り込みとして IRQ2（RX231 では IRQ4、RX65N では IRQ9、RX65N-2MB では IRQ13、RX72M では IRQ12、RX671 では IRQ10）が選択され、SW2 の押下の検出に使用されます。

これらのプロジェクトのデモの動作は同じで、コードがコンパイルされ、対象のボードにダウンロードして実行されると、SW2 を押して、IRQ2（RX231 では IRQ4、RX65N では IRQ9、RX65N-2MB では IRQ13、RX72M では IRQ12、RX671 では IRQ10）割り込みを発生させることができます。SW2 が押下されると、立ち下がりエッジに応じて LED3 が点灯し、SW2 がリリースされると、立ち上がりエッジに応じて LED3 が消灯します。

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 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.3.90)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.03.00 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 8.3.0.202004 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 -WI,--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。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.3.90
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxxx)

表 6.2 動作確認環境 (Rev.3.80)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.03.00 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 8.3.0.202004 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 -WI,--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。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.3.80
使用ボード	Renesas Starter Kit+ for RX140 (product No.: RTK5RX140xxxxxxxxx) Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxxx) Renesas Starter Kit+ for RX65N (product No.: RTK50565NCxxxxxBE) Renesas Starter Kit+ for RX72N (product No.: RTK5572Nxxxxxxxxxx)

表 6.3 動作確認環境 (Rev.3.70)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.2021-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.03.00 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 8.3.0.202004 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 -WI,--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。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.20.3 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	Rev.3.70
使用ボード	Renesas Starter Kit+ for RX671 (product No.: RTK55671xxxxxxxxx)

表 6.4 動作確認環境 (Rev.3.60)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.8.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.02.00 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 8.3.0.201904 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 -WI,--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。
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.3.60
使用ボード	Renesas Starter Kit+ for RX72M (product No.: RTK5572Mxxxxxxxxxx) Renesas Starter Kit+ for RX65N-2MB (product No.: RTK50565N2CxxxxxBR) Renesas Starter Kit+ for RX65N (product No.: RTK50565NCxxxxxBE) Renesas Starter Kit+ for RX64M (product No.: RTK50564Mxxxxxxxx) Renesas Starter Kit+ for RX71M (product No.: RTK50571Mxxxxxxxx) Renesas Starter Kit+ for RX113 (product No.: RTK505113xxxxxxxx) Renesas Starter Kit+ for RX231 (product No.: RTK505231xxxxxxxx)

表 6.5 動作確認環境 (Rev.3.50)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.02.00 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 8.3.0.201904 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 -WI,--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。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.50
使用ボード	Renesas Solution Starter Kit+ for RX23E-A (product No.: RTK0ESXBxxxxxxxxxx)

表 6.6 動作確認環境 (Rev.3.40)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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
	GCC for Renesas RX 4.8.4.201902 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 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 -WI,--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。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.40
使用ボード	Renesas Starter Kit+ for RX72N (product No.: RTK5572Nxxxxxxxxxx).

表 6.7 動作確認環境 (Rev.3.30)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.7.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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
	GCC for Renesas RX 4.8.4.201902 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 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 -WI,--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。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.30
使用ボード	RX13T CPU Card (product No.: RTK0EMXA10C00000BJ)

表 6.8 動作確認環境 (Rev.3.20)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.5.0 IAR Embedded Workbench for Renesas RX 4.12.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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
	GCC for Renesas RX 4.8.4.201902 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 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 -WI,--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。
エンディアン	IAR C/C++ Compiler for Renesas RX version 4.12.1 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	Rev.3.20
使用ボード	Renesas Starter Kit+ for RX72M（型名：RTK5572Mxxxxxxxxxx）

表 6.9 動作確認環境 (Rev.3.10)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio V.7.5.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.3.10
使用ボード	Renesas Solution Starter Kit for RX23W（型名：RTK5523Wxxxxxxxxx

表 6.10 動作確認環境 (Rev.3.00)

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 V.7.4.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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
	GCC for Renesas RX 4.8.4.201803 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 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 -WI,--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 4.10.1 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	Rev.3.00
使用ボード	Renesas Starter Kit+ for RX65N-2MB（型名：RTK50565Nxxxxxxxxx）

表 6.11 作確認環境 (Rev.2.40)

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 V.7.3.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.2.40
使用ボード	Renesas Starter Kit for RX72T（型名：RTK5572Txxxxxxxxxx)

表 6.12 動作確認環境 (Rev.2.31)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.7.3.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.2.31
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.13 動作確認環境 (Rev.2.30)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio V.7.0.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package for RX Family V3.00.00
	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	Rev.2.30
使用ボード	Renesas Starter Kit for RX66T（型名：RTK50566T0SxxxxxBE) Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.14 動作確認環境 (Rev.2.21)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e² studio Version 6.0.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.2.21
使用ボード	Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

表 6.15 動作確認環境 (Rev.2.20)

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 6.0.0
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler Package 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	Rev.2.20
使用ボード	Renesas Starter Kit+ for RX 65N-2MB（型名：RTK50565N2CxxxxxBR） Renesas Starter Kit+ for RX130-512KB（型名：RTK5051308CxxxxxBR）

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)」
- 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。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_irq_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「コンフィグ設定が間違っている場合のエラーメッセージ」エラーが発生します。

A：“r_irq_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_irq_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「2.8 コンパイル時の設定」を参照してください。

(4) Q：割り込みが発生しません。

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。

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
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／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
RX ファミリ CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
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

本モジュールは以下のテクニカルアップデートの内容を反映しています。
なし

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.11.15	--	初版発行
1.20	2014.04.10	1 4	テスト済み MCU リストに RX110 を追加。 セクション 1.2、1.3 に、コールバック関数の使用方法を追加。 Colophon を 4.0 に更新。
1.30	2014.07.23	複数	サポート対象 MCU に RX64M を追加、コードサイズ情報を追加、書式を更新。
1.40	2015.01.08	複数 –	最新のテンプレートに更新。 RX113 グループのサポートを追加。
1.50	2015.03.24	–	RX71M グループのサポートを追加。
1.60	2015.06.30	–	RX231 グループのサポートを追加。
1.70	2015.09.30	– – 7	RX23T グループのサポートを追加。 戻り値の誤植を修正。 IRQ_ERR_NOT_OPEN -> IRQ_ERR_NOT_OPENED 「2.8 コードサイズ」で、RX23T グループに対応するコードサイズを更新。
1.80	2015.10.01	– 7	RX130 グループのサポートを追加。 「2.8 コードサイズ」で、RX130 グループに対応するコードサイズを更新。
1.90	2015.12.01	– 1, 9 5 7 17	RX230 と RX24T 各グループのサポートを追加。 『ボードサポートパッケージモジュール Firmware Integration Technology』アプリケーションノートのドキュメント番号を変更。 セクション 2 の説明を変更。 「2.8 コードサイズ」で、RX230 と RX24T の各グループに対応するコードサイズを更新。 「4. デモプロジェクト」を追加。
1.91	2016.06.15	17 18	「4. デモプロジェクト」に RSKRX64M を追加。 「テクニカルアップデートの対応について」を追加。
2.00	2016.10.01	– 7, 8 18	RX65N グループのサポートを追加。 「2.8 コードサイズ」で、コードサイズの表形式を変更。 「2.8 コードサイズ」で、RX65N グループに対応するコードサイズを更新。 「4 ピン設定」を追加。
2.10	2017.02.28	— 7	RX24U グループのサポートを追加。 「2.8 コードサイズ」で、RX24U グループに対応するコードサイズを更新。
2.20	2017.07.21	— 5 6 11 19	RX130-512KB と RX65N-2MB のサポートを追加。 「2.4 対応ツールチェーン」に RXC v2.07.00 を追加。 「2.5 割り込みベクタ」を追加。 「2.12 プロジェクトへの FIT モジュールの追加」を追加。 「4 ピン設定」を追加。
2.21	2017.10.31	20 20 21	「5. デモプロジェクト」に RSKRX65N と RSKRX65N-2MB を追加。 「5.3 デモのダウンロード方法」を追加。 「6. 付録」を追加。

2.30	2018.09.28	– 1, 6 4 8 10 26	r_irq_rx_config.h と r_irq_rx_private.h 内でマクロの構造を変更。 RX66T のサポートを追加 Smart Configurator または FIT コンフィギュレータを使用する場合にマクロのコメント化/コメントアウト解除の方法に関する手順を追加。 r_irq_rx_config.h の「コンフィギュレーションオプション」に関する表で、IRQ_CFG_USE_IRQn_0 を削除、IRQ_PORT_IRQn を IRQ_PORT_IRQn_PORT に変更、IRQ_PORT_BIT_IRQn を IRQ_PORT_IRQn_BIT に変更。 RX66T に対応するコードサイズを追加。 「6.1 動作確認環境」に、Rev.2.30 に対応する表を追加
2.31	2018.11.16	– 26	XML 内にドキュメント番号を追加。 Renesas Starter Kit+ for RX66T の型名を変更。 Rev.2.31 に対応する表を追加
2.40	2019.02.01	– 1、7、11 16-23 26	RX72T グループのサポートを追加。 RX72T グループのサポートを追加。 各 API 関数で「Reentrant」の説明を削除。 「6.1 動作確認環境」Rev.2.40 に対応する表を追加。
3.00	2019.05.20	– 1 6 7 9 32 36 プログラム	以下のコンパイラをサポート。 - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX RX210 と RX63N の更新終了につき、「対象デバイス」からこれらのデバイスを削除。 「ターゲットコンパイラ」のセクションを追加。 関連ドキュメントを削除。 「2.2 ソフトウェアの要求」r_bsp v5.20 以上が必要 「2.4 使用する割り込みベクタ」：RX210 と RX63N を削除。 「2.8 コードサイズ」セクションを更新。 表 6.1「動作確認環境」： Rev.3.00 に対応する表を追加。 「Web サイトおよびサポート」のセクションを削除。 GCC と IAR コンパイラに関して、以下を変更。 1. R_IRQ_GetVersion 関数のインライン展開を削除。 2. NOP を BSP の固有関数で置き換えた。 3. 割り込み関数の宣言を、BSP のマクロ定義で置き換えた。
3.10	2019.06.28	1、7 10 32 プログラム	RX23W のサポートを追加。 RX23W に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.10 に対応する表を追加。 RX23W のサポートを追加。
3.20	2019.08.15	1, 7 11, 14,17 32 プログラム	RX72M のサポートを追加。 RX72M に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.20 に対応する表を追加。 表 6.2：RX23W ボード名変更。 RX72M のサポートを追加。
3.30	2019.11.25	1, 7 6 12, 13, 16 33	RX13T のサポートを追加。 2.3 制限事項 制限事項を追加。 RX13T に対応するコードサイズを追加。 「6.1 動作確認環境」：

		プログラム	Rev.3.30 に対応する表を追加。 RX13T のサポートを追加。 API 関数のコメントを Doxygen スタイルに変更。
3.40	2019.12.30	1, 8 13, 15, 16, 18, 19 34 プログラム	RX66N, RX72N のサポートを追加。 RX66N, RX72N に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.40 に対応する表を追加。 RX66N, RX72N のサポートを追加。
3.50	2020.03.31	1,9 14, 18, 22 37 プログラム	RX23E-A のサポートを追加。 RX23E-A に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.50 に対応する表を追加。 RX23E-A のサポートを追加。
3.60	2020.06.30	35 36 プログラム	デモプロジェクトの更新と追加 「5. デモプロジェクト」に RSKRX72M を追加。 「6.1 動作確認環境」： Rev.3.60 に対応する表を追加。 デモプロジェクトの更新と追加
3.70	2021.03.31	1, 8 6 13, 18, 22 37 プログラム	RX671 のサポートを追加。 「1.3.1 IRQ FIT モジュールを C++プロジェクト内で使用する」のセクションを追加。 RX671 に対応するコードサイズを追加。 「6.1 動作確認環境」： Rev.3.70 に対応する表を追加。 RX671 のサポートを追加。
3.80	2021.04.15	1, 8 14, 19, 24 6, 37 40 プログラム	RX140 のサポートを追加。 RX140 に対応するコードサイズを追加。 IR フラグをクリアするための関数 R_IRQ_IRClear()を追加。 「6.1 動作確認環境」： Rev.3.80 に対応する表を追加。 RX140 のサポートを追加。 デモプロジェクトに CS+ のサポートを追加。 IR フラグをクリアするための関数 R_IRQ_IRClear()を追加。
3.90	2021.09.13	39 40 プログラム	「5. デモプロジェクト」に RSKRX671 を追加。 「6.1 動作確認環境」： Rev.3.90 に対応する表を追加。 デモプロジェクトの更新と追加。

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。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

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。