RZ/T1グループ

RIICサンプルプログラム

R01AN2596JJ0140
Rev.1.40
2018.06.07

要旨

本アプリケーションノートでは、RZ/T1 の I²C バスインタフェース機能（RIIC）を使用し評価ボード上に実装されている EEPROM（R1EX24016ASAS0A）の Read/Write を実行するサンプルプログラムについて説明します。

RIIC サンプルプログラムの特長を以下に示します。

- マスタ送信、マスタ受信対応
- 通信モードはファストモードに対応（最大転送速度は 400kbps）

制限事項
本サンプルプログラムには以下の制限事項があります。
(1) DMA と組み合わせて使用することはできません。
(2) RIIC のタイムアウト機能には対応していません。
(3) RIIC の NACK アービトレーションロスト機能に対応していません。
(4) 10 ビットアドレスの送信に対応していません。
(5) スレーブデバイス時、リスタートコンディションの受付に対応していません。
リスタートコンディション直後のアドレスで本モジュールのアドレスを指定しないでください。

対象デバイス

RZ/T1

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。

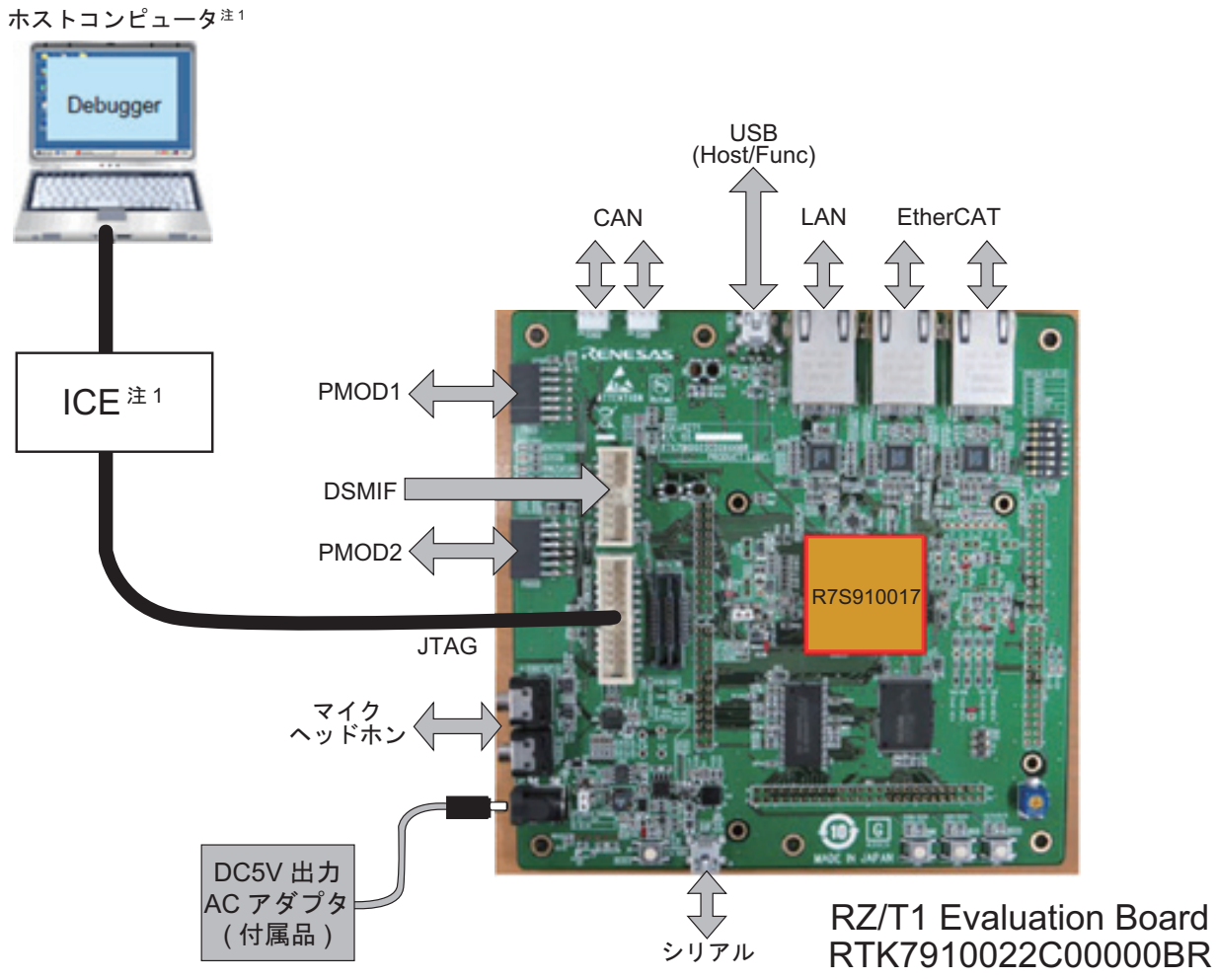
目　次

1.　仕様

表 1.1 に使用する周辺機能と用途を、図 1.1 にサンプルコード実行時の動作環境を示します。

表 1.1　使用する周辺機能と用途

周辺機能	用途
RIIC Ch(0)	I²C 通信
消費電力低減機能	RIIC モジュールの起動／停止制御（MSTPCRB3）
割り込みコントローラ（ICUA）	RIIC の割り込み制御 （Unit0/Unit1） データ送信終了（ベクタ 121/124） データ受信終了（ベクタ 122/125） 送信データエンプティ（ベクタ 123/126） 異常検出（ベクタ 260/261） コンペアマッチ割り込み（Unit0 ch0） コンペアマッチ割り込み（ベクタ 21）
I/O ポート（PF7, P56, P77, PA0）	LED 制御
タイマ（CMT Unit0 ch0）	1[ms]測定用タイマ



注 1.　別途お客様でご用意いただく必要があります。

図 1.1　動作環境

2. 動作環境

本アプリケーションノートのサンプルコードは、下記の環境を想定しています。

表2.1 動作環境

項目	内容
使用マイコン	RZ/T1グループ
動作周波数	CPUCLK = 450MHz
動作電圧	3.3V
統合開発環境	IARシステムズ製 Embedded Workbench® for Arm Version 8.20.2 Arm製 DS-5™ 5.26.2 RENESAS製 e2studio 6.1.0
動作モード	SPIブートモード 16ビットバスブートモード
使用ボード	RZ/T1 Evaluation Board （RTK7910022C00000BR）
使用デバイス （ボード上で使用する機能）	• NORフラッシュメモリ （CS0、CS1空間に接続） メーカ名：Macronix International Co..、型名：MX29GL512FLT2I-10Q • SDRAM （CS2、CS3空間に接続） メーカ名：Integrated Silicon Solution Inc、型名：IS42S16320D-7TL • シリアルフラッシュメモリ メーカ名：Macronix International Co..、型名：MX25L51245G • EEPROM メーカ名：Renesas Electronics Co..、型名：R1EX24016ASAS0A • LED LED0～3（PF7、P56、P77、PA0）

3. 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。

- RZ/T1 グループ 初期設定アプリケーションノート（R01AN2554JJ）

注 . 本アプリケーションノートで記載しないレジスタに関しては、RZ/T1 グループ 初期設定アプリケーションノートで設定した値のまま使用します。

4. 周辺機能説明

動作モード、I²C バスインタフェース（RIICa）、消費電力低減機能、割り込みコントローラ（ICUA）、汎用入出力ポート、コンペアマッチタイマ（CMT）についての基本的な内容は、RZ/T1 グループ・ユーザーズマニュアル ハードウェア編を参照してください。

5. ハードウェア説明

5.1 ハードウェア構成例

図 5.1 にハードウェア構成例を示します。

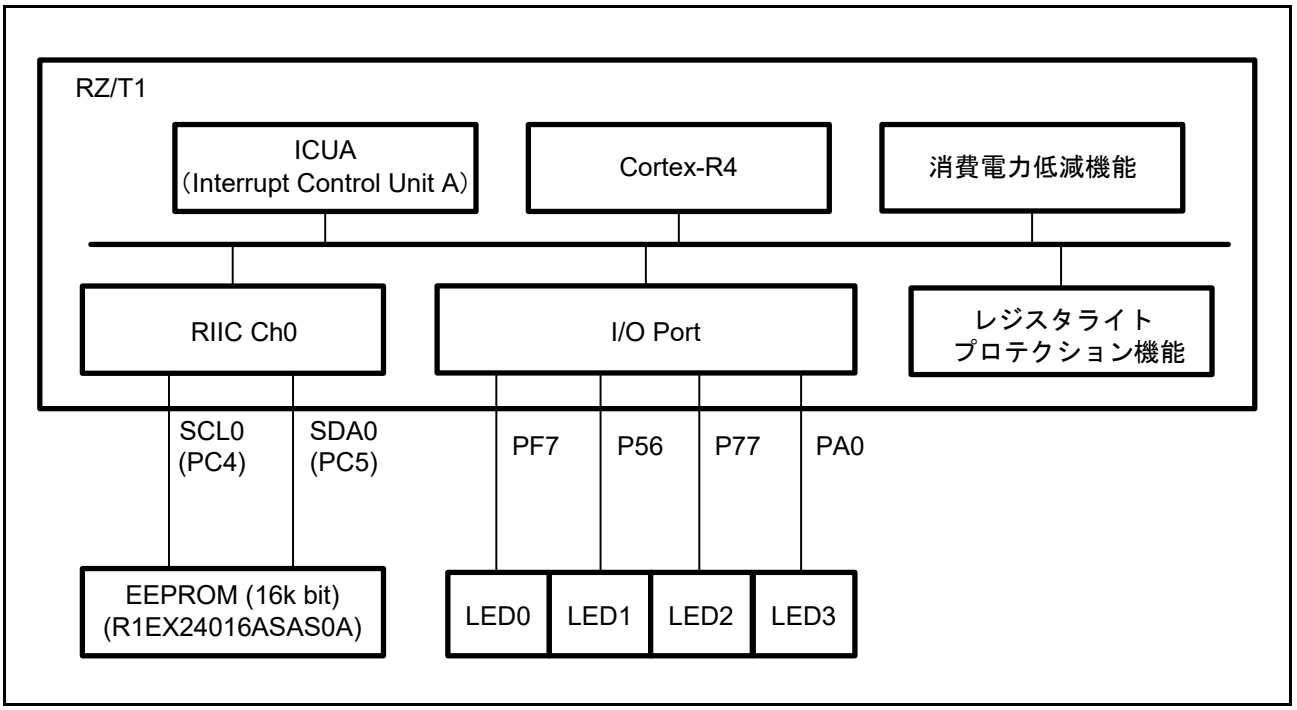


図 5.1 ハードウェア構成例

5.2 使用端子一覧

表 5.1 に使用端子と機能を示します。

表5.1 使用端子と機能

端子名	入出力	内容
SCL0（PC4）	入出力	I²Cのクロックライン
SDA0（PC5）	入出力	I²Cのデータライン
PF7	出力	LED0制御
P56	出力	LED1制御
P77	出力	LED2制御
PA0	出力	LED3制御

6.　ソフトウェア説明

6.1　動作概要

RIIC サンプルプログラムの機能概要を**表 6.1　動作概要**に示します。また、**図 6.1** にシステムブロック図を示します。

表6.1　　動作概要

機能	概要
兼用端子設定	• PC4、PC5をSCL0、SDA0に設定
RIICの通信チャネル	• EEPROMが接続されているチャネル0に設定
割り込み要因（割り込み優先度）	• RIICモジュール 送信終了（1）／受信終了（1）／送信バッファエンプティ（1）／異常検出（1） • CMTモジュール（1[ms]検出用） コンペアマッチ（15）
転送速度設定	• 400[kbps]
動作モード	• マスタ送信／マスタ受信
動作概要	1. EEPROMの全内容をバックアップ（RAM） 2. EEPROMの全体に0xFFを書き込む 3. EEPROMの全体に0xA5を書き込む 4. 書き込み内容の確認 5. EEPROMを変更前の内容を書き戻す （EEPROMへのRead/Writeアクセスを行う毎に1[ms]のインタバールを入れる）
動作結果表示	• LED0点灯 RIICの通信異常を検出 • LED1点灯 テストパターンが一致 • LED2点灯 EEPROMへwrite中 • LED3点灯 EEPROMのread中

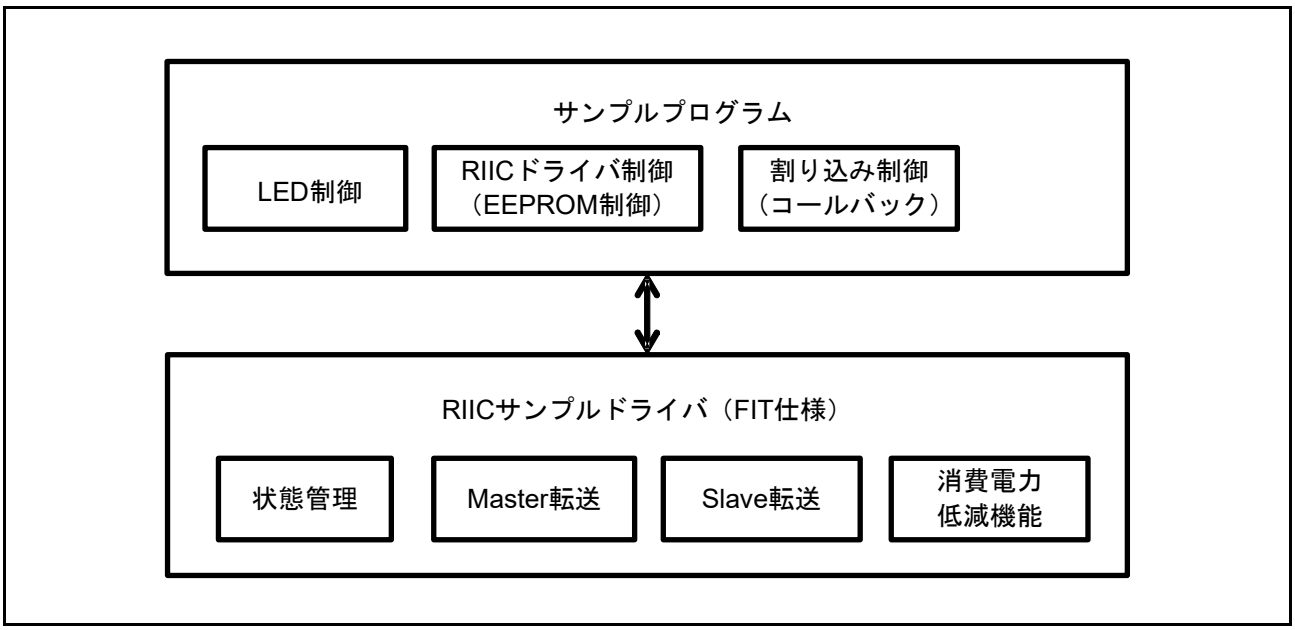


図 6.1　　システムブロック図

6.1.1 プロジェクト設定

開発環境となる EWARM 上で使用されるプロジェクト設定については、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.1.2 使用準備

本サンプルプログラムの実行準備は必要ありません。

6.1.3　RIIC サンプルドライバ動作概要

(1)　RIIC サンプルドライバ状態遷移図

図 6.2 に RIIC サンプルドライバの状態遷移図を示します。

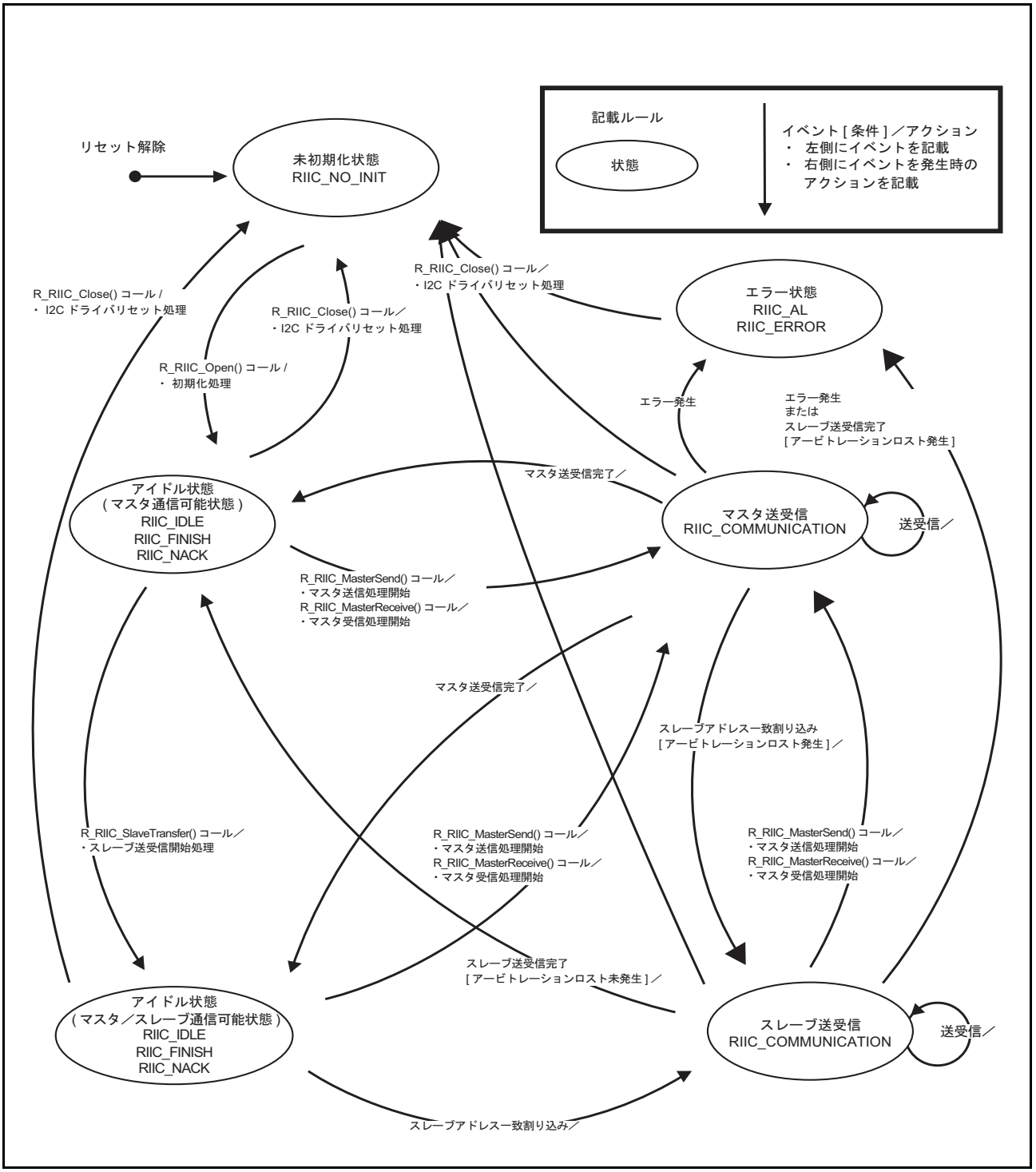


図 6.2　　RIIC サンプルドライバの状態遷移図

(2)　RIIC サンプルドライバ状態遷移時の各フラグ

I²C 通信情報構造体メンバには、デバイス状態フラグ（dev_sts）があります。デバイス状態フラグには、そのデバイスの通信状態が格納されます。また、このフラグにより、同一チャネル上の複数のスレーブデバイスの制御を行うことができます。**表 6.2** に状態遷移時のデバイス状態フラグの一覧を示します。

表6.2　　状態遷移時のデバイス状態フラグの一覧

状態	デバイス状態フラグ（dev_sts）
未初期化状態	RIIC_NO_INIT
アイドル状態	RIIC_IDLE RIIC_FINISH RIIC_NACK
通信中 （マスタ送信、マスタ受信、 スレーブ送信、スレーブ受信）	RIIC_COMMUNICATION
アービトレーションロスト検出状態	RIIC_AL
エラー	RIIC_ERROR

(3)　RIIC サンプルドライバアービトレーションロスト検出機能

本モジュールは、以下に示すアービトレーションロストを検出することができます。なお、RIIC は、以下に加えて、スレーブ送信時におけるアービトレーションロストの検出にも対応していますが、本モジュールは対応していません。

(i) バスビジー状態で、スタートコンディションを発行したとき

既に他のマスタデバイスがスタートコンディションを発行して、バスを占有している状態（バスビジー状態）でスタートコンディションを発行すると、本モジュールはアービトレーションロストを検出します。

(ii) バスビジー状態ではないが、他のマスタより遅れてスタートコンディションを発行したとき

本モジュールは、スタートコンディションを発行するとき、SDA ラインを Low にしようとします。しかし、他のマスタデバイスがこれよりも早くスタートコンディションを発行した場合、SDA ライン上の信号レベルは、本モジュールが出力したレベルと一致しなくなります。このとき、本モジュールはアービトレーションロストを検出します。

(iii) スタートコンディションが同時に発行されたとき

複数のマスタデバイスが、同時にスタートコンディションを発行すると、それぞれのマスタデバイス上でスタートコンディションの発行が正常に終了したと判断されることがあります。その後、それぞれのマスタデバイスは通信を開始しますが、以下に示す条件が成立すると、本モジュールはアービトレーションロストを検出します。

(a)　それぞれのマスタデバイスが送信するデータが異なる場合
データ通信中、本モジュールはSDA ライン上の信号レベルと、本モジュールが出力したレベルを比較しています。そのため、スレーブアドレス送信を含むデータ送信中に、SDA ライン上と本モジュールが出力したレベルが一致しなくなると、本モジュールはその時点でアービトレーションロストを検出します。

(b)　それぞれのマスタデバイスが送信するデータは同じだが、データの送信回数が異なる場合
上記a.に合致しない場合（スレーブアドレスおよび送信データが同じ）、本モジュールはアービトレーションを検出しませんが、それぞれのマスタデバイスがデータを送信する回数が異なる場合であれば、本モジュールはアービトレーションロストを検出します。

6.2 メモリマップ

RZ/T1 グループのアドレス空間と RZ/T1 評価ボードのメモリマッピングについては、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.2.1 サンプルプログラムのセクション配置

サンプルプログラムで使用するセクションおよびサンプルプログラムの初期状態のセクション配置（ロードビュー）、スキャッタローディング機能を使用後のセクション配置（実行ビュー）は、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.2.2 MPU の設定

MPU の設定は、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.2.3 例外処理ベクタテーブル

例外処理ベクタテーブルについては、RZ/T1 グループ 初期設定 アプリケーションノートに記載しています。

6.3 使用割り込み一覧

表 6.3 にサンプルコードで使用する割り込みを示します。

表6.3 サンプルコードで使用する割り込み

割り込み（要因ID）	優先度	処理概要
unit0通信エラー（EEI）	RIIC_CFG_CH0_INT_PRIORITY	通信エラー／イベント発生処理（ベクタ番号：260） • アービトレーションロスト検出 • NACK検出 • タイムアウト検出 • スタートコンディション検出 （リスタートコンディション含む） • ストップコンディション検出
unit0受信データフル（RXI）	RIIC_CFG_CH0_INT_PRIORITY	受信データフル処理 （ベクタ番号：122）
unit0送信データエンプティ（TXI）	RIIC_CFG_CH0_INT_PRIORITY	送信データエンプティ処理 （ベクタ番号：123）
unit0送信終了 （TEI）	RIIC_CFG_CH0_INT_PRIORITY	送信終了処理 （ベクタ番号：121）
コンペアマッチ割り込み（CMI0）	ICU_PRIORITY_15	1[ms]計測処理（ベクタ番号：21）

6.4 固定幅整数一覧

表 6.4 にサンプルコードで使用する固定幅整数を示します。

表 6.4 サンプルコードで使用する固定幅整数

シンボル	内容
int8_t	8 ビット整数、符号あり（標準ライブラリにて定義）
int16_t	16 ビット整数、符号あり（標準ライブラリにて定義）
int32_t	32 ビット整数、符号あり（標準ライブラリにて定義）
int64_t	64 ビット整数、符号あり（標準ライブラリにて定義）
uint8_t	8 ビット整数、符号なし（標準ライブラリにて定義）
uint16_t	16 ビット整数、符号なし（標準ライブラリにて定義）
uint32_t	32 ビット整数、符号なし（標準ライブラリにて定義）
uint64_t	64 ビット整数、符号なし（標準ライブラリにて定義）

6.5　定数／エラーコード一覧

表 6.5 にサンプルコードで使用する定数、表 6.6 にサンプルコードのエラーコードを示します。
表 6.7 にコンパイル時にコンフィグ可能な定数を示します。

表6.5　サンプルコードで使用する定数

定数名	設定値	内容
RIIC_NO_INIT注1	0	未初期化状態
RIIC_IDLE注1	1	アイドル状態
RIIC_FINISH注1	2	アイドル状態
RIIC_NACK注1	3	アイドル状態
RIIC_COMMUNICATION注1	4	マスタ送受信状態／スレーブ送受信状態
RIIC_AL注1	5	アービトレーションロスト検出状態
RIIC_ERROR注1	6	エラー状態
RIIC_GEN_START_CON注2	(uint8_t)(0x01)	スタートコンディションの生成
RIIC_GEN_STOP_CON注2	(uint8_t)(0x02)	ストップコンディションの生成
RIIC_GEN_RESTART_CON注2	(uint8_t)(0x04)	リスタートコンディションの生成
RIIC_GEN_SDA_HI_Z注2	(uint8_t)(0x08)	SDA 端子を Hi-Z 出力
RIIC_GEN_SCL_ONESHOT注2	(uint8_t)(0x10)	SCL クロックのワンショット出力
RIIC_GEN_RESET注2	(uint8_t)(0x20)	RIIC のモジュールリセット
FIT_NO_PTR	(void *)0	FIT で定義されている NULL ポインタ

注1.　riic_ch_dev_status_t 型の値として使用
注2.　R_RIIC_Control()の出力パターンとして使用

表6.6　サンプルコードのエラーコード

定数名	設定値	内容
RIIC_SUCCESS	0U	関数コールが正常にできた場合
RIIC_ERR_LOCK_FUNC	1U	他のモジュールで RIIC が使用されている場合
RIIC_ERR_INVALID_CHAN	2U	存在しないチャネルを指定した場合
RIIC_ERR_INVALID_ARG	3U	不正な引数を設定した場合
RIIC_ERR_NO_INIT	4U	未初期化状態の場合
RIIC_ERR_BUS_BUSY	5U	バスビジーの場合
RIIC_ERR_AL	6U	アービトレーションロスト検出状態で関数コールした場合
RIIC_ERR_OTHER	7U	その他エラー

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_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。

表6.7　　コンパイル時の設定（１／２）

定義	内容
RIIC_CFG_PARAM_CHECKING_ENABLE 注.　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”を選択すると、パラメータチェック処理をコードから省略できるため、コードサイズが削減できます。 “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。
RIIC_CFG_PCLK_Hz 注.　デフォルト値は“75000000”	RIIC0モジュールに供給されるクロック（PCLK）の周波数を設定してください。 “RIIC_CFG_CH0_kBPS”と“RIIC_CFG_PCLK_Hz”の設定値からビットレートレジスタおよび内部基準クロック選択ビットへの設定値が算出されます。
RIIC_CFG_CH0_INCLUDED 注.　デフォルト値は“1”	該当チャネルを使用するかを選択できます。 該当チャネルを使用しない場合は“0”に設定してください。 “0”の場合、該当チャネルに関する処理をコードから省略します。 “1”の場合、該当チャネルに関する処理をコードに含めます。
RIIC_CFG_CH0_kBPS 注.　デフォルト値は“400”	RIIC0の通信速度を設定できます。 “RIIC_CFG_CH0_kBPS”と“RIIC_CFG_PCLK_Hz”の設定値からビットレートレジスタおよび内部基準クロック選択ビットへの設定値が算出されます。 “400”以下の値を設定してください。
RIIC_SCL_100K_UP_TIME 注.　デフォルト値は“1000E-9”	RIIC0の通信速度が1-100[KBPS]時のSCLの立ち上がり時間[s]を設定してください。（double型で指定）
RIIC_SCL_100K_DOWN_TIME 注.　デフォルト値は“300E-9”	RIIC0の通信速度が1-100[KBPS]時のSCLの立ち下がり時間[s]を設定してください。（double型で指定）
RIIC_SCL_400K_UP_TIME 注.　デフォルト値は“175E-9”	RIIC0の通信速度が101-400[KBPS]時のSCLの立ち上がり時間[s]を設定してください。（double型で指定）
RIIC_SCL_400K_DOWN_TIME 注.　デフォルト値は“175E-9”	RIIC0の通信速度が101-400[KBPS]時のSCLの立ち下がり時間[s]を設定してください。（double型で指定）
RIIC_CFG_CH0_DIGITAL_FILTER 注.　デフォルト値は“2”	ノイズフィルタの段数を選択できます。 “0”の場合、ノイズフィルタは無効となります。 “1”～“4”の場合、選択した段数のフィルタが有効になるようノイズフィルタ段数選択ビットおよびデジタルノイズフィルタ回路有効ビットの設定値が選択されます。
RIIC_CFG_CH0_SCL0 注.　デフォルト値は“1”	RIIC0のSCLの出力端子を選択できます。 選択した端子をSCL端子とする処理をコードに含めます。 “0”の場合、SCL0端子の設定処理をコードから省略します。 “1”の場合、PC4をSCL0端子として設定します。
RIIC_CFG_CH0_SDA0 注.　デフォルト値は“1”	RIIC0のSDAの出力端子を選択できます。 選択した端子をSDA端子とする処理をコードに含めます。 “0”の場合、SDA0端子の設定処理をコードから省略します。 “1”の場合、PC5をSDA0 端子として設定します。
RIIC_CFG_CH0_MASTER_MODE 注.　デフォルト値は“1”	マスタアービトレーションロスト検出機能の有効／無効を選択できます。 マルチマスタで使用する場合は、“1”(有効)にしてください。 “0”の場合、マスタアービトレーションロスト検出を無効にします。 “1”の場合、マスタアービトレーションロスト検出を有効にします。
RIIC_CFG_CH0_SLV_ADDR0_FORMAT注1 RIIC_CFG_CH0_SLV_ADDR1_FORMAT注2 RIIC_CFG_CH0_SLV_ADDR2_FORMAT注2 注1.　デフォルト値は“1” 注2.　デフォルト値は“0”	スレーブアドレスのフォーマットを7 ビット／10 ビットから選択できます。 “0”の場合、スレーブアドレスを設定しません。 “1”の場合、7 ビットアドレスフォーマットに設定します。 “2”の場合、10 ビットアドレスフォーマットに設定します。
RIIC_CFG_CH0_SLV_ADDR0注1 RIIC_CFG_CH0_SLV_ADDR1注2 RIIC_CFG_CH0_SLV_ADDR2注2 注1.　デフォルト値は“0x0025” 注2.　デフォルト値は“0x0000”	スレーブアドレスを設定します。 “RIIC_CFG_CH0_SLV_ADDRi_FORMAT”の設定値によって、設定可能範囲が変わります。 “RIIC_CFG_CH0_SLV_ADDRi_FORMAT”が“0”の場合は、設定値は無効となります。 “1”の場合は、設定値の下位7 ビットが有効となります。 “2”の場合は、設定値の下位10 ビットが有効となります。

表6.7 コンパイル時の設定（２／２）

定義	内容
RIIC_CFG_CH0_SLV_GCA_ENABLE 注. デフォルト値は“0”	ゼネラルコールアドレスの有効／無効が選択できます。 “0”の場合、ゼネラルコールアドレスを無効にします。 “1”の場合、ゼネラルコールアドレスを有効にします。
RIIC_CFG_CH0_INT_PRIORITY 注. デフォルト値は“1”	通信エラー／イベント発生割り込み(EEI)、受信データフル割り込み(RXI)、送信データエンプティ割り込み(TXI)、送信終了割り込み(TEI)の優先レベルを選択できます。 “1”～“15”の範囲で設定してください。
RIIC_CFG_BUS_CHECK_COUNTER 注. デフォルト値は“1000”	RIIC のAPI 関数のバスチェック処理時の、タイムアウトカウンタ(バス確認回数)を設定できます。 “0xFFFFFFFF”以下の値を設定してください。 バスチェック処理は、 • スタートコンディション生成前 • ストップコンディション検出後 • RIIC 制御機能(R_RIIC_Control 関数)を使用した各コンディションおよびSCL ワンショットパルスの生成後 に行います。 バスチェック処理では、バスビジー時、バスフリーになるまでタイムアウトカウンタをデクリメントします。“0”になるとタイムアウトと判断し、戻り値でエラー（Busy）を返します。 注. バスロックなどでロックされないようにするためのカウンタであるため、相手デバイスがSCL 端子を“L”ホールドする時間以上になるよう値を設定してください。 「タイムアウト時間(ns) ≒ (1 / ICLK(Hz)) * カウンタ値 * 10」

6.6 構造体／共用体／列挙型一覧

図 6.3 にサンプルコードで使用する構造体／共用体／列挙型を示します。

```
/* ---- Return Value of IIC Driver API. ---- */
typedef enum
{
    RIIC_SUCCESS = 0U,              /* Successful operation                          */
    RIIC_ERR_LOCK_FUNC,             /* Lock has already been acquired by another task.  */
    RIIC_ERR_INVALID_CHAN,          /* None existent channel number                  */
    RIIC_ERR_INVALID_ARG,           /* Parameter error                               */
    RIIC_ERR_NO_INIT,               /* Uninitialized state                           */
    RIIC_ERR_BUS_BUSY,              /* Channel is on communication.                  */
    RIIC_ERR_AL,                    /* Arbitration lost error                        */
    RIIC_ERR_OTHER                  /* Other error                                   */
} riic_return_t;

/* ---- IIC Channel status type. ---- */
typedef uint8_t        riic_ch_dev_status_t;

/* ---- Callback function type. ---- */
typedef void (*riic_callback)(void);          /* Callback function type              */

/* ---- IIC Information structure type. ---- */
typedef volatile struct
{
  uint8_t                rsv2;            /* reserved                                 */
  uint8_t                rsv1;            /* reserved                                 */
  riic_ch_dev_status_t   dev_sts;         /* Device status flag                       */
  uint8_t                ch_no;           /* Channel No.                              */
  riic_callback          callbackfunc;    /* Callback function                        */
  uint32_t               cnt2nd;          /* 2nd Data Counter                         */
  uint32_t               cnt1st;          /* 1st Data Counter                         */
  uint8_t *              p_data2nd;       /* Pointer for 2nd Data buffer              */
  uint8_t *              p_data1st;       /* Pointer for 1st Data buffer              */
  uint8_t *              p_slv_adr;       /* Pointer for Slave address buffer         */
} riic_info_t;

/* ---- IIC Status type. ---- */
typedef union
{
    uint32_t             LONG;
    struct
  {
        uint32_t         rsv1:20;         /* reserve                                  */
        uint32_t         AL:1;            /* Arbitration lost detection flag          */
        uint32_t         rsv2:4;          /*                                          */
        uint32_t         SCLO:1;          /* SCL pin output control status            */
        uint32_t         SDAO:1;          /* SDA pin output control status            */
        uint32_t         SCLI:1;          /* SCL pin level                            */
        uint32_t         SDAI:1;          /* SDA pin level                            */
        uint32_t         NACK:1;          /* NACK detection flag                      */
        uint32_t         rsv3:1;          /* reserve                                  */
        uint32_t         BSY:1;           /* Bus status flag                          */
    }BIT;
}riic_mcu_status_t;
```

図 6.3 サンプルコードで使用する構造体 / 共用体 / 列挙型

6.7 大域変数一覧

表 6.8 に大域変数一覧を示します。

表6.8 大域変数一覧

型	変数名	内容	使用関数
riic_info_t *	gp_riic_info_m[]	I2C 通信情報構造体（マスター）注1	R_RIIC_MasterSend() R_RIIC_MasterReceive()
riic_info_t *	gp_riic_info_s[]	I2C 通信情報構造体（スレーブ）注1	R_RIIC_SlaveTransfer()
riic_ch_dev_status_t	g_riic_ChStatus[]	ドライバ状態注1	外部公開変数
riic_api_event_t	g_riic_api_Event[]	イベント注1	R_RIIC_Open() R_RIIC_MasterSend() R_RIIC_MasterReceive() R_RIIC_SlaveTransfer()
riic_api_info_t	g_riic_api_Info[]	内部管理用注1	—
volatile riic_callback	riic_callbackfunc_m	内部管理用	—
volatile riic_callback	riic_callbackfunc_s	内部管理用	—
static const riic_mtx_t	gc_riic_mtx_tbl[][]	状態遷移テーブル	r_riic_rzt1.c R_RIIC_Open() R_RIIC_MasterSend() R_RIIC_MasterReceive() R_RIIC_SlaveTransfer()
static uint8_t	s_riic_backup[2048]	EEPROMのバックアップ用領域	main.c main()
static volatile uint32_t	wait_flag	RIICサンプルドライバの割り込み待ちフラグ	main.c main()
static volatile uint32_t	wait_cmt_flag	コンペアマッチ割り込み待ちフラグ	main.c main()
static riic_info_t	riic_info	I2C通信情報構造体の実体	main.c main()
static uint8_t	init_ram[16]	EEPROM初期化データ	main.c main()

注1. チャネル数分の配列で宣言

6.8 関数一覧

表 6.9 に関数を示します。

表6.9 関数一覧

関数名	ページ番号
R_RIIC_Open	20
R_RIIC_MasterSend	21
R_RIIC_MasterReceive	28
R_RIIC_SlaveTransfer	33
R_RIIC_GetStatus	38
R_RIIC_Control	39
R_RIIC_Close	41
R_RIIC_GetVersion	42
main	42

6.9　関数仕様

サンプルコードの関数仕様を示します。

6.9.1　R_RIIC_Open

R_RIIC_Open		
概　要	本モジュールを使用する際に最初に使用する関数です。	
ヘッダ	r_riic_rx_if.h	
宣　言	riic_return_t R_RIIC_Open(riic_info_t * p_riic_info)	
説　明	RIIC の通信を開始するための初期設定をします。引数で指定した RIIC のチャネルを設定します。チャネルの状態が ” 未初期化状態（RIIC_NO_INIT）” の場合、次の処理を行います。 - 状態フラグの設定 - ポートの入出力設定 - I2C 出力ポートの割り当て - RIIC のモジュールストップ状態の解除 - API で使用する変数の初期化 - RIIC 通信で使用する RIIC レジスタの初期化 - RIIC 割り込みの禁止 *p_riic_info I2C 通信情報構造体のポインタ。 この構造体のうち、本関数で使用するメンバのみを以下に示します。この構造体の詳細については図 6.3 を参照してください。 下記のうち、API 実行中に値が更新される引数には、“ 更新あり ” と記載しています。 riic_ch_dev_status_t dev_sts; /* デバイス状態フラグポインタ（更新あり）*/ uint8_t ch_no; /* チャネル番号 */	
引　数	riic_info_t * p_riic_info	: RIIC 通信情報構造体のポインタ
リターン値	RIIC_SUCCESS	: 問題なく処理が完了した場合
	RIIC_ERR_LOCK_FUNC	: 他のタスクが API をロックしている場合
	RIIC_ERR_INVALID_CHAN	: 存在しないチャネルの場合
	RIIC_ERR_INVALID_ARG	: 不正な引数の場合
	RIIC_ERR_OTHER	: 現状態に該当しない不正なイベントが発生した場合
補足	なし	

Example

```
volatile riic_return_t ret;
riic_info_t iic_info_m;

iic_info_m.dev_sts = 0;
iic_info_m.ch_no = 0;

ret = R_RIIC_Open(&iic_info_m);
```

6.9.2　R_RIIC_MasterSend

R_RIIC_MasterSend	
概　要	マスタデバイスとして、送信を開始する際に使用する関数です。
ヘッダ	r_riic_rx_if.h
宣　言	riic_return_t R_RIIC_MasterSend(riic_info_t * p_riic_info)
説　明	RIIC のマスタ送信を開始します。引数で指定した RIIC のチャネル、送信パターンで送信します。チャネルの状態が “ アイドル状態 ”（RIIC_IDLE、RIIC_FINISH、RIIC_NACK）の場合、次の処理を行います。 - 状態フラグの設定 - API で使用する変数の初期化 - RIIC 割り込みの許可 - スタートコンディションの生成 *p_riic_info I²C 通信情報構造体のポインタ。引数によって、送信パターン（4 パターンあります）を変更できます。 各送信パターンの指定方法および引数の設定可能範囲は、表 6.10 を参照ください。また、送信パターンの波形のイメージは図 6.4 ～図 6.7 を参照ください。 この構造体のうち、本関数で使用するメンバのみを以下に示します。この構造体の詳細については図 6.3 を参照してください。 スレーブ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。 下記のうち、API 実行中に値が更新される引数には、“ 更新あり ” と記載しています。 riic_ch_dev_status_t dev_sts; /* デバイス状態フラグ (更新あり) */ uint8_t ch_no; /* チャネル番号 */ riic_callback callbackfunc; /* コールバック関数 */ uint32_t cnt2nd; /* 2nd データカウンタ (バイト数)(パターン 1、2 のみ更新あり) */ uint32_t cnt1st; /* 1st データカウンタ (バイト数)(パターン 1 のみ更新あり) */ uint8_t * p_data2nd; /* 2nd データ格納バッファポインタ */ uint8_t * p_data1st; /* 1st データ格納バッファポインタ */ uint8_t * p_slv_adr; /* スレーブアドレスのバッファポインタ */
引　数	riic_info_t * p_riic_info　：RIIC 通信情報構造体のポインタ
リターン値	RIIC_SUCCESS　：問題なく処理が完了した場合 RIIC_ERR_INVALID_CHAN　：存在しないチャネルの場合 RIIC_ERR_INVALID_ARG　：不正な引数の場合 RIIC_ERR_NO_INIT　：初期設定ができていない場合（未初期化状態） RIIC_ERR_BUS_BUSY　：バスビジーの場合 RIIC_ERR_AL　：アービトレーションエラーが発生した場合 RIIC_ERR_OTHER　：現状態に該当しない不正なイベントが発生した場合
補足	なし

Example

```
/* for MasterSend(Pattern 1) */
#include "r_riic_rx_if.h"

void CallbackMaster(void);
void main(void);

void main(void)
{
   volatile riic_return_t ret;
   riic_info_t iic_info_m;
   uint8_t addr_eeprom[1]={0x50};
   uint8_t access_addr1[1]={0x00};
   uint8_t mst_send_data[5]={0x81,0x82,0x83,0x84,0x85};

   /* Sets IIC Information for sending pattern 1. */
   iic_info_m.dev_sts = 0;
   iic_info_m.ch_no = 0;
   iic_info_m.callbackfunc = &CallbackMaster;
   iic_info_m.cnt2nd = 3;
   iic_info_m.cnt1st = 1;
   iic_info_m.p_data2nd = mst_send_data;
   iic_info_m.p_data1st = access_addr1;
   iic_info_m.p_slv_adr = addr_eeprom;

   /* RIIC open */
   ret = R_RIIC_Open(&iic_info_m);

   /* RIIC send start */
   ret = R_RIIC_MasterSend(&iic_info_m);
   while(1);
}

void CallbackMaster(void)
{
   /* callback process */
}
```

Special Notes:

送信パターンごとの引数の設定可能範囲は、下表を参照してください。

表6.10 送信パターンごとの引数の設定可能範囲

構造体メンバ	ユーザ設定可能範囲			
	マスタ送信 パターン1	マスタ送信 パターン2	マスタ送信 パターン3	マスタ送信 パターン4
*p_slv_adr	スレーブアドレス バッファポインタ	スレーブアドレス バッファポインタ	スレーブアドレス バッファポインタ	FIT_NO_PTR （注 1）
*p_data1st	[送信用]1st データ バッファポインタ	FIT_NO_PTR 注 1	FIT_NO_PTR 注 1	FIT_NO_PTR 注 1
*p_data2nd	[送信用]2nd データ バッファポインタ	[送信用]2nd データ バッファポインタ	FIT_NO_PTR 注 1	FIT_NO_PTR 注 1
cnt1st	0000 0001h ～ FFFF FFFFh 注 2	0	0	0
cnt2nd	0000 0001h ～ FFFF FFFFh 注 2	0000 0001h ～ FFFF FFFFh 注 2	0	0
callbackfunc	使用する関数名を指定してください。	使用する関数名を指定してください。	使用する関数名を指定してください。	使用する関数名を指定してください。
ch_no	00h ～ FFh	00h ～ FFh	00h ～ FFh	00h ～ FFh
dev_sts	デバイス状態 フラグ	デバイス状態 フラグ	デバイス状態 フラグ	デバイス状態 フラグ
rsv1,rsv2	予約領域 （設定無効）	予約領域 （設定無効）	予約領域 （設定無効）	予約領域 （設定無効）

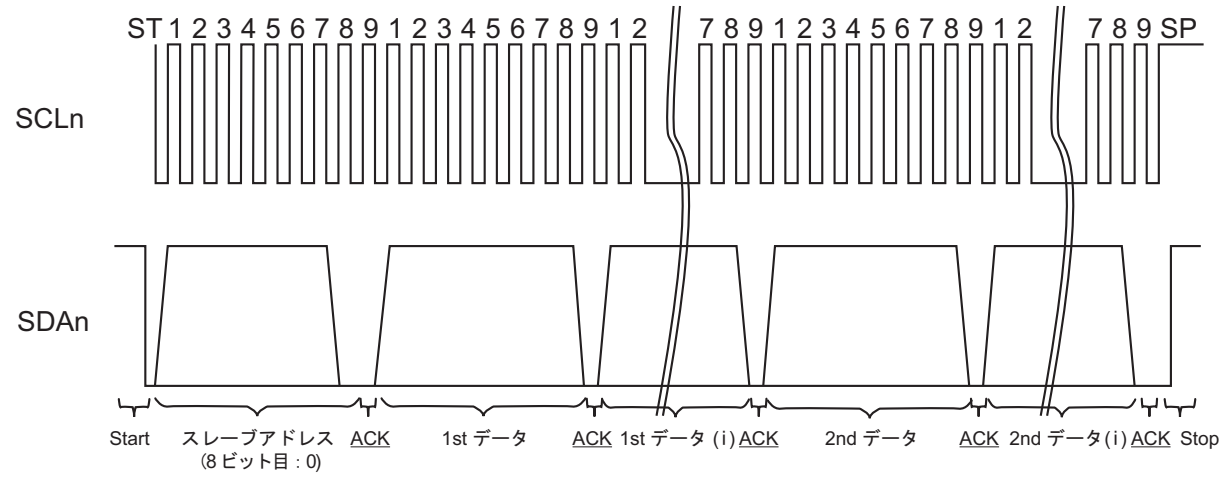
注1. パターン2、パターン3、パターン4 を使用する場合は、上表のとおり該当の構造体メンバに“FIT_NO_PTR”を入れてください。

注2. “0”は設定禁止です。

(1)　パターン 1

マスタデバイスとして、2 つのバッファのデータ（1st データと 2nd データ）をスレーブデバイスへ送信する機能です。

初めにスタートコンディション（ST）を生成し、次にスレーブデバイスのアドレスを送信します。このとき、8 ビット目は転送方向指定ビットになりますので、データ送信時には “0”（Write）を送信します。次に 1st データを送信します。1st データとは、データ送信を行う前に、事前に送信したいデータがある場合に使用します。例えばスレーブデバイスが EEPROM の場合、EEPROM 内部のアドレスを送信することができます。次に 2nd データを送信します。2nd データがスレーブデバイスへ書き込むデータになります。データ送信を開始し、全データの送信が完了すると、ストップコンディション（SP）を生成してバスを解放します。



凡例：
n: チャネル番号
ST: スタートコンディション生成
SP: ストップコンディション生成
ACK:Acknowledge“0”

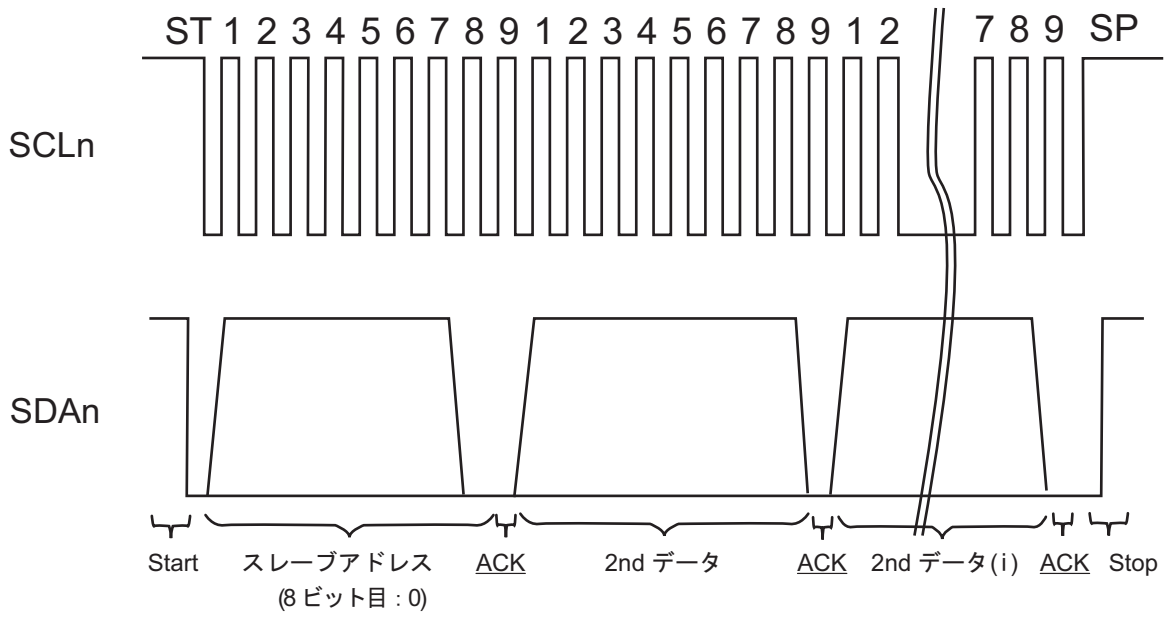
注 .　下線のある信号は、スレーブからマスタへのデータ送信です

図 6.4　　マスタ送信（パターン 1）信号図

(2) パターン 2

マスタデバイスとして、1 つのバッファのデータ（2nd データ）をスレーブデバイスへ送信する機能です。

スタートコンディション（ST）の生成からスレーブデバイスのアドレスを送信まではパターン 1 と同様に動作します。次に 1st データを送信せず、2nd データを送信します。全データの送信が完了すると、ストップコンディション（SP）を生成してバスを解放します。



凡例 :
n: チャネル番号
ST: スタートコンディション生成
SP: ストップコンディション生成
ACK:Acknowledge“0”

注 1. 下線のある信号は、スレーブからマスタへのデータ送信です

図 6.5 マスタ送信（パターン 2）信号図

(3) パターン 3

マスタデバイスとして、スレーブアドレスのみをスレーブデバイスへ送信する機能です。

スタートコンディション（ST）を生成から、スレーブアドレス送信まではパターン 1 と同様に動作します。

スレーブアドレス送信後、1st データ／ 2nd データを設定していない場合、データ送信は行わず、ストップコンディション（SP）を生成してバスを解放します。

接続されているデバイスを検索する場合や、EEPROM 書き換え状態を確認する Acknowledge Polling を行う際に有効な処理です。

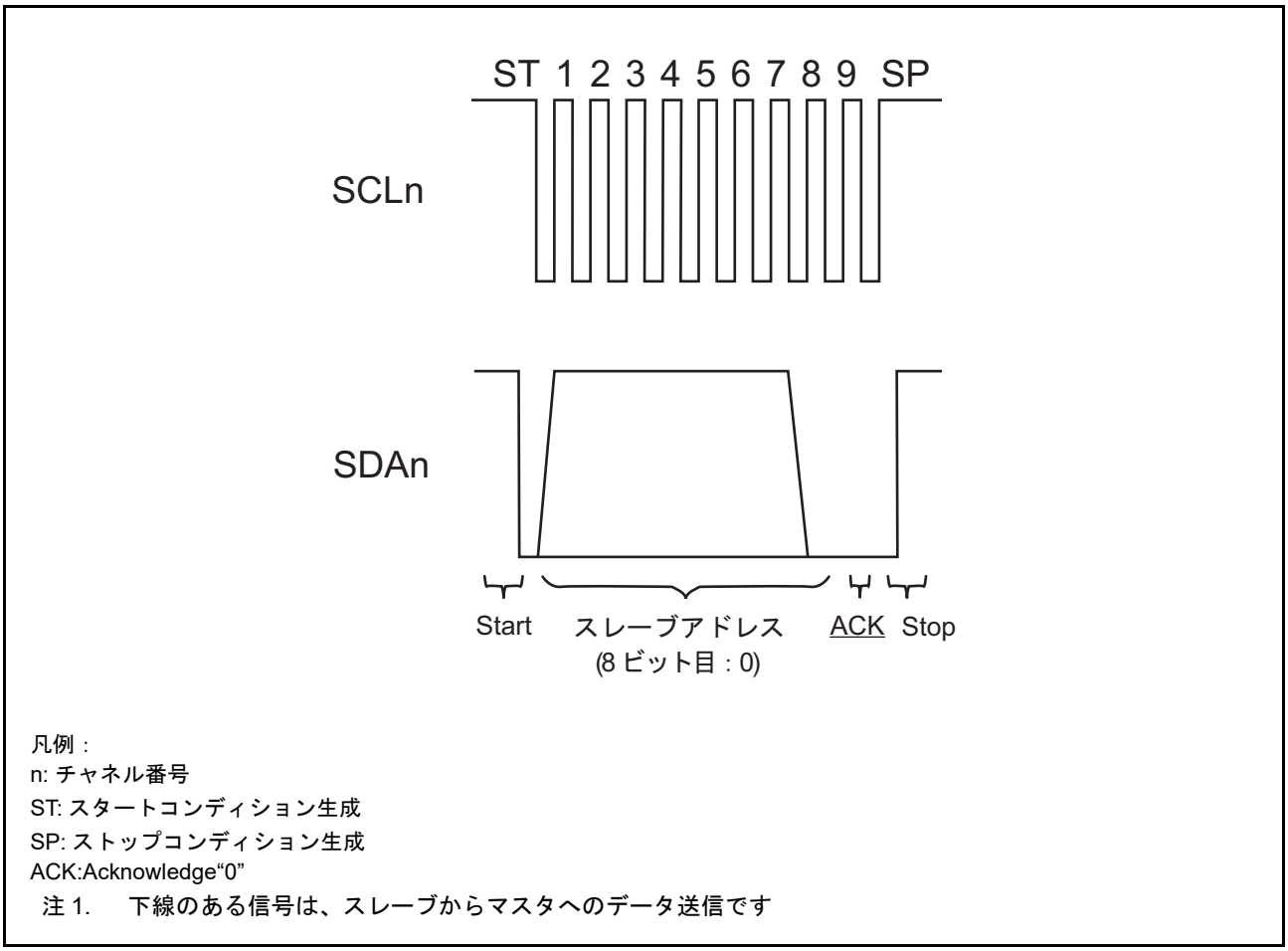


図 6.6 マスタ送信（パターン 3）信号図

(4) パターン 4

マスタデバイスとして、スタートコンディションとストップコンディションのみをスレーブデバイスへ送信する機能です。

スタートコンディション（ST）を生成後、スレーブアドレスと 1st データ／2nd データを設定していない場合、スレーブアドレス送信とデータの送信は行わず、ストップコンディション（SP）を生成してバスを解放します。バス解放のみを行いたい場合に有効な処理です。

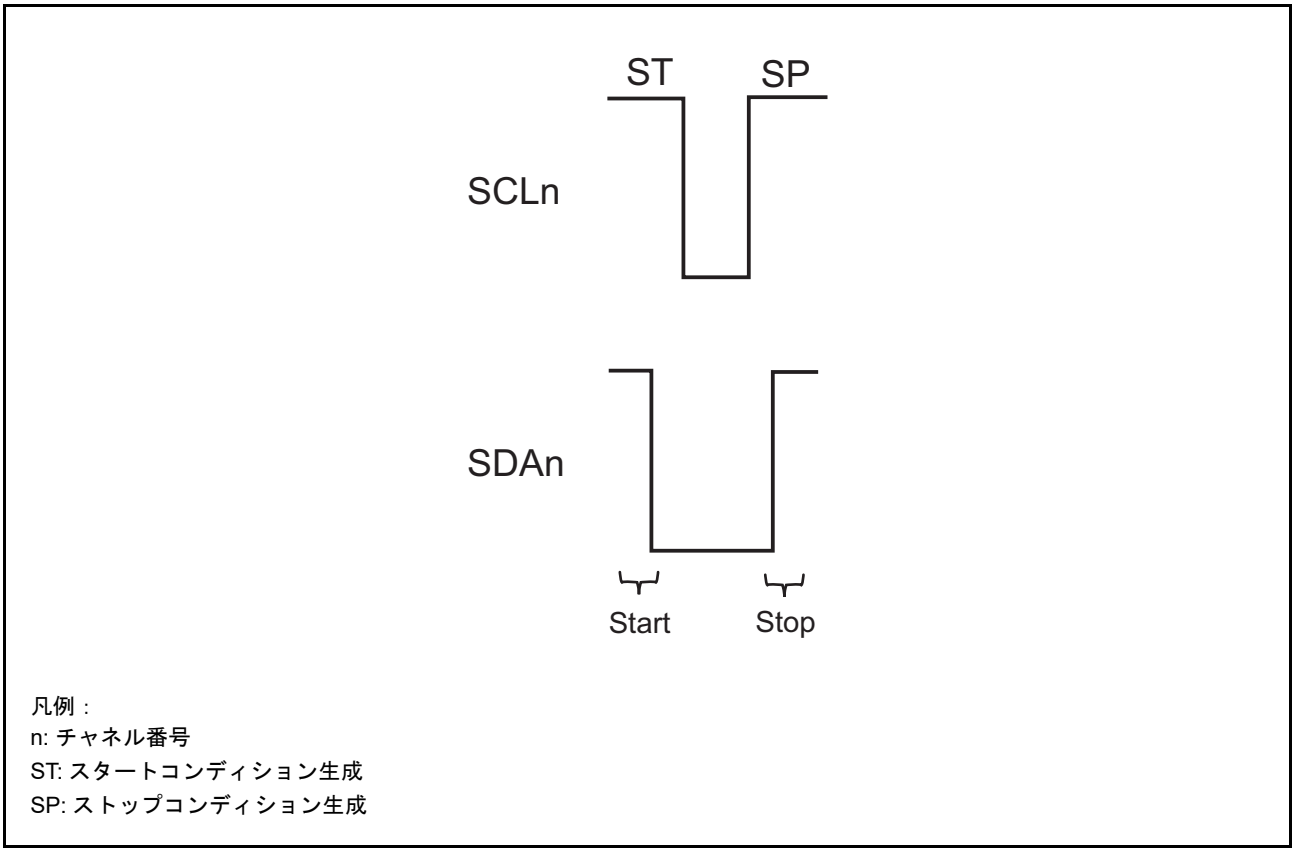


図 6.7 マスタ送信（パターン 4）信号図

6.9.3　R_RIIC_MasterReceive

R_RIIC_MasterReceive	
概　要	マスタデバイスとして、受信を開始する際に使用する関数です。
ヘッダ	r_riic_rx_if.h
宣　言	riic_return_t R_RIIC_MasterRecive(riic_info_t * p_riic_info)
説　明	RIIC のマスタ受信を開始します。引数で指定した RIIC のチャネル、受信パターンで受信します。チャネルの状態が “ アイドル状態 ”(RIIC_IDLE、RIIC_FINISH、RIIC_NACK) の場合、次の処理を行います。 - 状態フラグの設定 - API で使用する変数の初期化 - RIIC 割り込みの許可 - スタートコンディションの生成 *p_riic_info RIIC 通信情報構造体のポインタ。引数の設定によって、マスタ受信かマスタ複合を選択できます。マスタ受信およびマスタ複合の指定方法と引数の設定可能範囲は表 6.11 を参照ください。また、受信パターンの波形イメージは図 6.8、図 6.9 を参照ください。 この構造体のうち、本関数で使用するメンバのみを以下に示します。この構造体の詳細については図 6.3 を参照してください。 スレーブ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。 下記のうち、API 実行中に値が更新される引数には、“ 更新あり ” と記載しています。 riic_ch_dev_status_t dev_sts; /* デバイス状態フラグ (更新あり) */ uint8_t ch_no; /* チャネル番号 */ riic_callback callbackfunc; /* コールバック関数 */ uint32_t cnt2nd; /* 2nd データカウンタ (バイト数)(更新あり) */ uint32_t cnt1st; /* 1st データカウンタ (バイト数)(マスタ複合のみ更新あり) */ uint8_t * p_data2nd; /* 2nd データ格納バッファポインタ */ uint8_t * p_data1st; /* 1st データ格納バッファポインタ */ uint8_t * p_slv_adr; /* スレーブアドレスのバッファポインタ */
引　数	riic_info_t * p_riic_info　：RIIC 通信情報構造体のポインタ
リターン値	RIIC_SUCCESS　：問題なく処理が完了した場合 RIIC_ERR_INVALID_CHAN　：存在しないチャネルの場合 RIIC_ERR_INVALID_ARG　：不正な引数の場合 RIIC_ERR_NO_INIT　：初期設定ができていない場合（未初期化状態） RIIC_ERR_BUS_BUSY　：バスビジーの場合 RIIC_ERR_AL　：アービトレーションエラーが発生した場合 RIIC_ERR_OTHER　：現状態に該当しない不正なイベントが発生した場合
補足	なし

Example

```
/* for MasterReceive(combination mode) */
#include "r_riic_rx_if.h"

void CallbackMaster(void);
void main(void);

void main(void)
{
    volatile riic_return_t ret;
    riic_info_t iic_info_m;
    uint8_t addr_eeprom[1]={0x50};
    uint8_t access_addr1[1]={0x00};
    uint8_t mst_store_area[5]={0xFF,0xFF,0xFF,0xFF,0xFF};

    /* Sets IIC Information. */
    iic_info_m.dev_sts = 0;
    iic_info_m.ch_no = 0;
    iic_info_m.callbackfunc = &CallbackMaster;
    iic_info_m.cnt2nd = 3;
    iic_info_m.cnt1st = 1;
    iic_info_m.p_data2nd = mst_store_area;
    iic_info_m.p_data1st = access_addr1;
    iic_info_m.p_slv_adr = addr_eeprom;

    /* RIIC open */
    ret = R_RIIC_Open(&iic_info_m);

    /* RIIC receive start */
    ret = R_RIIC_MasterReceive(&iic_info_m);

    while(1);
}

void CallbackMaster(void)
{
  /* callback process */
}
```

Special Notes:

受信パターンごとの引数の設定可能範囲は、下表を参照してください。

表 6.11 受信パターンごとの引数の設定可能範囲

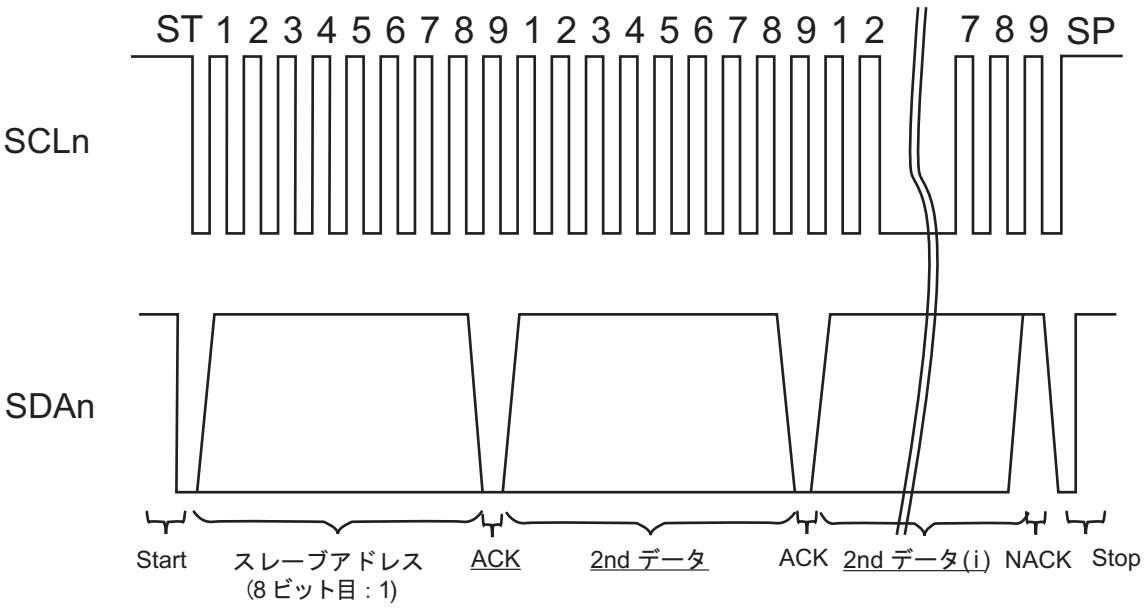
構造体メンバ	ユーザ設定可能範囲	
	マスタ受信	マスタ複合
*p_slv_adr	スレーブアドレスバッファポインタ	スレーブアドレスバッファポインタ
*p_data1st	未使用（設定無効）	[送信用]1st データバッファポインタ
*p_data2nd	[受信用]2nd データバッファポインタ	[受信用]2nd データバッファポインタ
dev_sts	デバイス状態フラグ	デバイス状態フラグ
cnt1st 注 1	0	0000 0001h ～ FFFF FFFFh 注 2
cnt2nd	0000 0001h ～ FFFF FFFFh 注 2	0000 0001h ～ FFFF FFFFh 注 2
callbackfunc	使用する関数名を指定してください。	使用する関数名を指定してください。
ch_no	00h ～ FFh	00h ～ FFh
rsv1,rsv2	予約領域（設定無効）	予約領域（設定無効）

注 1. 1st データが “0” か “0” 以外かで受信パターンが決まります。
注 2. “0” は設定禁止です。

(1)　マスタ受信

マスタデバイスとして、スレーブデバイスからデータを受信する機能です。

初めにスタートコンディション（ST）を生成し、次にスレーブデバイスのアドレスを送信します。このとき、8 ビット目は転送方向指定ビットになりますので、データ受信時には “1”（Read）を送信します。次にデータ受信を開始します。受信中は、1 バイト受信するごとに ACK を送信しますが、最終データ時のみ NACK を送信し、スレーブデバイスへ受信処理が完了したことを通知します。全データの受信が完了すると、ストップコンディション（SP）を生成してバスを解放します。



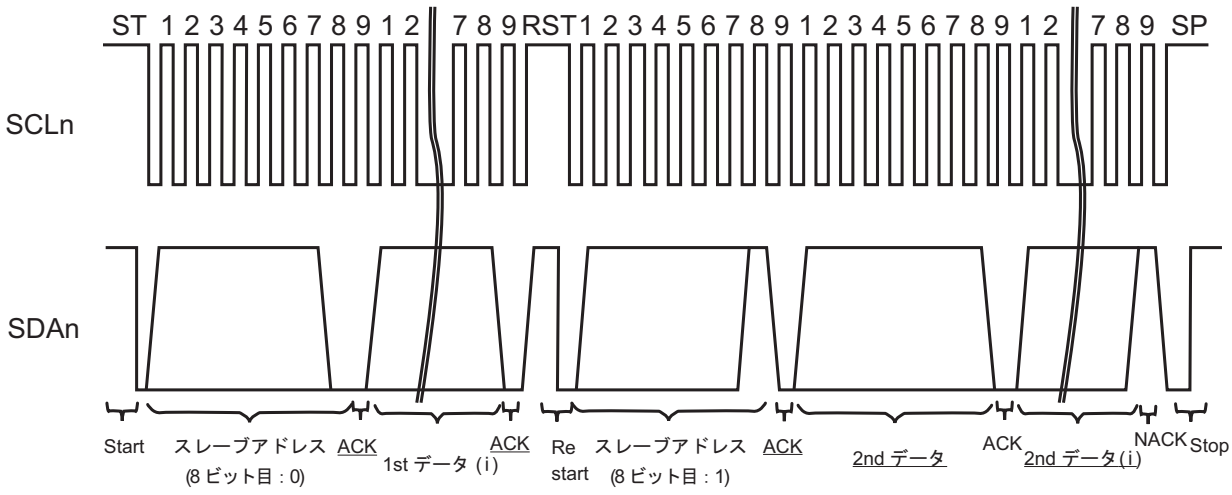
凡例：
n: チャネル番号
ST: スタートコンディション生成
SP: ストップコンディション生成
NACK:Acknowledge“1”
ACK:Acknowledge“0”
注 1.　下線のある信号は、スレーブからマスタへのデータ送信です

図 6.8　　マスタ受信 信号図

(2)　マスタ複合

マスタデバイスとして、スレーブデバイスへデータを送信します。送信完了後、リスタートコンディションを生成し、スレーブデバイスからデータを受信する機能です。

初めにスタートコンディション（ST）を生成し、次にスレーブデバイスのアドレスを送信します。このとき、8 ビット目の転送方向指定ビットには、“0”（Write）を送信します。次に 1st データを送信します。データの送信が完了すると、リスタートコンディション（RST）を生成し、スレーブアドレスを送信します。このとき、転送方向指定ビットには、“1”（Read）を送信します。次にデータ受信を開始します。受信中は、1 バイト受信するごとに ACK を送信しますが、最終データ時のみ NACK を送信し、スレーブデバイスへ受信処理が完了したことを通知します。全データの受信が完了すると、ストップコンディション（SP）を生成してバスを解放します。



凡例：
n: チャネル番号
ST: スタートコンディション生成
SP: ストップコンディション生成
RST: リスタートコンディション生成
NACK:Acknowledge“1”
ACK:Acknowledge“0”
注 1.　下線のある信号は、スレーブからマスタへのデータ送信です

図 6.9　　マスタ複合 信号図

6.9.4　R_RIIC_SlaveTransfer

R_RIIC_SlaveTransfer	
概　要	スレーブデバイスとして、送信および受信できる状態にする関数です。
ヘッダ	r_riic_rx_if.h
宣　言	riic_return_t R_RIIC_SlaveTransfer (riic_info_t *p_riic_info)
説　明	RIIC のスレーブ送信、またはスレーブ受信できる状態にします。マスタ通信中に本関数を呼び出した場合は、エラーとなります。引数で指定した RIIC のチャネルを設定します。チャネルの状態が “ アイドル状態 (RIIC_IDLE、RIIC_FINISH、RIIC_NACK)” の場合、次の処理を行います。 - 状態フラグの設定 - API で使用する変数の初期化 - RIIC 通信で使用する RIIC レジスタの初期化 - RIIC 割り込みの許可 - スレーブアドレスの設定、スレーブアドレス一致割り込みの許可 *p_riic_info RIIC 通信情報構造体のポインタ。引数の設定によって、スレーブ受信許可状態かスレーブ送信許可状態、またはその両方を選択できます。引数の設定可能範囲は表 6.12 を参照ください。また、受信パターンの波形イメージは図 6.10 を、送信パターンの波形イメージは図 6.11 を参照ください。 この構造体のうち、本関数で使用するメンバのみを以下に示します。この構造体の詳細については図 6.3 を参照してください。 下記のうち、API 実行中に値が更新される引数には、“ 更新あり ” と記載しています。 riic_ch_dev_status_t dev_sts; /* デバイス状態フラグ (更新あり) */ uint8_t ch_no; /* チャネル番号 */ riic_callback callbackfunc; /* コールバック関数 */ uint32_t cnt2nd; /* 2nd データカウンタ (バイト数)(スレーブ受信時のみ更新あり) */ uint32_t cnt1st; /* 1st データカウンタ (バイト数)(スレーブ送信時のみ更新あり) */ uint8_t * p_data2nd; /* 2nd データ格納バッファポインタ */ uint8_t * p_data1st; /* 1st データ格納バッファポインタ */
引　数	riic_info_t * p_riic_info　：RIIC 通信情報構造体のポインタ
リターン値	RIIC_SUCCESS　：問題なく処理が完了した場合 RIIC_ERR_INVALID_CHAN　：存在しないチャネルの場合 RIIC_ERR_INVALID_ARG　：不正な引数の場合 RIIC_ERR_NO_INIT　：初期設定ができていない場合（未初期化状態） RIIC_ERR_BUS_BUSY　：バスビジーの場合 RIIC_ERR_AL　：アービトレーションエラーが発生した場合 RIIC_ERR_OTHER　：現状態に該当しない不正なイベントが発生した場合
補足	なし

Example

```
/* for MasterReceive(combination mode) */
#include "r_riic_rx_if.h"

void CallbackMaster(void);
void CallbackSlave(void);
void main(void);

void main(void)
{
    volatile riic_return_t ret;
    riic_info_t iic_info_m;
    riic_info_t iic_info_s;
    uint8_t addr_eeprom[1]={0x50};
    uint8_t access_addr1[1]={0x00};
    uint8_t mst_send_data[5]={0x81,0x82,0x83,0x84,0x85};
    uint8_t slv_send_data[5]={0x71,0x72,0x73,0x74,0x75};
    uint8_t mst_store_area[5]={0xFF,0xFF,0xFF,0xFF,0xFF};
    uint8_t slv_store_area[5]={0xFF,0xFF,0xFF,0xFF,0xFF};

    /* Sets IIC Information for Master Send. */
    iic_info_m.dev_sts = 0;
    iic_info_m.ch_no = 0;
    iic_info_m.callbackfunc = &CallbackMaster;
    iic_info_m.cnt2nd = 3;
    iic_info_m.cnt1st = 1;
    iic_info_m.p_data2nd = mst_store_area;
    iic_info_m.p_data1st = access_addr1;
    iic_info_m.p_slv_adr = addr_eeprom;

    /* Sets IIC Information for Slave Transfer. */
    iic_info_s.dev_sts = 0;
    iic_info_s.ch_no = 0;
    iic_info_s.callbackfunc = &CallbackSlave;
    iic_info_s.cnt2nd = 3;
    iic_info_s.cnt1st = 3;
    iic_info_s.p_data2nd = slv_store_area;
    iic_info_s.p_data1st = slv_send_data;
    iic_info_s.p_slv_adr = (uint8_t*)FIT_NO_PTR;

    /* RIIC open */
    ret = R_RIIC_Open(&iic_info_m);

    /* RIIC slave transfer enable */
    ret = R_RIIC_SlaveTransfer(&iic_info_s);

    /* RIIC master send start */
    ret = R_RIIC_MasterSend(&iic_info_m);
    while(1);
}

void CallbackMaster(void)
{
    /* callback process (master)*/
}

void CallbackSlave(void)
{
    /* callback process (slave)
}
```

Special Notes:

スレーブ動作パターンごとの引数の設定可能範囲は、下表を参照してください。

表6.12　スレーブ動作パターンごとの引数の設定可能範囲

構造体メンバ	ユーザ設定可能範囲	
	スレーブ受信	スレーブ送信
*p_slv_adr	未使用（設定無効）	未使用（設定無効）
*p_data1st	（スレーブ送信用）	[送信用]1st データバッファポインタ注 1
*p_data2nd	[受信用]2nd データバッファポインタ注 2	（スレーブ受信用）
dev_sts	デバイス状態フラグ	デバイス状態フラグ
cnt1st	（スレーブ送信用）	0000 0001h ～ FFFF FFFFh
cnt2nd	0000 0001h ～ FFFF FFFFh	（スレーブ受信用）
callbackfunc	使用する関数名を指定してください。	使用する関数名を指定してください。
ch_no	00h ～ FFh	00h ～ FFh
rsv1,rsv2	予約領域（設定無効）	予約領域（設定無効）

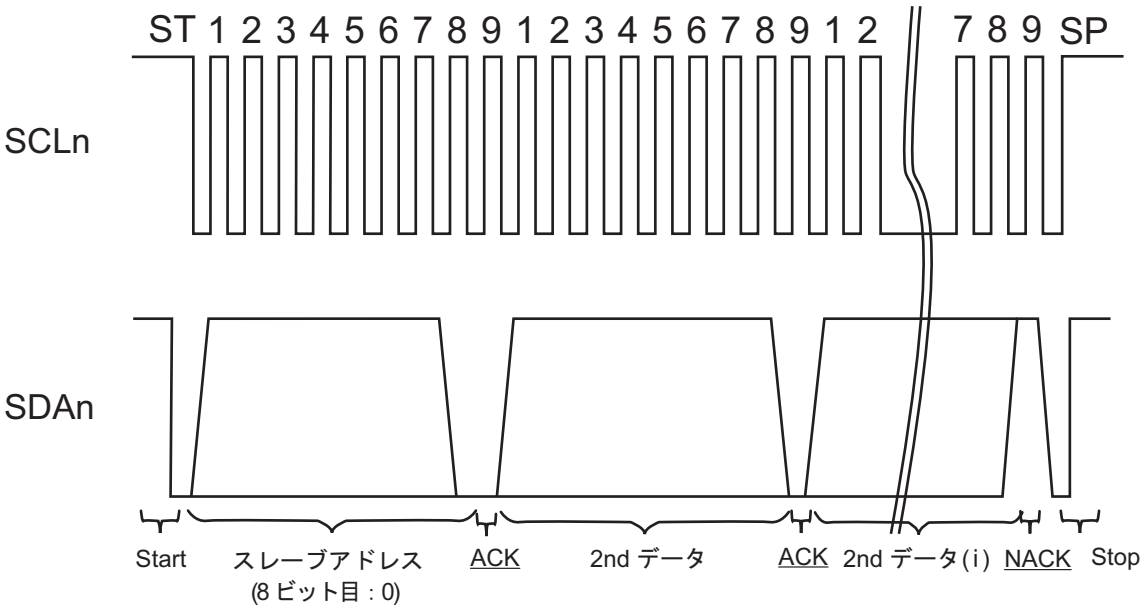
注1.　スレーブ送信を使用する場合、設定してください。
　　スレーブ送信を使用しない場合、“FIT_NO_PTR”を設定してください。
注2.　スレーブ受信をする場合、設定してください。
　　スレーブ受信を使用しない場合、“FIT_NO_PTR”を設定してください。

(1) スレーブ受信

スレーブデバイスとして、マスタデバイスからのデータを受信する機能です。

マスタデバイスが指定するスレーブアドレスが、「r_riic_config_if.h」で設定したスレーブデバイスのスレーブアドレスと一致したとき、スレーブ送受信を開始します。スレーブアドレスの 8 ビット目（転送方向指定ビット）によって、本モジュールが自動的にスレーブ受信かスレーブ送信かを判断して処理を行います。

マスタデバイスが生成したスタートコンディション（ST）を検出した後に、受信したスレーブアドレスが、自アドレスと一致し、かつスレーブアドレスの 8 ビット目（転送方向指定ビット）が “0”（Write）のとき、スレーブデバイスとして受信動作を開始します。最終データ（RIIC 通信情報構造体に設定された受信データ数）を受信時は、NACK を返すことでマスタデバイスに必要なデータをすべて受信したことを通知します。



凡例 :
n: チャネル番号
ST: スタートコンディション生成
SP: ストップコンディション生成
NACK:Acknowledge“1”
ACK:Acknowledge“0”
注 1. 下線のある信号は、スレーブからマスタへのデータ送信です

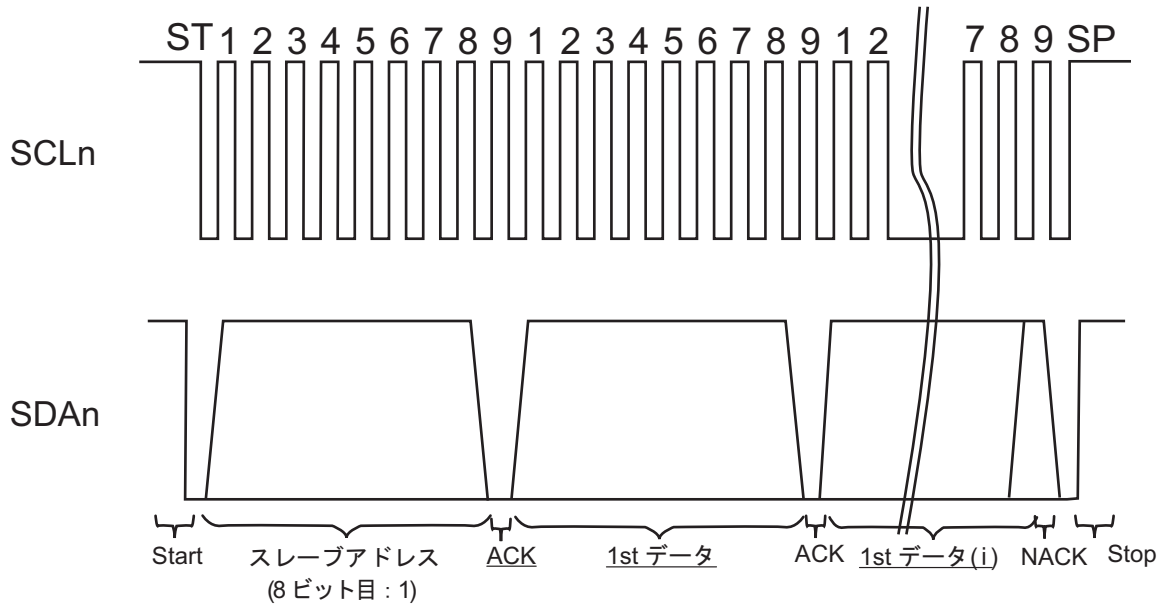
図 6.10 スレーブ受信 信号図

(2) スレーブ送信

スレーブデバイスとして、マスタデバイスへデータを送信する機能です。

マスタデバイスが指定するスレーブアドレスが、「r_riic_config_if.h」で設定したスレーブデバイスのスレーブアドレスと一致したとき、スレーブ送受信を開始します。スレーブアドレスの 8 ビット目（転送方向指定ビット）によって、本モジュールが自動的にスレーブ受信かスレーブ送信かを判断して処理を行います。

マスタデバイスからのスタートコンディション（ST）検出後に、受信したスレーブアドレスが、自アドレスと一致し、かつスレーブアドレスの 8 ビット目（転送方向指定ビット）が “1”（Read）のとき、スレーブデバイスとして送信動作を開始します。設定したデータ数（IC 通信情報構造体に設定した送信データ数）を超える送信データの要求があった場合、“0xFF” をデータとして送信します。ストップコンディション（SP）を検出するまでデータを送信します。



凡例 :
n: チャネル番号
ST: スタートコンディション生成
SP: ストップコンディション生成
NACK:Acknowledge“1”
ACK:Acknowledge“0”
注 1. 下線のある信号は、スレーブからマスタへのデータ送信です

図 6.11 スレーブ送信 信号図

6.9.5　R_RIIC_GetStatus

R_RIIC_GetStatus		
概　要	本モジュールの状態を確認する際に使用する関数です。	
ヘッダ	r_riic_rx_if.h	
宣　言	riic_sts_flg_t R_RIIC_GetStatus(riic_info_t * p_riic_info, riic_mcu_status_t * p_riic_status)	
説　明	本モジュールの状態を返します。 引数で指定した RIIC のチャネルの状態を、レジスタの読み出し、端子レベルの読み出し、変数の読み出しなどにより取得し、32 ビットの構造体で戻り値として返します。 本関数のコールで、RIIC のアービトレーションロストフラグ、および NACK フラグを “0” にクリアします。ステータスが “RIIC_ AL” の場合、“RIIC_FINISH” に更新します。 *p_riic_info RIIC 通信情報構造体のポインタ。 この構造体のうち、本関数で使用するメンバのみを以下に示します。この構造体の詳細については図 6.3 を参照してください。 下記のうち、API 実行中に値が更新される引数には、“ 更新あり ” と記載しています。 riic_ch_dev_status_t dev_sts; /* デバイス状態フラグ (ステータスが “RIIC_AL” 時、更新あり) */ uint8_t ch_no; /* チャネル番号 */ *p_riic_status RIIC のステータスを格納する変数のポインタ。	
引　数	riic_info_t * p_riic_info	: RIIC 通信情報構造体のポインタ
	riic_mcu_status_t * p_riic_status	: RIIC のステータスを格納する変数のポインタ
リターン値	RIIC_SUCCESS	: 問題なく処理が完了した場合
	RIIC_ERR_INVALID_CHAN	: 存在しないチャネルの場合
	RIIC_ERR_INVALID_ARG	: 不正な引数の場合
補足	なし	

Example

```
volatile riic_return_t ret;
riic_info_t iic_info_m;
riic_mcu_status_t riic_status;

iic_info_m.ch_no = 0;

ret = R_RIIC_GetStatus(&iic_info_m, &riic_status);
```

6.9.6　R_RIIC_Control

R_RIIC_Control	
概　要	主に通信エラー時に使用する関数です。 各コンディション出力、SDA 端子の Hi-Z 出力、SCL クロックのワンショット出力、および RIIC のモジュールリセットすることができます。
ヘッダ	r_riic_rx_if.h
宣　言	riic_return_t R_RIIC_Control(r_riic_info_t *p_riic_info, uint8_t ctrl_ptn)
説　明	RIIC の制御信号を出力します。引数で指定した各コンディション出力、SDA 端子 Hi-Z 出力、SCL クロックのワンショット出力、および RIIC のモジュールリセットを行います。 *p_riic_info I2C 通信情報構造体のポインタ。 この構造体のうち、本関数で使用するメンバのみを以下に示します。この構造体の詳細については図 6.3 を参照してください。 下記のうち、API 実行中に値が更新される引数には “ 更新あり ” と記載しています。 riic_ch_dev_status_t dev_sts; /* デバイス状態フラグ (出力パターンに "RIIC_GEN_RESET" 指定時、更新あり) */ uint8_t ch_no; /* チャネル番号 */ ctrl_ptn 出力パターンを設定します。 出力パターンに指定可能な値は表 6.5 を参照してください。 - 次の出力パターンは、同時指定が可能です。同時指定する場合は、“|” を用いてください。 • “RIIC_GEN_START_CON” と “RIIC_GEN_STOP_CON” と “RIIC_GEN_RESTART_CON” の、3 つ、または、いずれか 2 つの組み合わせで同時指定可能です。 • “RIIC_GEN_SDA_HI_Z” と “RIIC_GEN_SCL_ONESHOT” の 2 つを同時指定可能です。 RIIC_GEN_START_CON 0x01 /* スタートコンディションの生成 */ RIIC_GEN_STOP_CON 0x02 /* ストップコンディションの生成 */ RIIC_GEN_RESTART_CON 0x04 /* リスタートコンディションの生成 */ RIIC_GEN_SDA_HI_Z 0x08 /* SDA 端子を Hi-Z 出力 */ RIIC_GEN_SCL_ONESHOT 0x10 /* SCL クロックのワンショット出力 */ RIIC_GEN_RESET 0x20 /* RIIC のモジュールリセット */
引　数	riic_info_t * p_riic_info : RIIC 通信情報構造体のポインタ riic_mcu_status_t * p_riic_status : RIIC のステータスを格納する変数のポインタ

リターン値	RIIC_SUCCESS	: 問題なく処理が完了した場合
	RIIC_ERR_INVALID_CHAN	: 存在しないチャネルの場合
	RIIC_ERR_INVALID_ARG	: 不正な引数の場合
	RIIC_ERR_NO_INIT	: 初期設定ができていない場合（未初期化状態）
	RIIC_ERR_BUS_BUSY	: バスビジーの場合
	RIIC_ERR_AL	: アービトレーションエラーが発生した場合
	RIIC_ERR_OTHER	: 現状態に該当しない不正なイベントが発生した場合
補足	なし	

Example

```
/* Outputs an extra SCL clock cycle after changes the SDA pin in a high-impedance state*/
volatile riic_return_t ret;
riic_info_t iic_info_m;

iic_info_m.ch_no = 0;

ret = R_RIIC_Control(&iic_info_m, RIIC_GEN_SDA_HI_Z | RIIC_GEN_SCL_ONESHOT);
```

6.9.7 R_RIIC_Close

R_RIIC_Close	
概　要	RIIC の通信を終了し、使用していた RIIC を開放する際に使用する関数です。
ヘッダ	r_riic_rx_if.h
宣　言	riic_return_t R_RIIC_Close(riic_info_t *p_riic_info)
説　明	RIIC 通信を終了するための設定をします。引数で指定した RIIC のチャネルを無効にします。本関数では次の処理を行います。 - RIIC のモジュールストップ状態への遷移 - RIIC 出力ポートの開放（SCL0, SDA0 をポートモード（入力）に変更） - RIIC 割り込みの禁止 再度通信を開始するには、R_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。 *p_riic_info RIIC 通信情報構造体のポインタ。 この構造体のうち、本関数で使用するメンバのみを以下に示します。この構造体の詳細については図 6.3 を参照してください。 下記のうち、API 実行中に値が更新される引数には、“ 更新あり ” と記載しています。 riic_ch_dev_status_t dev_sts; /* デバイス状態フラグ (更新あり) */ uint8_t ch_no; /* チャネル番号 */
引　数	riic_info_t * p_riic_info　：RIIC 通信情報構造体のポインタ
リターン値	RIIC_SUCCESS　：問題なく処理が完了した場合 RIIC_ERR_INVALID_CHAN　：存在しないチャネルの場合 RIIC_ERR_INVALID_ARG　：不正な引数の場合
補足	なし

Example

```
volatile riic_return_t ret;
riic_info_t iic_info_m;

iic_info_m.ch_no = 0;

ret = R_RIIC_Close(&iic_info_m);
```

6.9.8 R_RIIC_GetVersion

R_RIIC_GetVersion	
概　要	API のバージョンを返す関数です。
ヘッダ	r_riic_rx_if.h
宣　言	uint32_t R_RIIC_GetVersion(void)
説　明	本 API のバージョン番号を返します。
引　数	なし　　：ー
リターン値	バージョン番号
補足	なし

Example

```
uint32_t version;

version = R_RIIC_GetVersion();
```

6.9.9 main

main	
概　要	サンプルプログラムのメイン関数です。
ヘッダ	―
宣　言	int main(void)
説　明	サンプルプログラムのメイン処理です。 処理内容は 6.10.1　メイン処理を参照してください。
引　数	なし
リターン値	0
補足	なし

6.10 フローチャート

6.10.1 メイン処理

図 6.12 にサンプルコードのメイン処理のフローチャートを示します。

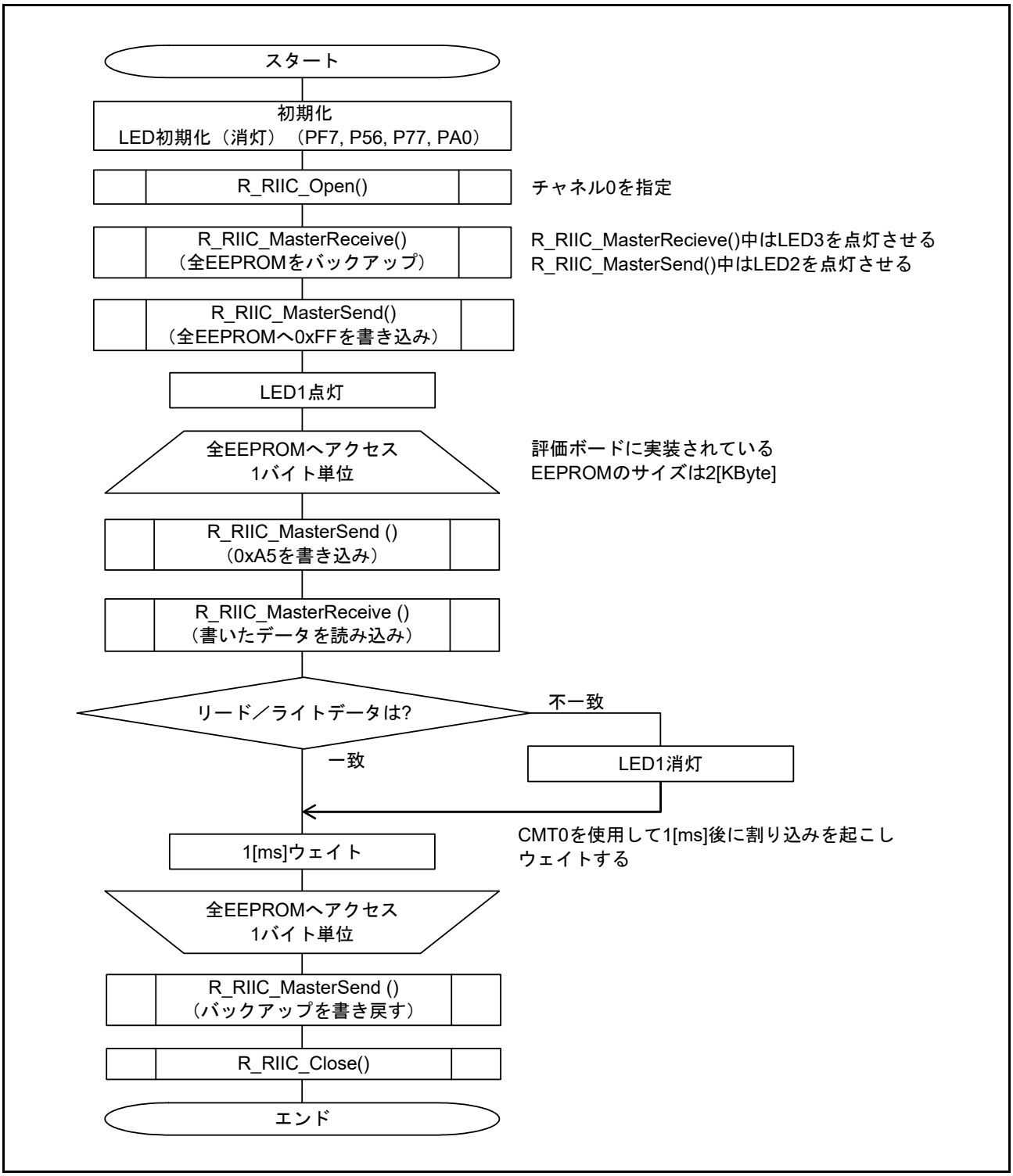


図 6.12 メイン処理

6.10.2 コールバック処理

図 6.13 にサンプルコードのコールバック処理のフローチャートを示します。

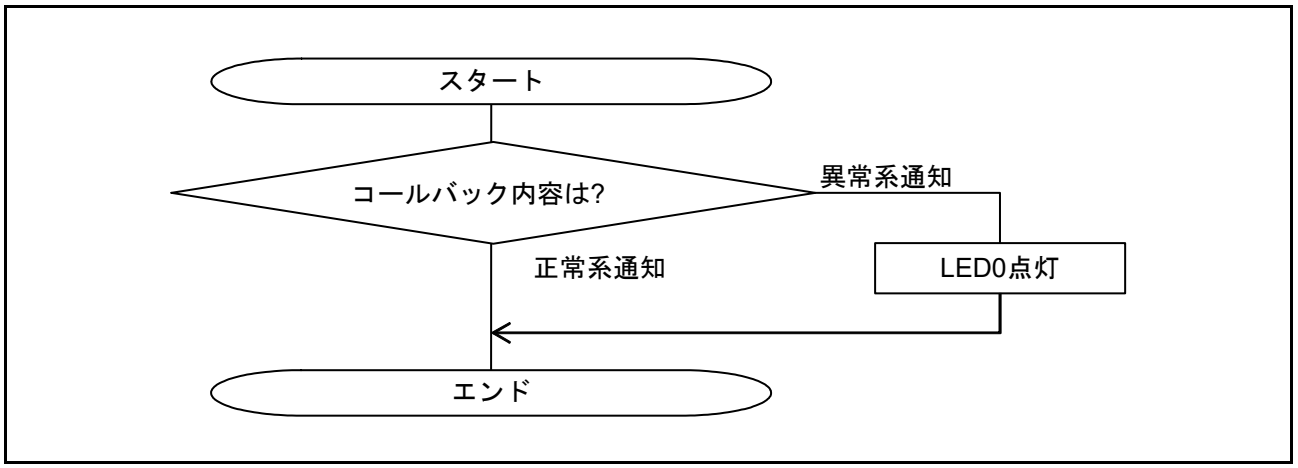


図 6.13 コールバック処理

6.10.3 コンペアマッチタイマ割り込み処理

図 6.14 にサンプルコードのコンペアマッチタイマ割り込み処理のフローチャートを示します。

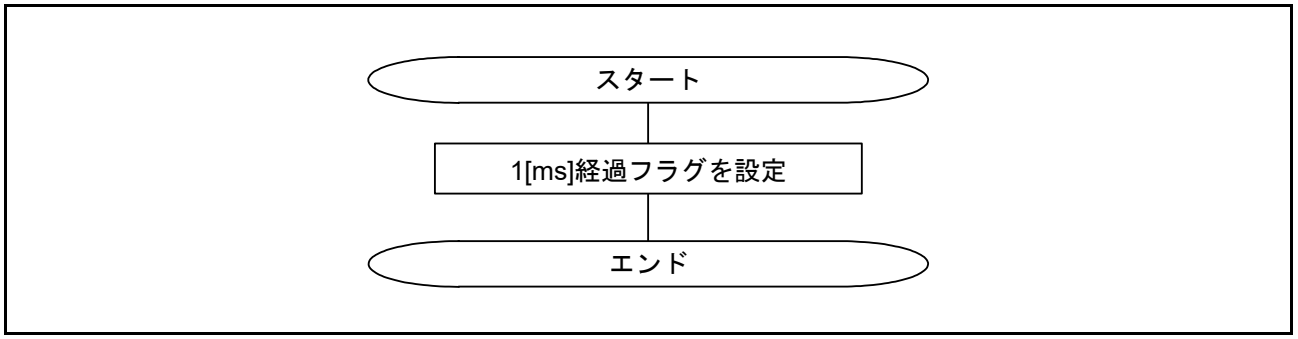


図 6.14 コンペアマッチタイマ割り込み処理

7. サンプルコード

サンプルコードは、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。

8. 参考ドキュメント

- ユーザーズマニュアル：ハードウェア
  RZ/T1 グループ ユーザーズマニュアル ハードウェア編
  （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。）

  RZ/T1 Evaluation Board RTK7910022C00000BR ユーザーズマニュアル
  （最新版をルネサス エレクトロニクスホームページから入手してください。）

- テクニカルアップデート／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
  IAR 統合開発環境（IAR Embedded Workbench® for Arm）に関しては、IAR ホームページから入手してください。
  （最新版を IAR 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	RIICサンプルプログラム アプリケーションノート

Rev.	発行日	改訂内容	
		ページ	ポイント
0.10	2015.03.06	—	初版発行
1.00	2015.04.10	—	Web掲載に際しRevのみ変更
1.10	2015.07.06	2. 動作環境	
		4	表2.1 動作環境　統合開発環境　表記一部修正、追加
		6. ソフトウェア説明	
		12	6.2.4 説明文 参照を追加
		12	表6.3 タイトル、サイズを一部修正。
		13	表6.4 追加
		13	表6.5 追加
1.20	2015.12.03	2. 動作環境	
		4	表2.1 動作環境　統合開発環境　一部修正
1.30	2017.07.18	1. 仕様	
		3	表1.1 使用する周辺機能と用途　I/Oポートを変更
		2. 動作環境	
		4	表2.1 動作環境　統合開発環境の内容変更
		5. ハードウェア説明	
		7	図5.1 ハードウェア構成例　I/Oポートと制御するLEDを変更
		7	表5.1 使用端子と機能　I/Oポートの端子名とその内容を変更
		6. ソフトウェア説明	
		8	表6.1 動作概要　動作結果表示のLEDを変更
		—	6.2.4 必要メモリサイズ　削除
		43	図6.12 メイン処理　I/OポートとLEDを変更
		44	図6.13 コールバック処理　LEDを変更
1.40	2018.06.07	2.動作環境	
		4	表2.1 動作環境　統合開発環境の内容変更
		8. 参考ドキュメント	
		46	IAR 統合開発環境名変更

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/