RXファミリ
EEPROM アクセス I²C バスインタフェース (RIIC) モジュール Firmware Integration Technology

R01AN1810JJ0141
Rev.1.41
2019.02.01

要旨

本アプリケーションノートでは、Firmware Integration Technology (FIT)を使用した EEPROM アクセス I²C バスインタフェース (RIIC) モジュールについて説明します。本モジュールは I²C バスインタフェース(RIIC)を使用して、EEPROM へ書き込み、読み出し通信を行います。以降、本モジュールを EEPROM アクセス (RIIC) FIT モジュールと称します。

対象デバイス

以下は、この API によってサポートできるデバイスの一覧です。

・RX111 グループ

・RX110 グループ

・RX113 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。

関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。併せて参照してください。

- Firmware Integration Technology ユーザーズマニュアル Rev.1.00 (R01AN1833JU)
- ボードサポートパッケージモジュール Firmware Integration Technology Rev.2.70 (R01AN1685JU)
- e²studio に組み込む方法 Firmware Integration Technology Rev.1.10 (R01AN1723JU)
- Cube Suite+に組み込む方法 Firmware Integration Technology Rev.1.00 (R01AN1826JJ)
- RX ファミリ RIIC モジュール Firmware Integration Technology Rev.1.40 (R01AN1692JJ)

目次

1. 概要

EEPROM アクセス (RIIC) FIT モジュールは、RIIC を使用し、EEPROM への書き込み、および読み出しを行う API 関数を提供します。RIIC は、NXP 社が提唱する I²C バス(Inter-IC-Bus) インタフェース方式に準拠した RIIC FIT モジュールを使用しています。以下に本モジュールがサポートしている機能を列挙します。

- EEPROM へのデータ書き込み、EEPROM からの読み出しに対応しています。
- EEPROM のメモリアアドレスが 1 バイトおよび 2 バイトに対応しています。
- 1 ブロックのサイズが 255 バイトまでの EEPROM に対応しています。
- 複数の EEPROM にアクセス可能です。
- 通信モードは、スタンダードモードとファストモードに対応し、最大転送速度は 400kbps です。

制限事項

本モジュールには、以下の制限事項があります。

- デバイスアドレスが 10 ビットの EEPROM には、対応していません。
- そのほか RIIC モジュールの制限事項を参照してください。

1.1 EEPROM アクセス (RIIC) 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、2.10を参照してください。

1.2　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。また、表 1.2に本モジュールで必要なメモリサイズを示します。

表1.1　API 関数一覧

関数	関数説明
R_EEPROM_RIIC_Open()	RIIC の初期設定を行い、本モジュールが使用できる状態にします。
R_EEPROM_RIIC_Write()	EEPROM への書き込みを開始します。
R_EEPROM_RIIC_Read()	EEPROM からの読み出しを開始します。
R_EEPROM_RIIC_Advance()	EEPROM への書き込み処理、および、EEPROM からの読み出し処理を進めます。
R_EEPROM_RIIC_Close()	RIIC の対象チャネルを解放します。
R_EEPROM_RIIC_GetVersion()	本モジュールのバージョン番号を返します。

表1.2　必要メモリサイズ

使用メモリ	サイズ	備考
ROM	1342 バイト	別途、RIIC FIT モジュール分必要です。 RIIC FIT モジュールの使用メモリサイズは、 RIIC FIT モジュールの アプリケーションノートをご確認ください。
RAM	12 バイト+ 96 バイト×チャネル	
最大使用ユーザスタック	200 バイト	
最大使用割り込みスタック	224 バイト	

※測定時のコンフィギュレーションオプションは、2.6 コンパイル時の設定 に示すデフォルト値です。

※コンパイルオプションがデフォルトの時の値です。
必要メモリサイズは 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。

1.3　EEPROM アクセス (RIIC) FIT モジュールの概要

1.3.1　EEPROM アクセス (RIIC) FIT モジュールの仕様

・本モジュールは、EEPROM への書き込み、読み出しをサポートします。

－ EEPROM への書き込みの処理例は、1.4.1 に示します。
－ EEPROM からの読み出しの処理例は、1.4.2 に示します。

・ メモリアドレスは、1～2 バイトのデータ (00 ~ FFh または 0000 ~ FFFFh) で指定可能です。
デバイスアドレスにメモリアドレスを含む EEPROM をご使用の場合は、1.3.2 の注意事項を参照してください。

・ 書き込みおよび読み出しは、書き込み、読み出し開始関数を呼び出す際に総バイト数および EEPROM の 1 ブロックサイズを指定する必要があります。設定に応じて 1 ブロックごとにスタートコンディション、ストップコンディションを発生させます。
1 ブロック単位のデータサイズ (ページサイズ) は、最大 255 バイトまで指定可能です。

・ NACK を検出した際、リトライを行います。下記の設定の詳細は、2.6 を参照してください。
－リトライ実施前に待つ時間が設定可能です。
－リトライ実施回数が設定可能です。

・ 通信速度や通信に使用する端子などは、RIIC FIT モジュールのコンフィグレーションファイル (r_riic_rx_config.h) で設定してください。

・ 1 つのチャネル・バス上の複数の EEPROM を制御できます。ただし、通信中（スタートコンディションから、ストップコンディション生成完了までの期間）は、そのデバイス以外の通信はできません。図 1.1 に複数の EEPROM にアクセスする場合の制御例を示します。

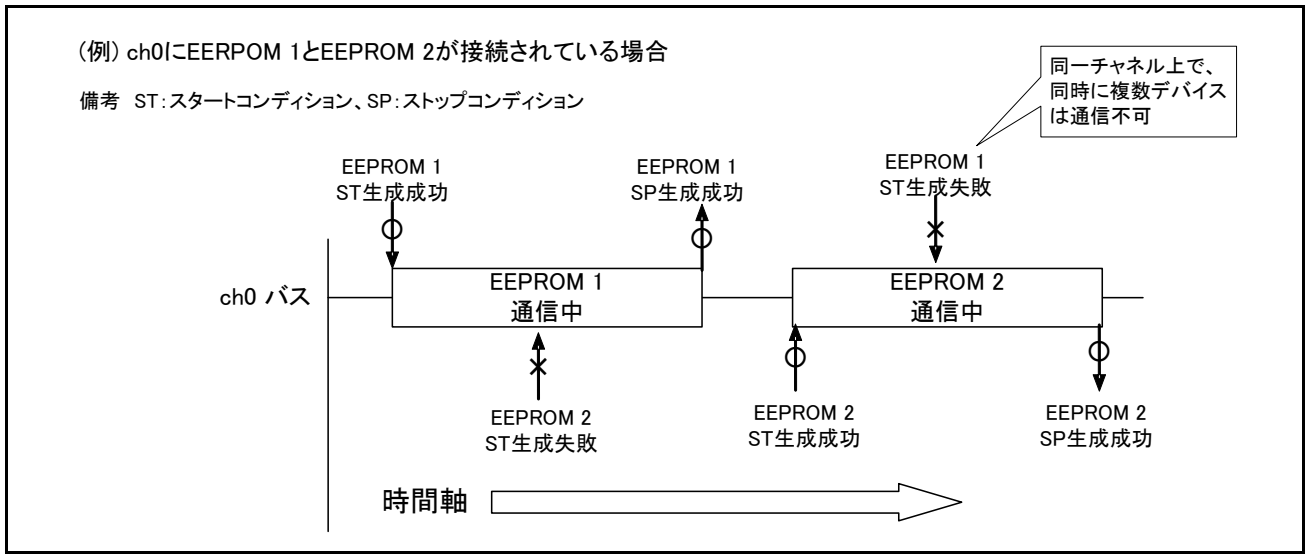

図 1.1　複数の EEPROM の制御例

1.3.2　デバイスアドレスにメモリアドレスを含む EEPROM を使用する場合の注意事項

本モジュールでは、デバイスアドレス内のメモリアドレスを自動更新することができません。
そのため、図 1.2の例のように、メモリアドレスが 11 ビットでそのうちの 3 ビットをデバイスアドレスに含む場合（図 1.2は a8 ~ a10）は、メモリアドレスのサイズを 1 バイトに設定してください。また、1 バイトのメモリアドレスがオーバーフローしないように下記を設定してください。

- メモリアドレス
- 書き込み/読み出しブロックサイズ
- 総書き込み/読み出しデータサイズ

その後、全データの書き込み/読み出し完了後にコールバック関数などでデバイスアドレス内のメモリアドレスを更新し、書き込み/読み出しを繰り返し行ってください。

図 1.2にデバイスアドレスにメモリアドレスを含む一例を示します。

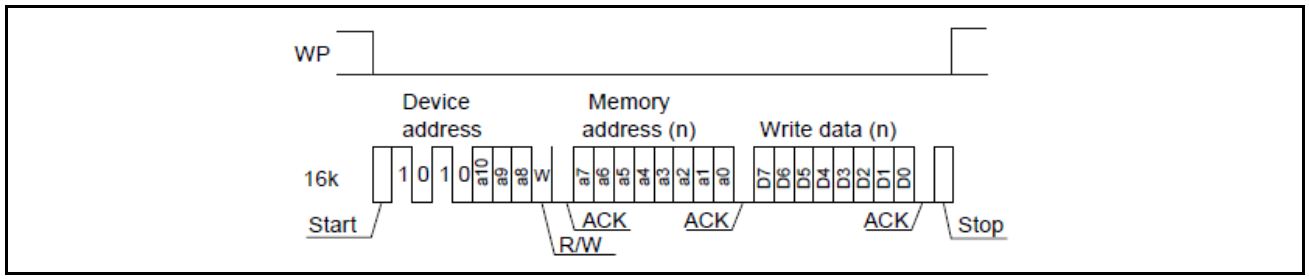

図 1.2　デバイスアドレスにメモリアドレスを含む一例

1.4　処理例

1.4.1　EEPROM への書き込み処理例

図 1.3に EEPROM への書き込む際の手順を示します。コールバック関数は、全データ書き込みが完了した、もしくはタイムアウトかアービトレーションが発生したタイミングで呼ばれます。EEPROM 情報構造体メンバ の callbackfunc に関数名を指定してください。

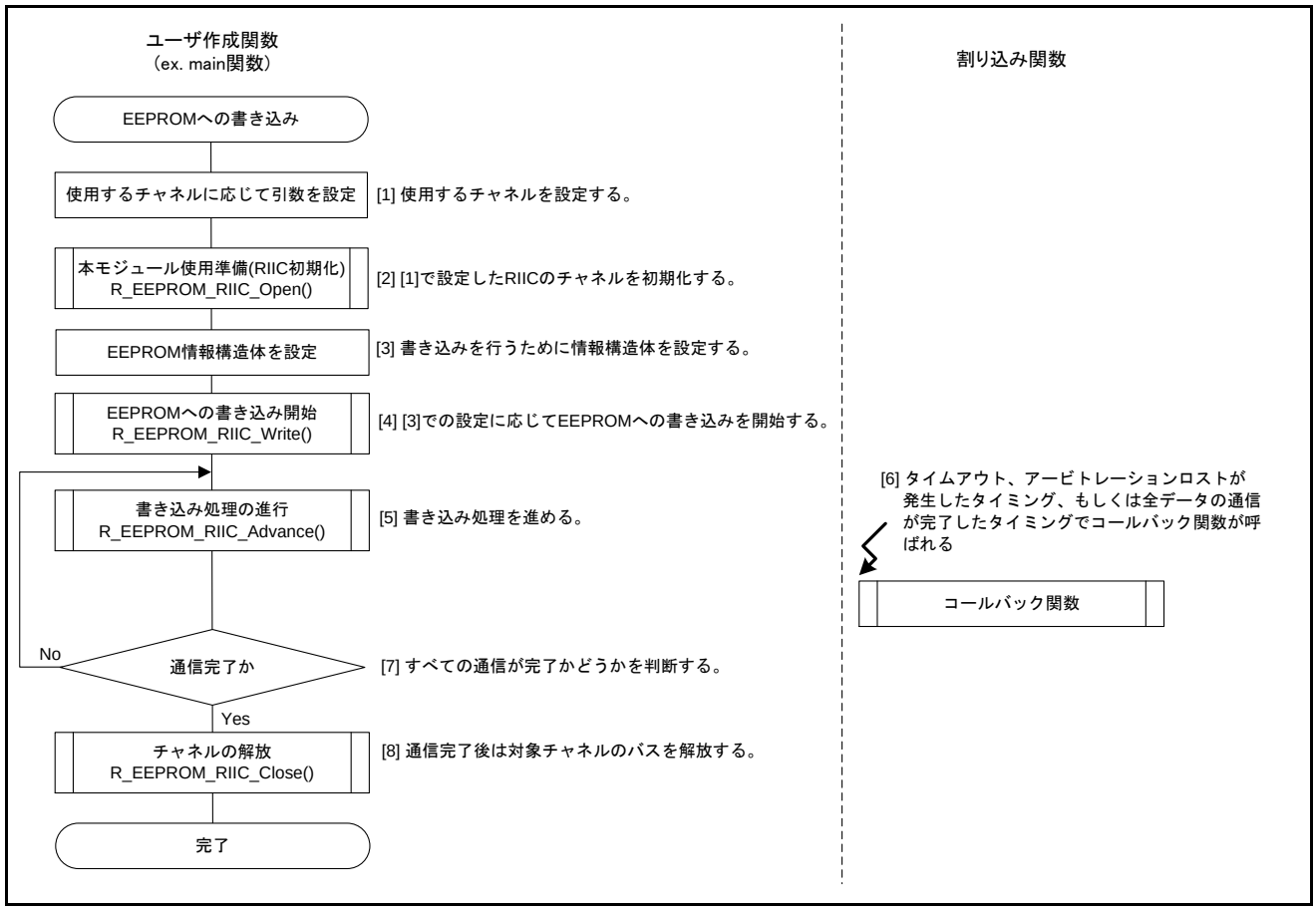

図 1.3　EEPROM への書き込み手順例

1.4.2　EEPROM からの読み出し処理例

図 1.4に EEPROM からの読み出す際の手順を示します。コールバック関数は、全データ読み出しが完了した、もしくはタイムアウトかアービトレーションロストが発生したタイミングで呼ばれます。EEPROM 情報構造体メンバの callbackfunc に関数名を指定してください。

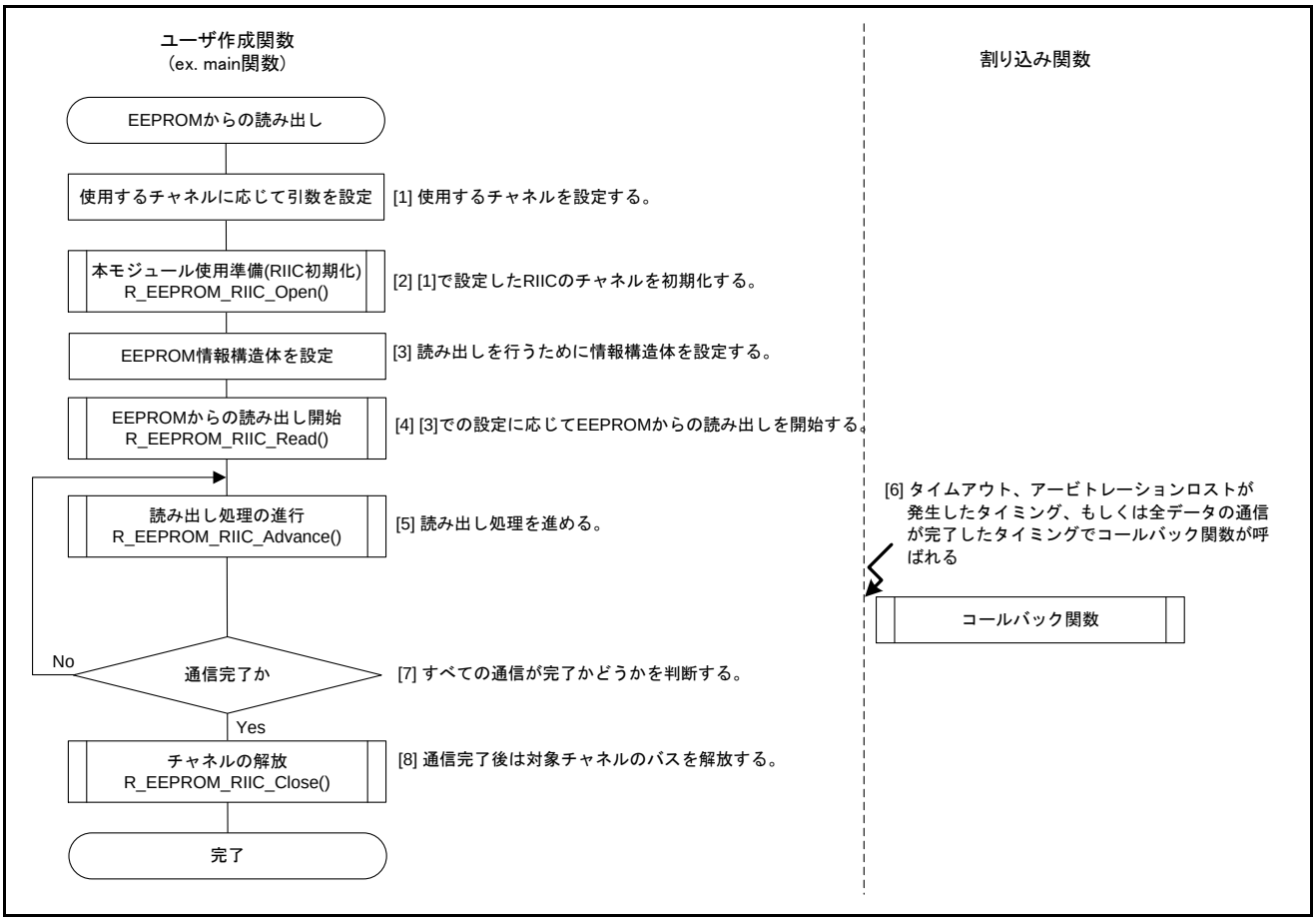

図 1.4 EEPROM からの読み出し手順例

1.5　状態遷移図

本モジュールの状態遷移図を図 1.5に示します。

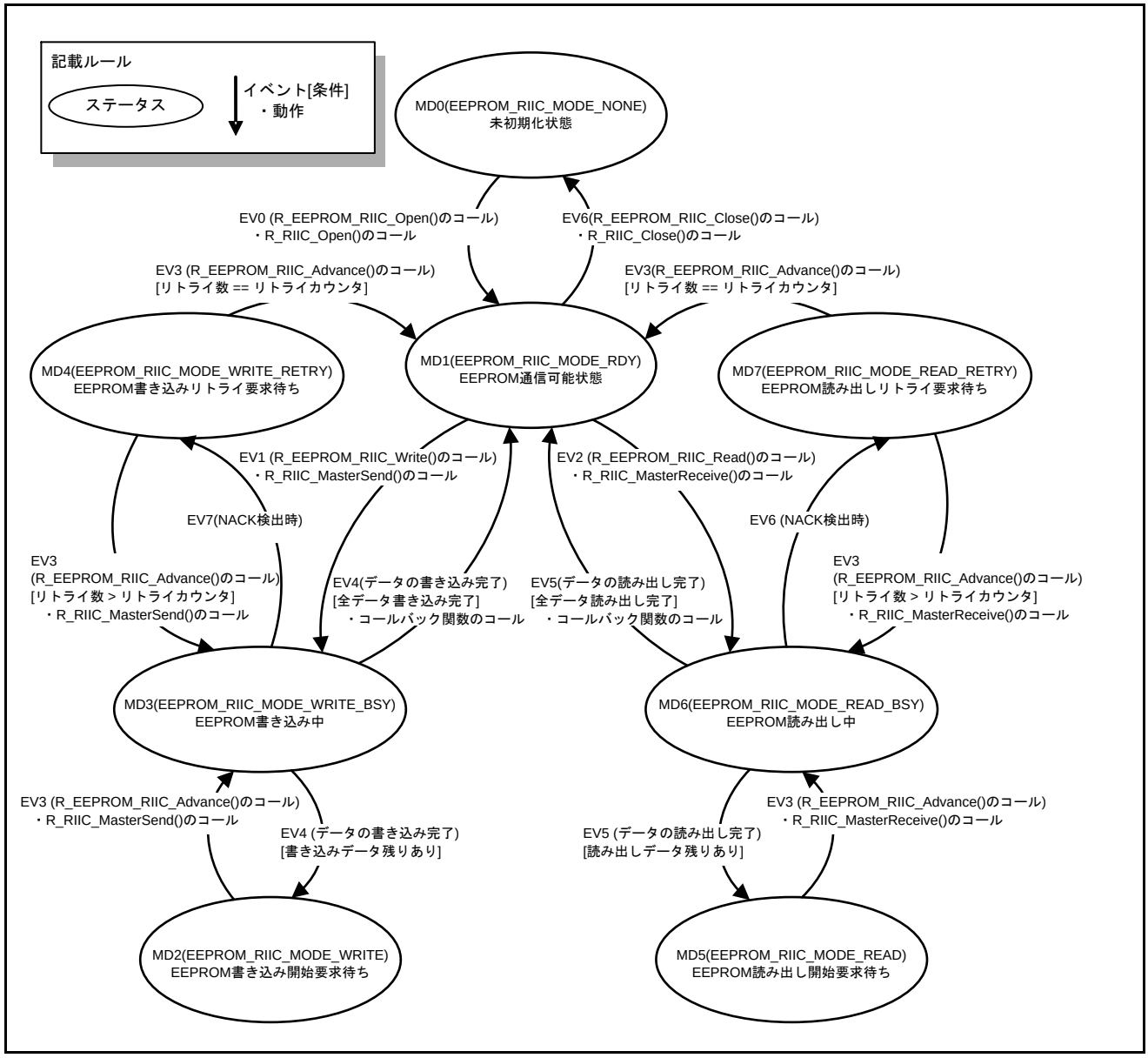

図 1.5　EEPROM アクセス (RIIC) FIT モジュールの状態遷移図

1.6　タイムアウト検出

タイムアウトを検出した場合、R_EEPROM_RIIC_Close 関数をコールした後、R_EEPROM_RIIC_Open 関数をコールし、通信を再開してください。

また、再開後も SDA ラインが Low 固定される場合があります。この時は、SCL クロック追加出力機能を使用して、バスを解放してください。詳細は、RIIC モジュール FIT (R01AN1692JJ)の「タイムアウト検出機能によるタイムアウト検出方法と対処方法」を参照ください。

2. API 情報

本ドライバの API はルネサスの API の命名基準に従っています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- RIIC

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。

- r_bsp
- r_riic_rx

2.3 サポートされているツールチェイン

このドライバは下記ツールチェインで動作確認を行っています。

- Renesas RX Toolchain v.2.02

2.4 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_eeprom_riic_rx_if.h に記載しています。

2.5 整数型

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

2.6 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_eeprom_riic_rx_config.h で行います。

オプション名および設定値に関する説明を、下表に示します。

Configuration options in *r_eeprom_riic_rx_config.h*	
EEPROM_RIIC_CFG_PARAM_CHECKING ※デフォルト値は“1”	パラメータチェック処理をコードに含めるか選択できます。 “0”の場合、パラメータチェック処理をコードから省略します。 “1”の場合、パラメータチェック処理をコードに含めます。
EEPROM_RIIC_CFG_COUNT_RETRY ※デフォルト値は“100”	NACK 検出時に行うリトライ (再書き込み、再読み出し) を行う回数を設定できます。 “0xFFFFFFFF”以下の値を設定してください。
EEPROM_ RIIC_CFG_WAIT_CNT ※デフォルト値は“1000”	リトライする前に一定の待ち時間 (for 文によるループ処理) が挿入されます。その一定の待ち時間 (for 文のループ回数) を設定できます。 “0xFFFFFFFF”以下の値を設定してください。
EEPROM_RIIC_CFG_CHi_TABLE_NO i=0～3 ※i = 0 のデフォルト値は“0” ※i = 1 のデフォルト値は“DUMMY” ※i = 2 のデフォルト値は“DUMMY” ※i = 3 のデフォルト値は“DUMMY”	RIICi を使用しないか、使用するかを設定します。 使用しない場合は、“DUMMY”を設定します。 使用する場合は、“0”から MAX_EEPROM_RIIC_CH_NUM より小さい値の範囲で設定します。同じ値を複数に設定しないでください。
EEPROM_RIIC_CH_MAXUSE_NUM ※デフォルト値は、“1”	使用する RIIC のチャネル数を設定します。 “1”以上、かつ MAX_EEPROM_RIIC_CH_NUM より小さい値を設定してください。

2.7 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_eeprom_riic_rx_if.h に記載されています。

```
typedef volatile struct
{
    uint8_t             rsv1;                   /* 予約領域                                */
    uint8_t             ch_no;                  /* RIIC のチャネル番号                     */
    uint8_t             size_wr_blk_byte;       /* 書き込みブロックサイズ                  */
    uint8_t             size_mem_adr_byte;      /* メモリアドレスのサイズ                  */
    eeprom_riic_callback    callbackfunc;       /* コールバック関数                        */
    uint32_t            cnt_all_wr_data;        /* 総書き込みデータ数                      */
    uint8_t *           p_wr_data;              /* 書き込みデータのポインタ                */
    uint16_t            mem_adr;                /* メモリアドレス                          */
    uint16_t            dev_adr;                /* デバイスアドレス                        */
} eeprom_riic_wr_info_t;        //EEPROM への書き込み時に使用する構造体

typedef volatile struct
{
    uint8_t             rsv1;                   /* 予約領域                                */
    uint8_t             ch_no;                  /* RIIC のチャネル番号                     */
    uint8_t             size_rd_blk_byte;       /* 読み出しブロックサイズ                  */
    uint8_t             size_mem_adr_byte;      /* メモリアドレスのサイズ                  */
    eeprom_riic_callback    callbackfunc;       /* コールバック関数                        */
    uint32_t            cnt_all_rd_data;        /* 総読み出しデータ数                      */
    uint8_t *           p_rd_data;              /* 読み出しデータのポインタ                */
    uint16_t            mem_adr;                /* メモリアドレス                          */
    uint16_t            dev_adr;                /* デバイスアドレス                        */
} eeprom_riic_rd_info_t;        //EEPROM からの読み出し時に使用する構造体
```

2.8 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_eeprom_riic_rx_if.h で記載されています。

```
typedef enum
{
    EEPROM_RIIC_SUCCESS = 0U,           /* 関数コールが正常にできた場合                */
    EEPROM_RIIC_ERR_LOCK_FUNC,          /* 他のモジュールで RIIC が使用されている場合  */
    EEPROM_RIIC_ERR_INVALID_CHAN,       /* 存在しないチャネルの場合                    */
    EEPROM_RIIC_ERR_INVALID_ARG,        /* 不正な引数の場合                            */
    EEPROM_RIIC_ERR_NO_INIT,            /* 初期設定ができていない場合 (未初期化状態)   */
    EEPROM_RIIC_ERR_BUS_BUSY,           /* バスビジーの場合                            */
    EEPROM_RIIC_ERR_AL,                 /* アービトレーションロストの場合              */
    EEPROM_RIIC_ERR_TMO,                /* バスタイムアウトの場合                      */
    EEPROM_RIIC_ERR_OTHER               /* その他のエラー                              */
} eeprom_riic_return_t;
```

2.9 コールバック関数

本ジュールでは、EEPROM へ全データ(※1)の書き込み、または、全データ(※2)の読み出しが完了したタイミングで、コールバック関数を呼び出します。

コールバック関数の設定は、「2.7 引数」に記載の構造体メンバ“callbackfunc” に、コールバック関数として登録したい関数のアドレスを格納してください。

コールバック関数が呼び出されるとき、表 2.1に示す定数が格納された変数が、引数として渡されます。コールバック関数実行後、変数の値が“EEPROM_RIIC_NONE”に初期化されます。引数の値をコールバック関数外で使用する場合は、グローバル変数などにコピーしてください。

使用例は、「3.2 R_EEPROM_RIIC_Write()」、「3.3 R_EEPROM_RIIC_Read()」の節の Example をご参照ください。

※1. 「2.7 引数」に記載の構造体メンバ“cnt_all_wr_data” に設定したデータ数

※2. 「2.7 引数」に記載の構造体メンバ“cnt_all_rd_data” に設定したデータ数

表2.1　コールバック関数の引数一覧(enum eeprom_riic_status_t)

定数定義	変化するタイミング
EEPROM_RIIC_NONE	初期値、コールバック関数実行後
EEPROM_RIIC_FINISH	全データの書き込み、または、全データの読み出しが完了したとき
EEPROM_RIIC_TIMEOUT	NACK 検出時のリトライ処理回数が、コンフィギュレーションオプションの“EEPROM_RIIC_CFG_COUNT_RETRY”の設定値を超えたとき
EEPROM_RIIC_AL	アービトレーションロスト検出により、通信中断されたとき
EEPROM_RIIC_TMO	バスタイムアウト検出により、通信中断されたとき

2.10 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、自動的にインクルードファイルパスが更新されます。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「e2studio に組み込む方法(R01AN1723JU)」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、BSP FIT 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 FIT モジュールの詳細については、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(R01AN1685JU)」を参照してください。

3. API 関数

3.1 R_EEPROM_RIIC_Open()

RIIC の初期設定を行い、本モジュールが使用できる状態にする関数です。

Format

```
eeprom_riic_return_t R_EEPROM_RIIC_Open (
        uint8_t             ch        /* チャネル番号 */
)
```

Parameters

uint8_t　ch

チャネル番号を設定してください。

Return Values

```
EEPROM_RIIC_SUCCESS ,          /* 関数コールが正常にできた場合                */
EEPROM_RIIC_ERR_LOCK_FUNC,     /* 他のモジュールでRIIC が使用されている場合   */
EEPROM_RIIC_ERR_INVALID_CHAN,  /* 存在しないチャネルの場合                    */
EEPROM_RIIC_ERR_INVALID_ARG,   /* 不正な引数の場合                            */
EEPROM_RIIC_ERR_OTHER          /* その他のエラー                              */
```

Properties

r_eeprom_riic_rx_if.h にプロトタイプ宣言されています。

Description

EEPROM との通信を開始するための設定を行います。引数で指定した RIIC のチャネルの初期設定を行います。本関数では次の処理を行います。

- 設定されたチャネル番号を RIIC モジュールの I²C 情報構造体に引渡し
- 動作モードを“通信可能状態”(EEPROM_RIIC_MODE_RDY)に更新
- RIIC モジュールの R_RIIC_Open 関数(初期化関数)の呼び出し

Reentrant

- なし

Example

```
volatile eeprom_riic_return_t   ret;
eeprom_riic_wr_info_t     eep_w;

eep_w.ch_no = 0;


/* Open channel for writing EEPROM */
ret = R_EEPROM_RIIC_Open(eep_w.ch_no);
```

Special Notes:

なし

3.2 R_EEPROM_RIIC_Write()

EEPROM への書き込みを開始する関数です。

Format

```
eeprom_riic_return_t R_EEPROM_RIIC_Write(
        eeprom_riic_wr_info_t *                 p_eeprom_riic_info      /* 構造体データ */
)
```

Parameters

*eeprom_riic_wr_info_t *　　p_eeprom_riic_info*

EEPROM 情報構造体のポインタ。この構造体の詳細については 2.7 を参照してください。
EEPROM デバイスのアドレスを設定する際は、1 ビット左シフトせずに格納してください。

```
uint8_t             rsv1;               /* 予約領域                                */
uint8_t             ch_no;              /* チャネル番号                            */
uint8_t             size_wr_blk_byte;   /* 書き込みブロックサイズ(単位：バイト)    */
uint8_t             size_mem_adr_byte;  /* メモリアドレスサイズ(単位：バイト)      */
eeprom_riic_callback    callbackfunc;   /* コールバック関数                        */
uint32_t            cnt_all_wr_data;    /* 書き込みデータの全データ数              */
uint8_t *           p_wr_data;          /* 書き込みデータのアドレス                */
uint16_t            mem_adr;            /* メモリアドレス                          */
uint16_t            dev_adr;            /* EEPROM デバイスのアドレス               */
```

Return Values

```
EEPROM_RIIC_SUCCESS ,           /* 関数コールが正常にできた場合                    */
EEPROM_RIIC_ERR_INVALID_CHAN,   /* 存在しないチャネルの場合                        */
EEPROM_RIIC_ERR_INVALID_ARG,    /* 不正な引数の場合                                */
EEPROM_RIIC_ERR_NO_INIT,        /* 初期設定ができていない場合 (未初期化状態)       */
EEPROM_RIIC_ERR_BUS_BUSY,       /* バスビジーの場合                                */
EEPROM_RIIC_ERR_AL,             /* アービトレーションロストの場合                  */
 EEPROM_RIIC_ERR_TMO,           /* バスタイムアウトの場合                          */
EEPROM_RIIC_ERR_OTHER           /* その他のエラー                                  */
```

Properties

r_eeprom_riic_rx_if.h にプロトタイプ宣言されています。

Description

引数で指定した RIIC のチャネルを使用して、EEPROM への書き込みを開始します。チャネルの状態が “通信可能状態” (EEPROM_RIIC_MODE_RDY)の場合、次の処理を行います。

- API で使用するグローバル変数の初期化
- 動作モードを更新
- RIIC モジュールの I²C 情報構造体に EEPROM 情報構造体を引渡し
- RIIC モジュールの R_RIIC_MasterSend 関数(マスタ送信関数)の呼び出し
  (この API にてマスタ送信を開始します。)

Reentrant

- なし

Example

```
void CallbackEEPROM(void *);
eeprom_riic_status_t g_event; //source to terminate the EEPROM communication

void main (void)
{
    volatile eeprom_riic_return_t   ret;
    eeprom_riic_wr_info_t    eep_w;
    uint16_t       addr_eeprom = 0x0050;
    uint16_t       access_addr = 0x0000;
    uint8_t        wr_data[9]={0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89};

    /* Set Informations */
    eep_w.ch_no = 0;
    eep_w.callbackfunc = &CallbackEEPROM;
    eep_w.size_wr_blk_byte = 2;
    eep_w.size_mem_adr_byte = 1;
    eep_w.cnt_all_wr_data = 8;
    eep_w.p_wr_data = wr_data;
    eep_w.mem_adr = access_addr;
    eep_w.dev_adr = addr_eeprom;

    /* Open channel for writing EEPROM (refer to section 3.1) */
    ret = R_EEPROM_RIIC_Open(eep_w.ch_no);

    /* Start writing to EEPROM */
    ret = R_EEPROM_RIIC_Write(&eep_w);

    while(EEPROM_RIIC_FINISH != g_event)
    {
        /* Proceeds with programming an EEPROM. (refer to section 3.4) */
        ret = R_EEPROM_RIIC_Advance(eep_w.ch_no);
    }

    /* Close Channel (refer to section 3.5) */
    ret = R_EEPROM_RIIC_Close(eep_w.ch_no);

    while(1)
    {
       /* Do Nothing */
    }
}


/* Callback function */
void CallbackEEPROM(void * p_eeprom_status)
{
      g_event = *(eeprom_riic_status_t *)p_eeprom_status;
}
```

Special Notes:

なし

3.3 R_EEPROM_RIIC_Read()

EEPROM からの読み出しを開始する関数です。

Format

```
eeprom_riic_return_t R_EEPROM_RIIC_Read(
        eeprom_riic_rd_info_t *                 p_eeprom_riic_info      /* 構造体データ */
)
```

Parameters

*eeprom_riic_rd_info_t *　　p_eeprom_riic_info*

EEPROM 情報構造体のポインタ。この構造体の詳細については 2.7 を参照してください。
EEPROM デバイスアドレスを設定する際、1 ビット左シフトせずに格納してください。

```
uint8_t             rsv1;               /* 予約領域                                 */
uint8_t             ch_no;              /* チャネル番号                             */
uint8_t             size_rd_blk_byte;   /* 読み出しブロックサイズ(単位：バイト)     */
uint8_t             size_mem_adr_byte;  /* メモリアドレスサイズ(単位：バイト)       */
eeprom_riic_callback    callbackfunc;   /* コールバック関数                         */
uint32_t            cnt_all_rd_data;    /* 読み出しデータの全データ数               */
uint8_t *           p_rd_data;          /* 読み出しデータのアドレス                 */
uint16_t            mem_adr;            /* メモリアドレス                           */
uint16_t            dev_adr;            /* EEPROM デバイスのアドレス                */
```

Return Values

```
EEPROM_RIIC_SUCCESS ,           /* 関数コールが正常にできた場合                     */
EEPROM_RIIC_ERR_INVALID_CHAN,   /* 存在しないチャネルの場合                         */
EEPROM_RIIC_ERR_INVALID_ARG,    /* 不正な引数の場合                                 */
EEPROM_RIIC_ERR_NO_INIT,        /* 初期設定ができていない場合 (未初期化状態)       */
EEPROM_RIIC_ERR_BUS_BUSY,       /* バスビジーの場合                                 */
EEPROM_RIIC_ERR_AL,             /* アービトレーションロストの場合                   */
EEPROM_RIIC_ERR_TMO,            /* バスタイムアウトの場合                           */
EEPROM_RIIC_ERR_OTHER           /* その他のエラー                                   */
```

Properties

r_eeprom_riic_rx_if.h にプロトタイプ宣言されています。

Description

引数で指定した RIIC のチャネルを使用して、EEPROM からの読み出しを開始します。チャネルの状態が“通信可能状態”(EEPROM_RIIC_MODE_RDY)の場合、次の処理を行います。

- API で使用するグローバル変数の初期化
- 動作モードの更新
- EEPROM 情報構造体を RIIC モジュールの I²C 情報構造体に引渡し
- RIIC モジュールの R_RIIC_MasterReceive 関数(マスタ受信関数)の呼び出し
  (この API にて RIIC 割り込みの許可、スタートコンディションの生成までを行います)

Reentrant

- なし

Example

```
void CallbackEEPROM(void *);
eeprom_riic_status_t g_event; //source to terminate the EEPROM communication

void main(void)
{
    volatile eeprom_riic_return_t   ret;
    eeprom_riic_rd_info_t    eep_r;
    uint16_t       addr_eeprom = 0x0050;
    uint16_t       access_addr = 0x0000;
    uint8_t        store_area[9]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};

    /* Set Informations */
    eep_r.ch_no = 0;
    eep_r.callbackfunc = &CallbackEEPROM;
    eep_r.size_rd_blk_byte = 2;
    eep_r.size_mem_adr_byte = 1;
    eep_r.cnt_all_rd_data = 8;
    eep_r.p_rd_data = store_area;
    eep_r.mem_adr = access_addr;
    eep_r.dev_adr = addr_eeprom;

    /* Open channel for reading EEPROM (refer to section 3.1)*/
    ret = R_EEPROM_RIIC_Open(eep_r.ch_no);

    /* Start reading from EEPROM */
    ret = R_EEPROM_RIIC_Read(&eep_r);

    while(EEPROM_RIIC_FINISH != g_event)
    {
        /* Proceeds with reading an EEPROM (refer to section 3.4) */
        ret = R_EEPROM_RIIC_Advance(eep_r.ch_no);
    }

    /* Close Channel (refer to section 3.5) */
    ret = R_EEPROM_RIIC_Close(eep_r.ch_no);

    while(1)
    {
       /* Do Nothing */
    }

}

/* Callback function */
void CallbackEEPROM(void * p_eeprom_status)
{
      g_event = *(eeprom_riic_status_t *)p_eeprom_status;
}
```

Special Notes:

なし

3.4 R_EEPROM_RIIC_Advance()

EEPROM への書き込み処理、および EEPROM からの読み出し処理を進める関数です。

Format

```
eeprom_riic_return_t R_EEPROM_RIIC_Advance(
        uint8_t             ch        /* チャネル番号 */
)
```

Parameters

uint8_t　　ch

チャネル番号を設定してください。

Return Values

```
EEPROM_RIIC_SUCCESS ,          /* 関数コールが正常にできた場合                    */
EEPROM_RIIC_ERR_INVALID_CHAN,  /* 存在しないチャネルの場合                        */
EEPROM_RIIC_ERR_INVALID_ARG,   /* 不正な引数の場合                                */
EEPROM_RIIC_ERR_NO_INIT,       /* 初期設定ができていない場合 (未初期化状態) */
EEPROM_RIIC_ERR_BUS_BUSY,      /* バスビジーの場合                                */
EEPROM_RIIC_ERR_AL,            /* アービトレーションロストの場合                  */
EEPROM_RIIC_ERR_TMO,           /* バスタイムアウトの場合                          */
EEPROM_RIIC_ERR_OTHER          /* その他のエラー                                  */
```

Properties

r_eeprom_riic_rx_if.h にプロトタイプ宣言されています。

Description

R_EEPROM_RIIC_Write 関数 (EEPROM への書き込み開始関数)もしくは R_EEPROM_RIIC_Read 関数 (EEPROM からの読み出し開始関数)を実行した後、書き込み処理、読み出し処理を進めていくための関数です。動作モードごとに以下の処理を行います。

・動作モードが“EEPROM_RIIC_MODE_WRITE”(EEPROM 書き込み開始要求待ち)
　－ リトライカウンタのクリア
　－ 動作モードを“EEPROM_RIIC_MODE_WRITE_BSY” (EEPROM 書き込み中)に更新
　－ RIIC モジュールの R_RIIC_MasterSend 関数(マスタ送信関数) の呼び出し
　　(EEPROM への書き込み処理を進めます。)

・動作モードが“EEPROM_RIIC_MODE_WRITE_RETRY”(EEPROM 書き込みリトライ要求待ち)
　－ リトライカウンタをインクリメント
　－ リトライカウンタ値が設定している最大数を超えた場合、動作モードを
　　“EEPROM_RIIC_MODE_RDY” (EEPROM 通信可能状態)に更新
　－ リトライカウンタ値が最大数以下の場合、以下の処理を実行
　　－ 動作モードを“EEPROM_RIIC_MODE_WRITE_BSY” (EEPROM 書き込み中)に更新
　　－ RIIC モジュールの R_RIIC_MasterSend 関数(マスタ送信関数) の呼び出し

・動作モードが“EEPROM_RIIC_MODE_READ”(EEPROM 読み出し開始要求待ち)
　－ リトライカウンタのクリア
　－ 動作モードを“EEPROM_RIIC_MODE_READ_BSY” (EEPROM 読み出し中)に更新
　－ RIIC モジュールの R_RIIC_MasterReceive 関数(マスタ受信関数) の呼び出し
　　(EEPROM からの読み出し処理を進めます。)

・動作モードが“EEPROM_RIIC_MODE_READ_RETRY”(EEPROM 読み出しリトライ要求待ち)
　－ リトライカウンタのインクリメント
　－ リトライカウンタ値が設定している最大数を超えた場合、動作モードを
　　“EEPROM_RIIC_MODE_RDY” (EEPROM 通信可能状態)に更新
　－ リトライカウンタ値が最大数以下の場合、以下の処理を実行
　　－ 動作モードを“EEPROM_RIIC_MODE_READ_BSY” (EEPROM 読み出し中)に更新
　　－ RIIC モジュールの R_RIIC_MasterReceive 関数(マスタ受信関数) の呼び出し

Reentrant

- なし

Example

使用方法は、R_EEPROM_RIIC_Write 関数および R_EEPROM_RIIC_Read 関数の“Example”を参照してください。

Special Notes:

なし

3.5 R_EEPROM_RIIC_Close()

EEPROM デバイスとの通信を終了し、使用していた RIIC のチャネルを開放する関数です。

Format

```
eeprom_riic_return_t R_EEPROM_RIIC_Close(
        uint8_t             ch        /* チャネル番号 */
)
```

Parameters

uint8_t　　ch

チャネル番号を設定してください。

Return Values

```
EEPROM_RIIC_SUCCESS ,          /* 関数コールが正常にできた場合      */
EEPROM_RIIC_ERR_INVALID_CHAN,  /* 存在しないチャネルの場合          */
EEPROM_RIIC_ERR_INVALID_ARG,   /* 不正な引数の場合                  */
EEPROM_RIIC_ERR_OTHER          /* その他のエラー                    */
```

Properties

r_eeprom_riic_rx_if.h にプロトタイプ宣言されています。

Description

EEPROM デバイスとの通信を終了するための設定を行います。引数で指定した RIIC のチャネルを開放します。本関数では次の処理を行います。

－ 設定されたチャネル番号を RIIC モジュールの I²C 情報構造体に引渡し
－ 動作モードを“未初期化状態”(EEPROM_RIIC_MODE_NONE)に更新
－ RIIC モジュールの R_RIIC_Close 関数の呼び出し
(この API にて I²C 出力ポートの開放、RIIC 割り込みの禁止を実施します。)

また、再度 EEPROM と通信を開始するには、R_EEPROM_RIIC_Open(初期化関数)をコールする必要があります。通信中に強制的に停止した場合、その通信は保証しません。

Reentrant

- なし

Example

```
eeprom_riic_rd_info_t    eep_r;
eep_r.ch_no = 0;

/* Close Channel */
ret = R_EEPROM_RIIC_Close(eep_r.ch_no);
```

Special Notes:

なし

3.6 R_EEPROM_RIIC_GetVersion()

API のバージョンを返す関数です。

Format
uint32_t　R_EEPROM_RIIC_GetVersion(void)

Parameters
なし

Return Values
バージョン番号

Properties
r_eeprom_riic_rx_if.h にプロトタイプ宣言されています。

Description
本 API のバージョン番号を返します。

Reentrant
- なし

Example
```
uint32_t   version;

version = R_EEPROM_RIIC_GetVersion();
```

Special Notes:
この関数は“#pragma inline”を使用してインライン化されています。

4. 提供するモジュール

提供するモジュールは、ルネサス エレクトロニクスホームページから入手してください。

5. 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 V2.01.00 ユーザーズマニュアル RX コーディング編（R20UT2748JJ）
(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。)

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	RX ファミリ EEPROM アクセス I²C バスインタフェース (RIIC) モジュール Firmware Integration Technology

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2013.09.30	—	初版発行
1.10	2013.11.15	—	タイトルの変更 「EEPROM RIIC FIT モジュール」 →「EEPROM アクセス I²C バスインタフェース (RIIC) モジュール Firmware Integration Technology」
		4	「表 1.2　必要メモリサイズ」 EEPROM アクセス (RIIC) FIT モジュールのみの ROM サイズ、RAM サイズ表記に変更
		12	「2.9 コールバック関数」節を追加
1.20	2014.04.01	—	FIT モジュールの RX100 シリーズ対応
1.30	2014.10.01	1	対象デバイスの追加 　RX110 グループ 「関連ドキュメント」の項目を追加
		4	必要メモリサイズの変更 　必要メモリサイズの情報を変更
		9	タイムアウト検出の記載を追加
		10	記の訂正 　r_cgc_rx に依存していないため、「2.2 ソフトウェアの要求」から r_cgc_rx を削除 ツールチェインの変更 　コンパイラのバージョンアップ
		11	コンパイル時の設定の追加 　パラメータチェック処理用の定義を追加 　使用するチャネルの定義を追加
		12,13	タイムアウト検出エラーの定数を追加
		13	FIT モジュールの追加方法 　追加方法の記載を見直し、変更
		14	Return Values の記載を見直し Example のコード記載を追加(変数 ret の定義を追加)
		15,17,19	Return Values にタイムアウト検出エラーの定数を追加
		21	Return Values の記載を見直し
		23	参照ドキュメントのバージョンアップ 　「ユーザーズマニュアル：開発環境」の変更
1.40	2014.12.01	—	FIT モジュールの RX113 グループ対応
1.41	2019.02.01	—	機能関連 Smart Configurator での GUI によるコンフィグオプション設定機能に対応 ■内容 GUI によるコンフィグオプション設定機能に対応するため、設定ファイルを追加。

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)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

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

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

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

Colophon 6.0