RX ファミリ
IRQ モジュール
Firmware Integration Technology

R01AN1668JJ0200
Rev. 2.00
2016.10.01

要旨

本モジュールでサポートする RX ファミリ MCU には、外部端子をトリガにできる割り込みベクタを備えています。これらの割り込みは IRQ0 から始まり、それぞれ IRQ(n)と称されます。割り込み要求は、端子の状態の High から Low への変更（立ち下がりエッジ）、Low から High への変更（立ち上がりエッジ）、または Low レベル固定をトリガとして生成することができます。デジタルフィルタ機能を割り当てることもできます。これらの選択可能な設定は、各 IRQ 割り込みに個別に割り当てることができます。

本ドキュメントは、Firmware Integration Technology (FIT)を使用した IRQ モジュールについて説明します。ユーザアプリケーションに IRQ モジュールを容易に組み込んでいただけるように、ソフトウェアアーキテクチャ、システムインタフェース、および使用方法について詳しく説明していきます。

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

本モジュールは以下のデバイスで使用できます。

- RX110 グループ
- RX111 グループ
- RX113 グループ
- RX130 グループ
- RX210 グループ
- RX230 グループ
- RX231 グループ
- RX23T グループ
- RX24T グループ
- RX63N グループ
- RX64M グループ
- RX65N グループ
- RX71M 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。

関連ドキュメント

- Firmware Integration Technology ユーザーズマニュアル (R01AN1833)
- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)
- CS+に組み込む方法 Firmware Integration Technology (R01AN1826)

目次

1. 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。

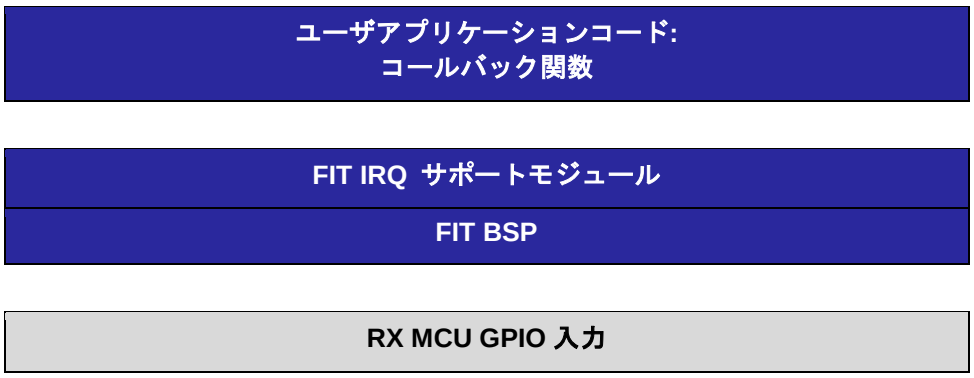

図 1.1　プロジェクト層の例

1.1　IRQ FIT モジュールを使用する

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2.7をご覧ください。

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。

1.2 割り込み

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。

1.3 コールバック関数

コールバック関数の定義は FIT 1.0 の仕様に準じています。

a. コールバック関数では 1 つの引数‘void * pdata’を使用する。

b. コールバック関数を呼び出す前に、関数ポインタが有効であることを確認する。最低でも下記は確認すること。

   i. null でない。

   ii. FIT_NO_FUNC マクロと同等のマクロでない。

1.3.1 コールバック関数のプロトタイプ宣言例

```
void callback(void * pdata)
```

1.3.2 pdata 引数を逆参照する

ISR がコールバック関数を呼び出すと、割り込みをトリガした IRQ 番号を含む値にポインタを渡します。この値のタイプは irq_number_t で、コールバックに渡される引数のタイプ（irq_number_t *）となります。FIT のコールバックは void のポインタを取るため、逆参照ができるようにポインタを型変換する必要があります。

Example

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

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

2. API 情報

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

2.1 ハードウェアの要求

本モジュールを使用するには、ご使用の MCU が以下の周辺機能をサポートしていることが要求されます。

- 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

本モジュールは以下の FIT モジュールに依存します。

- Renesas ボードサポートパッケージ（r_bsp）。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

本モジュールは、外部端子割り込みなど、IRQ タイプの割り込みとのみ使用できます。

2.4 対応ツールチェーン

本モジュールは下記ツールチェーンで動作確認を行っています。

- Renesas RX Toolchain v.2.02.00（RX110、RX111、RX113、RX210、RX231、RX63N、RX64M、RX71M）
- Renesas RX Toolchain v.2.03.00（RX130、RX230、RX23T、RX24T）
- Renesas RX Toolchain v.2.05.00（RX65N）

2.5 ヘッダファイル

すべての API 呼び出しと対応するインタフェース定義は、r_irq_rx_if.h ファイルに記述されています。

r_irq_rx_config.h ファイルで、ビルド時に設定可能なコンフィギュレーションオプションを選択あるいは定義できます。

上記 2 ファイルは、ユーザアプリケーションによってインクルードする必要があります。

2.6 整数型

コードをわかりやすく、また移植が容易に行えるように、本プロジェクトでは ANSI C99 (Exact width integer types (固定幅の整数型)) 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

ビルド時に設定可能なコンフィギュレーションオプションは r_irq_rx_config.h ファイルに含まれます。下表に各設定の概要を示します。

コンフィギュレーションオプション（r_irq_rx_config.h）	
定義	**説明**
IRQ_CFG_USE_IRQn ※デフォルト値は“0”	n が使用される IRQ 番号のとき、このオプションを 1 に設定すると、ビルド時に未使用の IRQ 用のコードを省きます。
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。
	Example:
	`/* 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
IRQ_PORT_BIT_IRQn
	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* (PORTm)`
	（m はポート番号）
	`#define IRQ_PORT_BIT_IRQ* (IRQ_BITn)`
	（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。

表 1：コンフィギュレーション情報

2.8 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「2.7 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 対応ツールチェーン」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

デバイス	分類		使用メモリ		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	
RX110, RX111	ROM	1 IRQ	701 バイト	448 バイト	
		8 IRQs	1107 バイト	854 バイト	
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	40 バイト	32 バイト	
	最大使用スタックサイズ		48 バイト	40 バイト	
RX130, RX230, RX24T	ROM	1 IRQ	701 バイト	450 バイト	
		8 IRQs	1107 バイト	856 バイト	
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	40 バイト	32 バイト	
	最大使用スタックサイズ		48 バイト	40 バイト	
RX113, RX231	ROM	1 IRQ	701 バイト	448 バイト	
		8 IRQs	1107 バイト	854 バイト	
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	40 バイト	32 バイト	
	最大使用スタックサイズ		48 バイト	40 バイト	

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		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	
RX210	ROM	1 IRQ	701 バイト	448 バイト	
		8 IRQs	1107 バイト	854 バイト	
	RAM	1 IRQ	12 バイト	4 バイト	
		8 IRQs	40 バイト	32 バイト	
	最大使用スタックサイズ		48 バイト	40 バイト	
RX23T	ROM	1 IRQ	961 バイト	444 バイト	
		6 IRQs	981 バイト	734 バイト	
	RAM	1 IRQ	10 バイト	4 バイト	
		6 IRQs	30 バイト	24 バイト	
	最大使用スタックサイズ		48 バイト	40 バイト	
RX63N, RX64M, RX65N, RX71M	ROM	1 IRQ	815 バイト	563 バイト	
		16 IRQs	1685 バイト	1433 バイト	
	RAM	1 IRQ	20 バイト	4 バイト	
		16 IRQs	80 バイト	64 バイト	
	最大使用スタックサイズ		48 バイト	40 バイト	

表 2；ROM、RAM およびスタックのコードサイズ

2.9 API データ型

本モジュールの API で使用されるデータ構造体について説明します。

2.9.1 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。

2.10 戻り値

以下に API 関数の戻り値を示します。戻り値の型は r_irq_rx_if.h ファイルで定義されます。

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。

2.11 FIT モジュールの追加方法

本モジュールは、e² studio で、使用するプロジェクトごとに追加する必要があります。

プロジェクトへの追加方法は、FIT プラグインを使用する方法と、手動で追加する方法があります。

FIT プラグインを使用すると、簡単にプロジェクトに FIT モジュールを追加でき、またインクルードファイルパスも自動的に更新できます。このため、プロジェクトへ FIT モジュールを追加する際は、FIT プラグインの使用を推奨します。

FIT プラグインを使用して FIT モジュールを追加する方法は、アプリケーションノート「e² studio に組み込む方法(R01AN1723)」の「3. FIT プラグインを使用して FIT モジュールをプロジェクトに追加する方法」を参照してください。

FIT プラグインを使用せず手動で FIT モジュールを追加する方法は、「4. 手作業で FIT モジュールをプロジェクトに追加する方法」を参照してください。

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。

3. API 関数

3.1 概要

本モジュールには以下の関数が含まれます。

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。

3.2 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

初期化対象の IRQ 番号

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

priority

IRQ の割り込み優先レベルに使用する列挙型

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。

Reentrant

本関数は同一 IRQ に対して再入動作を行いません。別の IRQ に対しては再入可能（リエントラント）です。これは FIT BSP のハードウェアロック関数を使用して、再入動作を制限し、コードの脆弱な部分を保護するためです。

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_t       my_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

3.3 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

handle

IRQ のハンドル

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      /* Control コマンドが認識されません。*/
IRQ_ERR_INVALID_PTR      /* pcmd_data ポインタ、またはハンドルが NULL です。*/
IRQ_ERR_INVALID_ARG      /* pcmd_data 構造体の要素が無効な値を含みます。*/
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。

Reentrant

ロック機能が有効な場合は再入可能（リエントラント）です。同一 IRQ で動作するための再入は、BSP ロック機能によって拒否されます。ロック機能が無効な場合、別の IRQ で再入する場合のみ正常に動作します。ロック機能が有効な場合、デッドロックがかからないように特に注意が必要です。ロック機能を有効にした状態で行われる処理が正常に完了できるように、呼び出し元で戻り値を確認してください。

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

3.4 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

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、または利用不可です。*/
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。

Reentrant

再入可能（リエントラント）。ただし、同一 IRQ 番号に対して再入した場合、IRQ_ERR_NOT_OPENED コードが返されます。

Example

```
/* */
irq_err_t result;

result = R_IRQ_Close(my_handle);
```

Special Notes:

なし

3.5 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

handle

IRQ のハンドル

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、または利用不可です。 */*
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。

Reentrant

再入可能（リエントラント）です。

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

3.6 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

handle

IRQ のハンドル

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、または利用不可です。 */*
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。

Reentrant

再入可能（リエントラント）です。ただし、同一 IRQ 番号に対する不用意な再入は、アプリケーションの動作も予測不能となりますのでご注意ください。

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

3.7 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。

Reentrant

この関数は再入可能（リエントラント）です。

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

4. 端子設定

本モジュールで使用する端子の設定は、R_IRQ_Open() 関数を呼び出した後に行ってください。

5. デモプロジェクト

デモプロジェクトはスタンドアロンプログラムです。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_rskrx113、irq_demo_rskrx231、irq_demo_rskrx64m および irq_demo_rskrx71m は、それぞれ RSKRX113、RSKRX231、RSKRX64M および RSKRX71M 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）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 割り込み（RX231 では IRQ4 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。

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。

- 対応しているテクニカルアップデートはありません。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.70	2015.09.30	—	初版発行
1.80	2015.10.01	— — 7	FIT モジュールが RX130 グループに対応 戻り値の誤記訂正: IRQ_ERR_NOT_OPEN → IRQ_ERR_NOT_OPENED 2.8「コードサイズ」に RX130 グループのコードサイズを追加。
1.90	2015.12.01	— 1, 9 7 19	FIT モジュールが RX230 グループ、RX24T グループに対応 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 Firmware Integration Technology」のドキュメント番号変更 2.8「コードサイズ」に RX230、RX24T グループを追加。 「4. デモプロジェクト」追加
1.91	2016.06.15	19 20	「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」追加
2.00	2016.10.01	— 7, 8 20	FIT モジュールが RX65N グループに対応 2.8「コードサイズ」の表のフォーマットを変更 2.8「コードサイズ」に RX65N グループを追加。 「4. 端子設定」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. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 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に関し、当社は、何らの責任を負うものではありません。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. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、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。お客様が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. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
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。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。
注2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/contact/