RL78/G14 グループ
I²C バス圧力センサ用制御ソフトウェア

R01AN3341JJ0101
Rev.1.01
2016.10.07

要旨

本アプリケーションノートでは、ルネサス エレクトロニクス製 MCU を使用した I²C バス圧力センサ用制御方法と制御ソフトウェアの使用方法を説明します。

本制御ソフトウェアは、圧力センサの各コマンドに対応した制御が可能です。

本制御ソフトウェアでは、スレーブデバイスを制御するためのソフトウェアを上位層、I²C シングルマスタ基本プロトコル制御を実現するソフトウェアを下位層とし扱います。上位層は下位層で用意されているプロトコルを組み合わせてスレーブデバイスを制御します。

本制御ソフトウェアは、上位層の圧力センサを制御するソフトウェアを提供します。

動作確認デバイス

圧力センサ

ミツミ電機社製 MMR931XA

動作確認に使用した MCU

RL78/G14 グループ　：RL78/G14（IICA を使用）

本アプリケーションノートを他の MCU や他メモリへ適用する場合、その MCU やメモリの仕様に合わせて変更し、十分評価してください。

また、本制御ソフトウェアはセンサデバイスへのアクセス制御方法のサンプルコードであり、動作を保証するものではありません。

目次

1. 概要

1.1 機能

ルネサス エレクトロニクス製 MCU を使用した I²C バス圧力センサ用制御ソフトウェアを提供します。

別途、MCU 個別の I²C シングルマスタ制御ソフトウェアが必要です。

表 1-1 に使用する周辺機器と用途を、図 1-1 に使用例を示します。

以下に、I²C バス制御の圧力センサ制御ソフトウェアの概略を示します。

- I²C マスタデバイスをルネサス エレクトロニクス製 MCU、I²C スレーブデバイスを圧力センサとするデバイスドライバです。
- 各制御を開始する開始関数と、通信を監視しプロトコル処理を進めるアドバンス関数により通信を実現します。通信開始後は、アドバンス関数をコールすることで通信を完了させる必要があります。
- 複数のチャネルをサポートします。
- 通信中に通信が一定時間止まった場合や、ノイズ等の影響によりバスハングアップが発生した場合、復帰処理を行い、バスを開放することができます。

さらに、圧力センサ個別の機能を以下に示します。

(1) ミツミ電機社製 MMR931XA 制御時

- リセットコマンド、アクティブコマンド、測定結果読み出しコマンド（リザルトコマンド）の 3 コマンドプロトコルをサポートします。
- 割り込みステータス取得関数をサポートします。

表 1-1　使用する周辺機器と用途

周辺機器	用途
MCU 内蔵の I²C バス制御機能	I²C バス通信機能 1 もしくは複数チャネル（必須）
割り込み	スレーブデバイスからの割り込み検出（オプション）

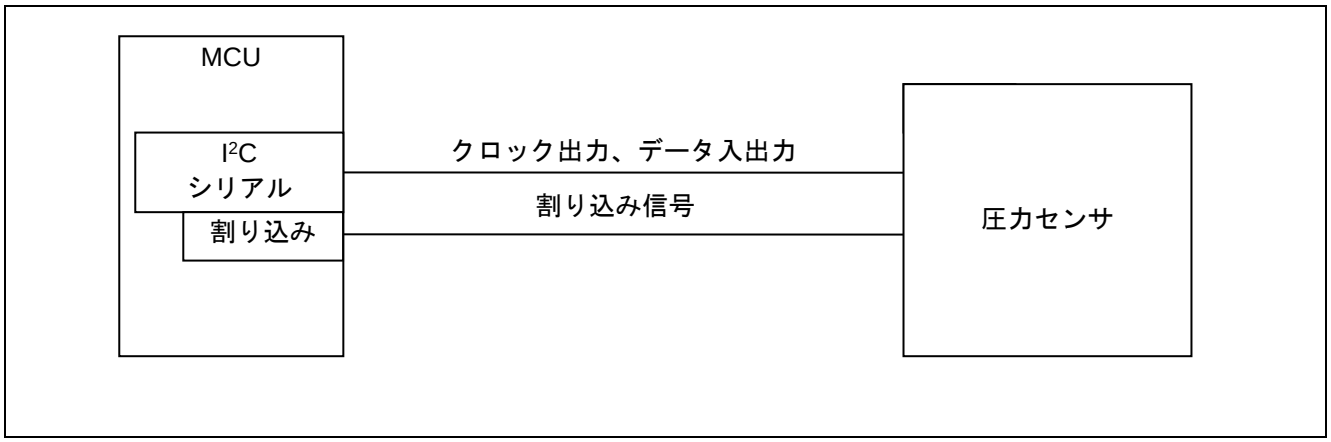


図 1-1　使用例

1.2　API の概要とメモリサイズ

1.2.1　API の概要

表 1-2 に圧力センサ制御ソフトウェアに含まれる API 関数を示します。

表 1-2　API 関数

関数名	説明
R_PRSSRSNSR_IIC_Open()	本制御ソフトウェアの初期化処理
R_PRSSRSNSR_IIC_Close()	本制御ソフトウェアの終了処理
R_PRSSRSNSR_IIC_Reset()	リセットコマンド送信開始処理
R_PRSSRSNSR_IIC_Active()	アクティブコマンド送信開始処理
R_PRSSRSNSR_IIC_Result()	リザルトコマンド送信開始処理
R_PRSSRSNSR_IIC_AckPolling()	Acknowledge polling 開始（ビジー終了待ち）処理
R_PRSSRSNSR_IIC_Advance()	アドバンス処理
R_PRSSRSNSR_IIC_GetIntStatus()	割り込みステータス取得処理
R_PRSSRSNSR_IIC_Recovery()	復帰処理
R_PRSSRSNSR_IIC_GetVersion()	本制御ソフトウェアのバージョン情報取得処理

1.2.2　動作環境とメモリサイズ

(1)　RL78/G14 の場合

表 1-3 に動作確認条件、表 1-4 に圧力センサ制御ソフトウェアで必要なメモリサイズを示します。

メモリサイズは「2.6 コンパイル時の設定」のデフォルト設定を選択した場合の値です。選択する定義により、メモリサイズは異なります。

表 1-3　動作確認条件

項目	内容
使用 MCU	RL78/G14 グループ（プログラム ROM 512KB／RAM 48KB）
動作周波数	メイン・システム・クロック：32MHz 周辺ハードウェア・クロック：32MHz I²C シリアル・クロック：400kHz
動作電圧	3.3V
統合開発環境	ルネサス エレクトロニクス製 CS+ for CC V4.00.00
Cコンパイラ	ルネサス エレクトロニクス製 RL78 ファミリ用 C コンパイラパッケージ V1.02.00
	コンパイルオプション： 統合開発環境のデフォルト設定を使用しています。
ソフトウェアのバージョン	Ver.1.00
使用ボード	サイレックス・テクノロジー社製 SX-ULPAN-DVK-S01

表 1-4　必要メモリサイズ

使用メモリ	サイズ	備考
ROM	1,221 バイト（圧力センサ用制御ソフトウェア） 4,082 バイト（I²C シングルマスタ制御ソフトウェア）	r_prssrsnsr_iic.c r_prssrsnsr_iic_type.c r_prssrsnsr_iic_type_sub.c r_prssrsnsr_iic_drvif_int.c r_prssrsnsr_iic_drvif_int_dev0.c r_prssrsnsr_iic_drvif_serial.c r_prssrsnsr_iic_drvif_serial_dev0.c
RAM	28 バイト（圧力センサ用制御ソフトウェア） 30 バイト（I²C シングルマスタ制御ソフトウェア）	r_iic_drv_api.c r_iic_drv_int.c r_iic_drv_os.c r_iic_drv_sfr.c r_iic_drv_sub.c 上記の動作確認条件による
最大使用ユーザスタック	100 バイト	
最大使用割り込みスタック	18 バイト	

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。
エンディアンにより、上記のメモリサイズは、異なります。
上記は、I²C シングルマスタ制御ソフトウェアに IICA を使用し、データ転送に Software を使用する場合のメモリサイズです。
上記の I²C シングルマスタ制御ソフトウェアの ROM/RAM サイズは使用ボードに合わせ込んだ場合のメモリサイズです。
最大使用ユーザスタックサイズは、下位層の I²C シングルマスタ制御ソフトウェアのスタックサイズも含みます。

1.3　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。

1.3.1　I²C シングルマスタ制御ソフトウェア

- RL78/G14、RL78/G1C、RL78/L12、RL78/L13、RL78/L1C グループ　IICA を使った I²C シングルマスタ制御ソフトウェア(R01AN1074JJ)

1.3.2　センサノード応用例

- RL78/G14 グループ　IoT SDK WLAN センサーノード版ソリューション(R01AN3363JJ)

上記の応用例は、以下の制御ソフトウェアと組み合わせて使用します。

- RL78/G14 グループ　サイレックス・テクノロジー社製無線 LAN モジュール SX-ULPAN-2402 用制御ソフトウェア(R01AN3150JJ)

1.4　ハードウェア接続

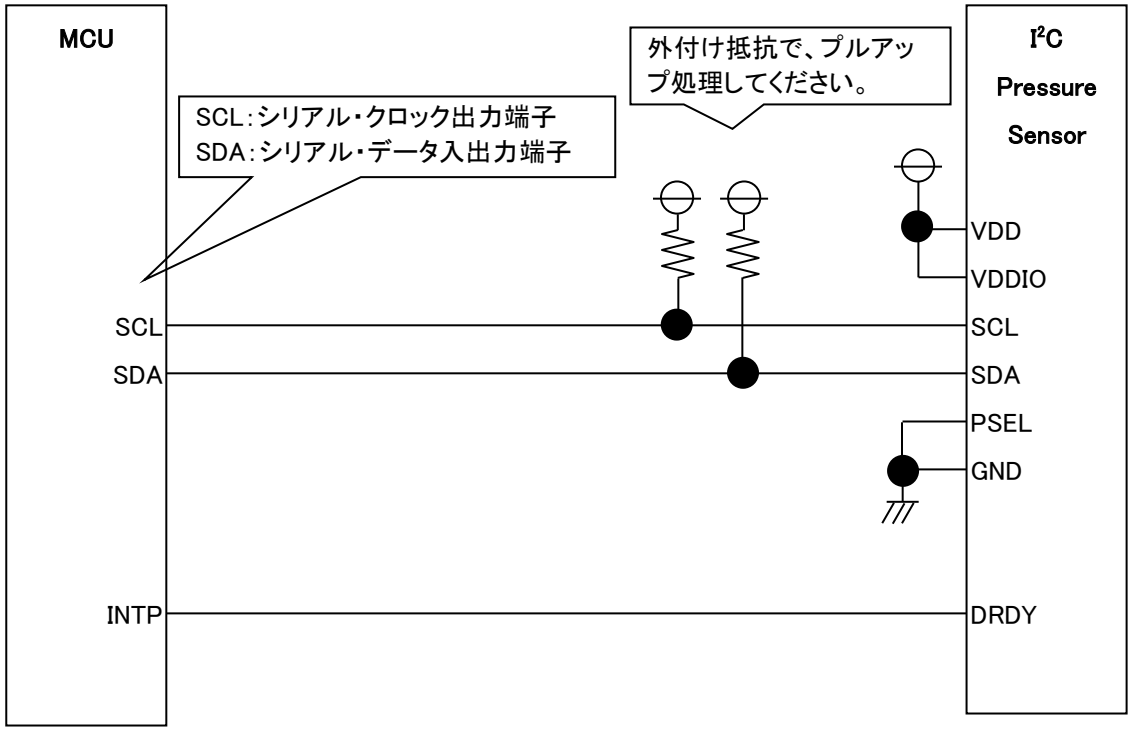
MCU とシリアル・インタフェースにより端子名が異なります

シリアル・クロック・ラインおよびシリアル・データ・バス・ラインは、出力が N-ch オープン・ドレーンのため、外部にプルアップ抵抗が必要です。システムに応じて適正な抵抗を検討してください。また、各信ラインの回路的マッチングを取るためのダンピング抵抗を検討してください。

(1)　ミツミ電機社製 MMR931XA 制御時

図 1-2 に接続例を示します。

バスを供給させ、複数デバイスの制御が可能です。ただし、デバイス毎に割り込み信号端子を設ける必要があります。



・MCU 上のシリアル I/O に使用される端子名は、MCU に依存します。
・INTP は割り込み入力端子を示します。
・本アプリケーションノートでは、制御ソフトウェアの記述に合わせて、SCL 端子、SDA 端子として表します。

図 1-2　MMR931XA と接続例

表 1-5　使用端子と機能

端子名	入出力	内容
SCL	出力	シリアル・クロック出力
SDA	入出力	シリアル・データ入出力
INTP	入力	割り込み入力

1.4.2　最大転送速度

最大 400kHz までの設定可能です。ただし、標準モード・デバイスとファーストモード・デバイスが混載されるバスでは、標準モードの最大 100kHz に設定する必要があります。

以下に、混載バス・システムでの最大転送速度を示します。

表 1-6　混載バス・システムでの最大転送速度

通信デバイス	混載デバイス	
	ファーストモード	標準モード
ファーストモード	0 – 400kHz	0 – 100kHz
標準モード	0 – 100kHz	0 – 100kHz

1.5　ソフトウェア制御

1.5.1　概要

MCU 内蔵の I²C バス制御機能を使用してシングルマスタ I²C バスのマスタとして、スレーブデバイスの圧力センサを制御するための方法を示します。

なお、使用可能なシリアル・クロック周波数は、MCU のユーザーズマニュアル ハードウェア編およびスレーブデバイスのデータシートで、確認してください。

(1)　ミツミ電機社製 MMR931XA 制御時

圧力センサ用リセットコマンド、アクティブコマンド、測定結果読み出しコマンド（リザルトコマンド）の 3 コマンド制御機能を提供します。これらの 3 コマンドを使って、圧力測定結果、温度測定結果、高度換算結果を取得できます。

制御ソフトウェアでは、I²C 制御用端子以外に、圧力センサからの割り込み（アクティブハイ）検出機能を提供します。

1.5.2　ソフトウェア構成

(1)　ミツミ電機社製 MMR931XA 制御時

図 1-3 にソフトウェア構成を示します。

本制御ソフトウェアを使用して、スレーブデバイスを制御するためのアプリケーションを作成してください。

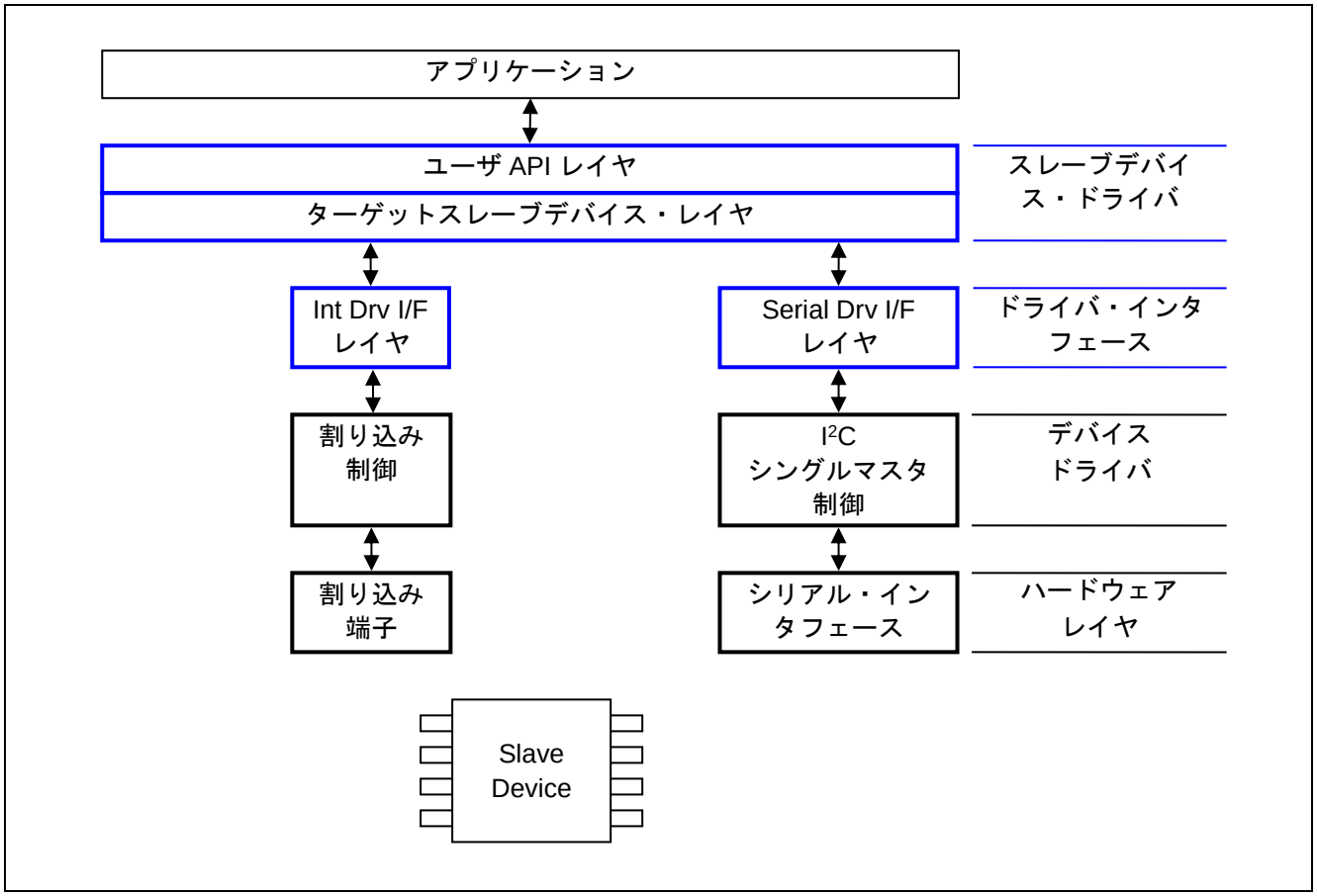


図 1-3　ソフトウェア構成

(a)　アプリケーション

圧力センサの制御例（デモプログラム）を同梱済です。参照してください。

(b)　ユーザ API レイヤ（r_prssrsnsr_iic.c）

圧力センサ制御ソフトウェアのユーザインタフェース部です。

(c)　ターゲットスレーブデバイス・レイヤ（r_prssrsnsr_iic_type.c）

圧力センサ制御ソフトウェアの圧力センサデバイス依存部分です。

(d)　Serial Drv I/F レイヤ（r_prssrsnsr_iic_drvif_serial.c）

上位層の圧力センサ制御ソフトウェアと下位層のデバイスドライバとの接続部分です。

I²C シングルマスタ制御ソフトウェア毎に合わせ込みが必要です。

(e)　Int Drv I/F（r_prssrsnsr_iic_drvif_int.c）

上位層の圧力センサ制御ソフトウェアと下位層のデバイスドライバ（割り込み制御ソフトウェア）との接続部分です。

割り込み制御ソフトウェア毎に合わせ込みが必要です。ただし、本制御ソフトウェアでは、下位層の割り込み制御ソフトウェアを使用せず、このレイヤで端子設定と割り込みを制御するレジスタを直接制御します。

(f)　I²C シングルマスタ制御（r_iic_drv_api.c）

I²C シングルマスタ制御ソフトウェアです。

(g)　割り込み制御

割り込み制御のデバイスドライバです。

ただし、本制御ソフトウェアでは、割り込み制御のデバイスドライバを使用せず、Int Drv I/F レイヤでト制御しています。

1.5.3　データバッファと送信／受信データの関係

本制御ソフトウェアは、送信／受信データポインタを引数として設定します。RAM 上のデータバッファのデータ並びと送信／受信順番の関係は、以下のとおりで、エンディアンや使用するシリアル通信機能に関係なく、送信データバッファの並びの順に送信し、また、受信の順に受信データバッファに書き込みます。

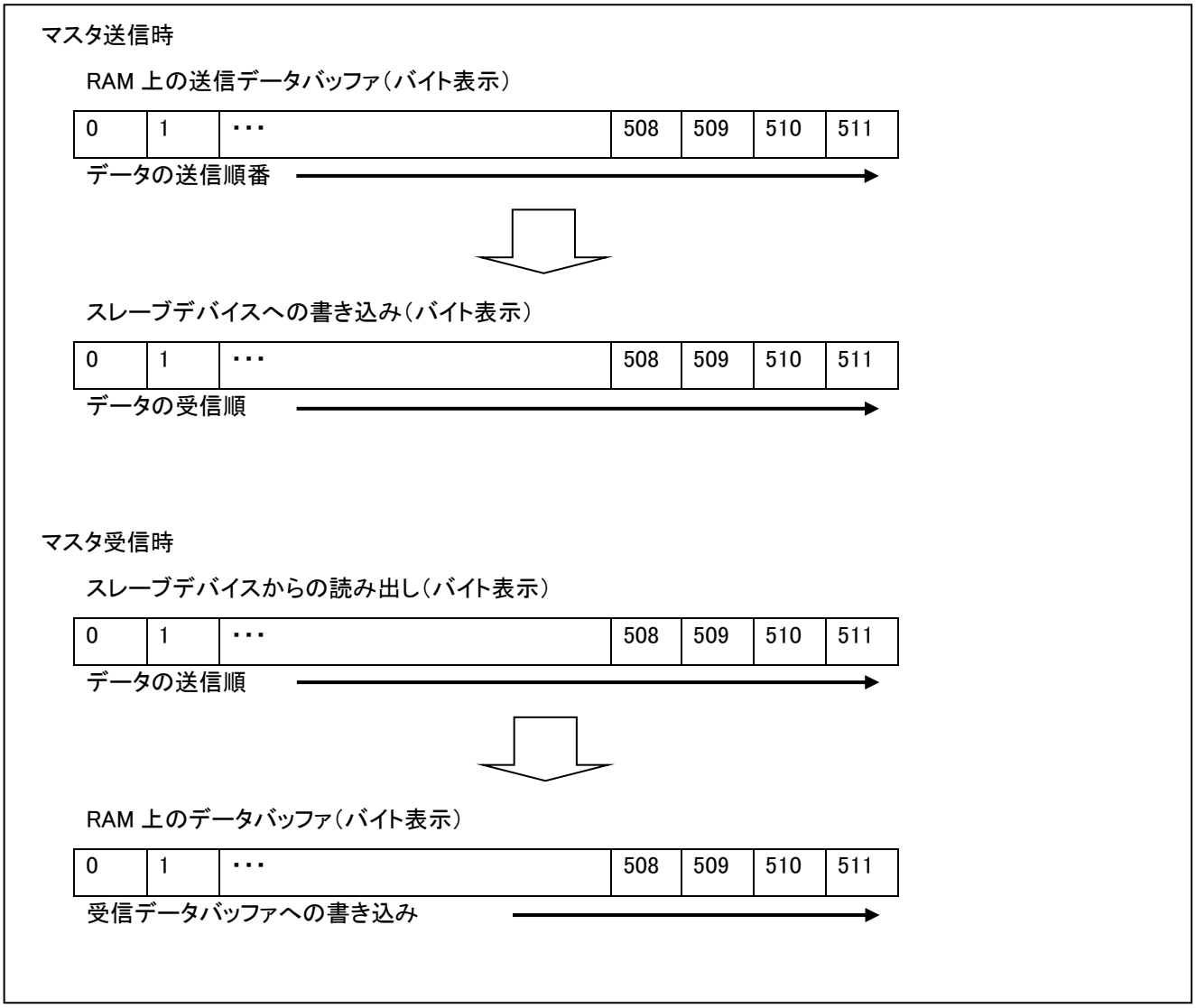


図 1-4　データバッファと送信／受信データの関係

1.6　DTC 使用の制限

DTC の使用に制限があります。制限事項を以下に示します。

(1)　RL78/G14 の場合

DTC：MCU が DTC 機能を持ちますが、本制御ソフトウェアは未サポートです。

1.7　ファイル構成

表 1.7 に制御ソフトウェアで使用するファイルを示します。

表 1.7 ファイル構成

¥an-r01an3341jj0101-rl78g14-iic	<DIR>	圧力センサ用制御ソフトウェアのフォルダ
r01an3341jj0101-rl78g14.pdf		アプリケーションノート
¥source	<DIR>	プログラム格納用フォルダ
¥r_bsp	<DIR>	BSP 用フォルダ
platform.h		ヘッダファイル
¥r_config	<DIR>	コンフィギュレーションファイル用フォルダ
r_prssrsnsr_iic_config.h		圧力センサ設定ヘッダファイル（参照用）
¥r_prssrsnsr_iic	<DIR>	圧力センサドライバ用フォルダ
r_prssrsnsr_iic_if.h		ユーザ I/F モジュール　ヘッダファイル
¥ref	<DIR>	コンフィギュレーションファイル用フォルダ（参照用）
r_prssrsnsr_iic_config_reference.h		圧力センサ設定ヘッダファイル（参照用）
¥src	<DIR>	プログラム用フォルダ
r_prssrsnsr_iic.c		ユーザ I/F モジュール
r_prssrsnsr_iic_private.h		ヘッダファイル
¥driver_interfaces	<DIR>	ドライバ I/F 用フォルダ
¥interrupt	<DIR>	割り込み用フォルダ
r_prssrsnsr_iic_drvif_int.c		ドライバ I/F モジュール
¥rl78_intp	<DIR>	RL78 用フォルダ
r_prssrsnsr_iic_drvif_int_dev0.c		内部モジュール（デバイス 0 用）
r_prssrsnsr_iic_drvif_int_dev1.c		内部モジュール（デバイス 1 用）
¥serial	<DIR>	シリアル用フォルダ
r_prssrsnsr_iic_drvif_serial.c		ドライバ I/F モジュール
¥rl78_iica	<DIR>	RL78 用フォルダ
r_prssrsnsr_iic_drvif_serial_dev0.c		内部モジュール（デバイス 0 用）
r_prssrsnsr_iic_drvif_serial_dev1.c		内部モジュール（デバイス 1 用）
¥prssrsnsr_types	<DIR>	ユーザ I/F モジュール
r_prssrsnsr_iic_type.c		内部モジュール
¥prssrsnsr_mmr931	<DIR>	MMR931XA 用フォルダ
r_prssrsnsr_iic_type_sub.c		内部モジュール
r_prssrsnsr_iic_type_sub.h		内部モジュール　ヘッダファイル
¥sample	<DIR>	動作確認プログラム格納用フォルダ
r_prssrsnsr_iic_mcu_demo_mmr931.c		動作確認用のサンプルソースファイル
r_prssrsnsr_iic_mcu_demo_mmr931.h		動作確認用のサンプルソースヘッダファイル
¥cg_src	<DIR>	コード生成用フォルダ【注 1】

注 1：動作確認用のサンプルコードで使用します。フォルダ内のファイルの詳細を省略します。

2. API 情報

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。

- 割り込み

2.2 ソフトウェアの要求

I²C シングルマスタ制御ソフトウェアを用意してください。

2.3 サポートされているツールチェイン

本制御ソフトウェアは、「1.2.2 動作環境とメモリサイズ」に示すツールチェインで動作確認を行っています。

2.4 ヘッダファイル

すべての API 呼び出しと使用されるインタフェース定義は r_prssrsnsr_iic_if.h に記載しています。

ビルド毎の構成オプションは、r_prssrsnsr_iic_config.h で選択します。以下の順番でインクルードしてください。

```
#include "r_prssrsnsr_iic_if.h"
#include "r_prssrsnsr_iic_config.h"
```

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

(1)　ミツミ電機社製 MMR931XA 制御時

(a)　制御ソフトウェアの設定

本制御ソフトウェアのコンフィグレーションオプションの設定は、r_prssrsnsr_iic_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_prssrsnsr_iic_config.h*	
#define PRSSRSNSR_IIC_CFG_DEV0_INCLUDED　(1) ※デフォルト値は“1（有効）”	デバイス番号 0 を必ず選択してください。（1：有効） デバイス 1 個の場合、デバイス番号 0 のみを有効にしてください。
#define PRSSRSNSR_IIC_CFG_DEV1_INCLUDED　(0) ※デフォルト値は“0（無効）”	デバイス番号 1 を使用するかを選択できます。（1：有効、0：無効）
#define PRSSRSNSR_IIC_CFG_DEVx_DRVIF_CH_NO　(0) ※デフォルトは“0”	デバイス番号 x で使用する下位層の I²C シングルマスタ制御ソフトウェアのチャネル番号を指定してください。
#define PRSSRSNSR_IIC_CFG_DEVx_MMR931　(1) ※デバイス番号 0 のデフォルト値は、“1” ※デバイス番号 1 のデフォルト値は、“0”	デバイス番号 x で圧力センサ MMR931XA を制御する場合、制御対象に設定してください。（1：制御対象、0：制御非対象）
#define PRSSRSNSR_IIC_CFG_DEVx_MODE PRSSRSNSR_IIC_MODE_RL78_IICA_INT ※デフォルトは、PRSSRSNSR_IIC_MODE_RL78_IICA_INT	デバイス番号 x で使用する下位層の I²C シングルマスタ制御ソフトウェアを設定してください。また、DMAC もしくは DTC と組み合わせて使用する場合は、合わせて定義してください。 デバイス分の設定が必要です。

注：DEVx の“x”はデバイス番号（x=0 or 1）を示します。

(b)　I²C シリアル端子の設定

I²C シングルマスタ制御ソフトウェアを参照し、端子を割り当ててください。

(c)　割り込み入力端子の設定

圧力センサの DRY 信号を割り込み入力端子で受けます。

割り込み入力端子は、ドライバ・インタフェース層の r_prssrsnsr_iic_drvif_int_devN.c（N はデバイス番号）で設定します。

本制御ソフトウェアは、

設定箇所を以下に示します。

以下は、INTP4 を割り当てる例です。

表 2-1　割り込み入力端子の設定方法

#define 定義【注 1】	設定方法
PRSSRSNSR_IIC_INT_EGP_SFR_DEV0	外部割り込み立ち上がりエッジ許可レジスタ（EGPx）
PRSSRSNSR_IIC_INT_EGN_SFR_DEV0	外部割り込み立ち下がりエッジ許可レジスタ（EGNx）
PRSSRSNSR_IIC_INT_PMK_SFR_DEV0	割り込みマスク・フラグ・レジスタの INTPx 用ビット
PRSSRSNSR_IIC_INT_PIF_SFR_DEV0	割り込み要求フラグ・レジスタの INTPx 用ビット
PRSSRSNSR_IIC_INT_PPR0_SFR_DEV0	優先順位指定フラグ・レジスタ（PPR0x）
PRSSRSNSR_IIC_INT_PPR1_SFR_DEV0	優先順位指定フラグ・レジスタ（PPR1x）
PRSSRSNSR_IIC_INT_EGP_UP_EDGE_DEV0	EGPx レジスタ設定用の#define 定義。

	本定義と EGPx レジスタで or 処理を行い、対象のビットを 1（立ち上がりエッジ）にします。 本定義の値は、使用する割り込み番号のビットを 1、その他を 0 としてください。
PRSSRSNSR_IIC_INT_EGN_UP_EDGE_DEV0	EGNx レジスタ設定用の#define 定義。 本定義と EGNx レジスタで and 処理を行い、対象のビットを 0（立ち下がりエッジ）にします。 本定義の値は、使用する割り込み番号のビットを 0、その他を 1 としてください。
PRSSRSNSR_IIC_INT_PPR0_LEVEL_DEV0、 PRSSRSNSR_IIC_INT_PPR1_LEVEL_DEV0	PPR0x レジスタ、PPR1x レジスタ設定用の#define 定義 下表に従い設定してください。

PPR0	PPR1	優先順位レベルの選択
0	0	レベル 0 を指定（高優先順位）
0	1	レベル 1 を指定
1	0	レベル 2 を指定
1	1	レベル 3 を指定（低優先順位）

注 1：デバイス番号 1 の場合は、DEV0 を DEV1 に置き換えてください。

変更箇所のソースコード：r_prssrsnsr_iic_drvif_int_devN.c　61 行目付近

```
/* ==== Configuration ==== */
#define PRSSRSNSR_IIC_INT_EGP_SFR_DEV0        (EGP0)
#define PRSSRSNSR_IIC_INT_EGN_SFR_DEV0        (EGN0)
#define PRSSRSNSR_IIC_INT_PMK_SFR_DEV0        (PMK4)
#define PRSSRSNSR_IIC_INT_PIF_SFR_DEV0        (PIF4)
#define PRSSRSNSR_IIC_INT_PPR0_SFR_DEV0       (PPR04)
#define PRSSRSNSR_IIC_INT_PPR1_SFR_DEV0       (PPR14)
#define PRSSRSNSR_IIC_INT_EGP_UP_EDGE_DEV0    (0x10) /* EGP4 = 1 Up trigger
interrupt */
#define PRSSRSNSR_IIC_INT_EGN_UP_EDGE_DEV0    (0xef) /* EGN4 = 0 Up trigger
interrupt */
#define PRSSRSNSR_IIC_INT_PPR0_LEVEL_DEV0     (1)    /* PPR0 = 1 Level 3 */
#define PRSSRSNSR_IIC_INT_PPR1_LEVEL_DEV0     (1)    /* PPR1 = 1 Level 3 */
/* ======================== */
```

2.7　引数

API 関数の引数を参照してください。

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_prssrsnsr_iic_if.h で記載されています。

```
typedef enum e_prssrsnsr_iic_err
{
  PRSSRSNSR_IIC_NO_INIT           = 0,  /* Uninitialized state               */
  PRSSRSNSR_IIC_IDLE              = 1,  /* Idle state                        */
  PRSSRSNSR_IIC_FINISH            = 2,  /* Idle state (Finished communication)   */
  PRSSRSNSR_IIC_NACK              = 3,  /* Idle state (Occurred nack)        */
  PRSSRSNSR_IIC_COMMUNICATION     = 4,  /* Communication state               */
  PRSSRSNSR_IIC_LOCK_FUNC         = 5,  /* Using other API function          */
  PRSSRSNSR_IIC_BUS_BUSY          = 6,  /* Bus busy                          */
  PRSSRSNSR_IIC_SUCCESS           = 99,
  PRSSRSNSR_IIC_ERR_PARAM         = -1, /* Error parameter                   */
  PRSSRSNSR_IIC_ERR_AL            = -2, /* Error arbitration-lost            */
  PRSSRSNSR_IIC_ERR_NON_REPLY     = -3, /* Error non reply                   */
  PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD  = -4, /* Error holding SDA low             */
  PRSSRSNSR_IIC_ERR_OTHER         = -5, /* Error other                       */
} prssrsnsr_iic_err_t;
```

3. API 関数

3.1 R_PRSSRSNSR_IIC_Open

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_Open(
    uint8_t devno
)
```

Parameters

devno

デバイス番号（0, 1）

Return Values

PRSSRSNSR_IIC_IDLE	*/* 初期化完了した場合 */*
	→開始関数をコールすることで通信が可能です。
PRSSRSNSR_IIC_LOCK_FUNC	*/* 他の API 処理が実行中の場合 */*
	→他の API 処理終了後、本関数をコールしてください。
PRSSRSNSR_IIC_BUSY	*/* 通信中の場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_ERR_PARAM	*/* パラメータエラーの場合 */*
	→設定値を確認してください。
PRSSRSNSR_IIC_ERR_AL	*/* アービトレーションロスト発生の場合 */*
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_NON_REPLY	*/* 未応答エラー発生の場合 */*
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD	*/* SDA が Low ホールド状態の場合 */*
	→スレーブデバイスが Low ホールドしていないか、
	マスタデバイスから Low 信号が出力されていないか等、
	システムの状態を確認してください。
PRSSRSNSR_IIC_ERR_OTHER	*/* その他エラー発生の場合 */*
	→設定が正しく行われているか確認してください。

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

引数 dev で指定したデバイスに対応する割り込み設定を初期化し、引き続き対象チャネルの I²C ドライバを初期化します。実施後、アイドル状態に遷移し、通信可能状態になります。

Special Notes:

なし

3.2　R_PRSSRSNSR_IIC_Close

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_Close(
    uint8_t devno
)
```

Parameters

devno

デバイス番号（0, 1）

Return Values

PRSSRSNSR_IIC_NO_INIT	*/* 正常終了した場合*
	*（I²C ドライバリセットを行い、未初期化状態に遷移） */*
	→通信を再開するには、初期化関数をコールしてください。
PRSSRSNSR_IIC_LOCK_FUNC	*/* 他の API 処理が実行中の場合 */*
	→他の API 処理終了後、本関数をコールしてください。
PRSSRSNSR_IIC_ERR_PARAM	*/* パラメータエラー の場合 */*
	→設定値を確認してください。

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

引数 dev で指定したデバイスに対応する割り込み設定を無効化し、引き続き対象チャネルの I²C ドライバをリセットします。

Special Notes:

なし

3.3　R_PRSSRSNSR_IIC_Reset

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_Reset(
    uint8_t devno
)
```

Parameters

devno

デバイス番号（0, 1）

Return Values

＜本関数コール時＞

PRSSRSNSR_IIC_COMMUNICATION	*/* リセットコマンド送信を開始した場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_NO_INIT	*/* 未初期化状態の場合 */*
	→初期化関数をコールして初期化を終了させてください。
PRSSRSNSR_IIC_LOCK_FUNC	*/* 他の API 処理が実行中の場合 */*
	→他の API 処理終了後、本関数をコールしてください。
PRSSRSNSR_IIC_BUS_BUSY	*/* 通信中の場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_ERR_PARAM	*/* パラメータエラーの場合 */*
	→設定値を確認してください。
PRSSRSNSR_IIC_ERR_AL	*/* アービトレーションロストが発生中の場合 */*
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_NON_REPLY	*/* バス未解放もしくは ST（注 1）未検出の場合 */*
	注 1：スタートコンディションを示す。
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD	*/* SDA Low ホールド状態の場合 */*
	→スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。
PRSSRSNSR_IIC_ERR_OTHER	*/* その他エラーの場合 */*
	→ハードウェアエラーの可能性があります。復帰処理関数をコールしてください。

＜本関数コール後のアドバンス関数コール後＞

PRSSRSNSR_IIC_COMMUNICATION	*/* 通信中の場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_FINISH	*/* 全通信が正常終了した場合 */*
	→開始関数をコールすることで通信が可能です。

上記以外の Return Values は、アドバンス関数を参照ください。

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

引数 dev で指定したデバイスにリセットコマンドの送信を開始します。

Special Notes:

I²C シングルマスタ制御ソフトウェアのマスタ送信モード（パターン 1）を使用します。
本関数から戻った時点では、I²C 通信は完了していません。I²C 通信を終了させるためには、アドバンス関数をコールする必要があります。

3.4　R_PRSSRSNSR_IIC_Active

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_Active(
    uint8_t devno,
    uint8_t w_mode
)
```

Parameters

devno

デバイス番号（0, 1）

w_mode

書き込みモード

以下から 1 つ選択してください。

＜ミツミ電機社製 MMR931XA の場合＞

```
PRSSRSNSR_IIC_W_MODE_ACTIVE_1      /* Measure at MODE 1 */
PRSSRSNSR_IIC_W_MODE_ACTIVE_2      /* Measure at MODE 2 */
PRSSRSNSR_IIC_W_MODE_ACTIVE_3      /* Measure at MODE 3 */
PRSSRSNSR_IIC_W_MODE_ACTIVE_4      /* Measure at MODE 4 */
```

Return Values

＜本関数コール時＞

PRSSRSNSR_IIC_COMMUNICATION　/ アクティブコマンド送信を開始した場合 */*
→アドバンス関数をコールして通信を終了させてください。

PRSSRSNSR_IIC_NO_INIT　/ 未初期化状態の場合 */*
→初期化関数をコールして初期化を終了させてください。

PRSSRSNSR_IIC_LOCK_FUNC　/ 他の API 処理が実行中の場合 */*
→他の API 処理終了後、本関数をコールしてください。

PRSSRSNSR_IIC_BUS_BUSY　/ 通信中の場合 */*
→アドバンス関数をコールして通信を終了させてください。

PRSSRSNSR_IIC_ERR_PARAM　/ パラメータエラーの場合 */*
→設定値を確認してください。

PRSSRSNSR_IIC_ERR_AL　/ アービトレーションロストが発生中の場合 */*
→復帰処理関数をコールしてください。

PRSSRSNSR_IIC_ERR_NON_REPLY　/ バス未解放もしくは ST（注 1）未検出の場合 */*
注 1：スタートコンディションを示す。
→復帰処理関数をコールしてください。

PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD　/ SDA Low ホールド状態の場合 */*
→スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。

PRSSRSNSR_IIC_ERR_OTHER　/ その他エラーの場合 */*
→ハードウェアエラーの可能性があります。復帰処理関数をコールしてください。

＜本関数コール後のアドバンス関数コール後＞

PRSSRSNSR_IIC_COMMUNICATION　/ 通信中の場合 */*
→アドバンス関数をコールして通信を終了させてください。

PRSSRSNSR_IIC_FINISH　/ 全通信が正常終了した場合 */*
→開始関数をコールすることで通信が可能です。

上記以外の Return Values は、アドバンス関数を参照ください。

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

引数 dev で指定したデバイスにアクティブコマンドの送信を開始します。

Special Notes:

I²C シングルマスタ制御ソフトウェアのマスタ送信モード（パターン 1）を使用します。
本関数から戻った時点では、I²C 通信は完了していません。I²C 通信を終了させるためには、アドバンス関数をコールする必要があります。

3.5　R_PRSSRSNSR_IIC_Result

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_Result(
    uint8_t devno,
    uint8_t r_mode,
    uint8_t * p_result
)
```

Parameters

devno

デバイス番号（0, 1）

r_mode

読み出しモード

以下から 1 つ選択してください。

＜ミツミ電機社製 MMR931XA の場合＞

```
PRSSRSNSR_IIC_R_MODE_RESULT_PRESSURE      /* Read Pressure Result */
PRSSRSNSR_IIC_R_MODE_RESULT_TEMP          /* Read Temperature Result */
PRSSRSNSR_IIC_R_MODE_RESULT_ALT_CONV      /* Read Altitude Conversion Result */
```

** p_result*

リザルトデータバッファポインタ（3 バイト）

Return Values

＜本関数コール時＞

```
PRSSRSNSR_IIC_COMMUNICATION       /* リザルトコマンド送信を開始した場合 */
                                  →アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_NO_INIT             /* 未初期化状態の場合 */
                                  →初期化関数をコールして初期化を終了させてください。
PRSSRSNSR_IIC_LOCK_FUNC           /* 他の API 処理が実行中の場合 */
                                  →他の API 処理終了後、本関数をコールしてください。
PRSSRSNSR_IIC_BUS_BUSY            /* 通信中の場合 */
                                  →アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_ERR_PARAM           /* パラメータエラーの場合 */
                                  →設定値を確認してください。
PRSSRSNSR_IIC_ERR_AL              /* アービトレーションロストが発生中の場合 */
                                  →復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_NON_REPLY       /* バス未解放もしくは ST（注 1）未検出の場合 */
                                     注 1：スタートコンディションを示す。
                                  →復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD    /* SDA Low ホールド状態の場合 */
                                  →スレーブデバイスが Low ホールドしていないか、マスタ
                                  デバイスから Low 信号が出力されていないか等、システムの
                                  状態を確認してください。
PRSSRSNSR_IIC_ERR_OTHER           /* その他エラーの場合 */
                                  →ハードウェアエラーの可能性があります。復帰処理関数を
                                  コールしてください。
```

＜本関数コール後のアドバンス関数コール後＞

```
PRSSRSNSR_IIC_COMMUNICATION       /* 通信中の場合 */
                                  →アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_FINISH              /* 全通信が正常終了した場合 */
                                  →開始関数をコールすることで通信が可能です。
```

上記以外の Return Values は、アドバンス関数を参照ください。

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

Description
引数 dev で指定したデバイスに測定結果読み出しコマンド（リザルトコマンド）の送信を開始します。
本関数から戻った時点では、I²C 通信は完了していません。I²C 通信を終了させるためには、アドバンス関数をコールする必要があります。

Special Notes:
I²C シングルマスタ制御ソフトウェアのマスタ複合モードを使用します。

3.6　R_PRSSRSNSR_IIC_AckPolling

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_AckPolling(
    uint8_t devno
)
```

Parameters

devno

デバイス番号（0, 1）

Return Values

＜本関数コール時＞

PRSSRSNSR_IIC_COMMUNICATION	*/* Acknowledge polling（ビジー終了待ち）を開始した場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_NO_INIT	*/* 未初期化状態の場合 */*
	→初期化関数をコールして初期化を終了させてください。
PRSSRSNSR_IIC_LOCK_FUNC	*/* 他の API 処理が実行中の場合 */*
	→他の API 処理終了後、本関数をコールしてください。
PRSSRSNSR_IIC_BUS_BUSY	*/* 通信中の場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_ERR_PARAM	*/* パラメータエラーの場合 */*
	→設定値を確認してください。
PRSSRSNSR_IIC_ERR_AL	*/* アービトレーションロストが発生中の場合 */*
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_NON_REPLY	*/* 未応答エラーが発生中の場合 */*
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD	*/* SDA Low ホールド状態の場合 */*
	→スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。
PRSSRSNSR_IIC_ERR_OTHER	*/* その他エラーの場合 */*
	→ハードウェアエラーの可能性があります。復帰処理関数をコールしてください。

＜本関数コール後のアドバンス関数コール後＞

PRSSRSNSR_IIC_COMMUNICATION	*/* 通信中の場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_FINISH	*/* 全通信が正常終了した場合 */*
	→開始関数をコールすることで通信が可能です。

上記以外の Return Values は、アドバンス関数を参照ください。

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

引数 dev で指定したデバイスの Acknowledge Polling を利用したビジー終了待ちを開始します。
本関数から戻った時点では、I²C 通信は完了していません。I²C 通信を終了させるためには、アドバンス関数をコールする必要があります。

Special Notes:

I²C シングルマスタ制御ソフトウェアのマスタ送信モード（パターン 3）を使用します。

3.7　R_PRSSRSNSR_IIC_Advance

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_Advance(
    uint8_t devno
)
```

Parameters

devno

デバイス番号（0, 1）

Return Values

PRSSRSNSR_IIC_COMMUNICATION	*/* 通信中の場合 */*
	→アドバンス関数をコールして通信を終了させてください。
PRSSRSNSR_IIC_FINISH	*/* 全通信が正常終了した場合 */*
	→開始関数をコールすることで通信が可能です。
PRSSRSNSR_IIC_NACK	*/* NACK を検出した場合 */*
	→開始関数をコールすることで通信が可能です。
PRSSRSNSR_IIC_NO_INIT	*/* 未初期化状態の場合 */*
	→初期化関数をコールして初期化を終了させてください。
PRSSRSNSR_IIC_IDLE	*/* アイドル状態の場合 */*
	→開始関数をコールすることで通信が可能です。
PRSSRSNSR_IIC_LOCK_FUNC	*/* 他の API 処理が実行中の場合 */*
	→他の API 処理終了後、本関数をコールしてください。
PRSSRSNSR_IIC_BUS_BUSY	*/* 通信中の場合 */*
	→他のデバイスの通信を終了させてください。
PRSSRSNSR_IIC_ERR_PARAM	*/* パラメータエラーの場合 */*
	→設定値を確認してください。
PRSSRSNSR_IIC_ERR_AL	*/* アービトレーションロストが発生中の場合 */*
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_NON_REPLY	*/* アドバンス関数のコール回数が上限値を超えた、もしくは SP（注 1）生成処理を行ったが SP を検出できなかった場合 */*
	注 1：ストップコンディションを示す。
	→復帰処理関数をコールしてください。
PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD	*/* SDA Low ホールド状態の場合 */*
	→スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。
PRSSRSNSR_IIC_ERR_OTHER	*/* その他エラーの場合 */*
	→ハードウェアエラーの可能性があります。復帰処理関数をコールしてください。

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

引数 dev で指定したデバイスの通信を監視し通信を進める処理を実行します。リターン値に通信の状態を返します。

Special Notes:

各開始関数コール後は、アドバンス関数をコールして通信を完了させてください。
コールした開始関数により、通信完了時のアドバンス関数のリターン値が異なります。本項の説明、もしくは各開始関数で説明しているアドバンス関数のリターン値をご確認ください。
通信途中でエラーが発生した場合には、復帰処理後、再度データを設定し直してから開始関数をコールしてください。

3.8 R_PRSSRSNSR_IIC_GetIntStatus

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_GetIntStatus(
    uint8_t devno,
    uint8_t * p_status
)
```

Parameters

devno

デバイス番号（0, 1）

* *p_status*

割り込みステータスバッファポインタ（1 バイト）

Return Values

PRSSRSNSR_IIC_SUCCESS
PRSSRSNSR_IIC_ERR_PARAM

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

引数 dev で指定したデバイスの RDRY 端子の状態を p_status にセットします。
RDRY 端子の H 変化を検知した場合、p_status を PRSSRSNSR_IIC_INT_OCCUR にセットします。
RDRY 端子の H 変化を検知しなかった場合、p_status を PRSSRSNSR_IIC_INT_NOT_OCCUR にセットします。

Special Notes:

なし

3.9　R_PRSSRSNSR_IIC_Recovry

Format

```
prssrsnsr_iic_err_t R_PRSSRSNSR_IIC_Recovery(
    uint8_t devno
)
```

Parameters

devno

デバイス番号（0, 1）

Return Values

PRSSRSNSR_IIC_IDLE	*/* アイドル状態に遷移した場合 */* *→開始関数をコールすることで通信が可能です。*
PRSSRSNSR_IIC_LOCK_FUNC	*/* 他の API 処理が実行中の場合 */* *→他の API 処理終了後、本関数をコールしてください。*
PRSSRSNSR_IIC_ERR_PARAM	*/* パラメータエラーの場合 */* *→設定値を確認してください。*
PRSSRSNSR_IIC_ERR_AL	*/* アービトレーションロストが発生中の場合 */* *→復帰処理関数をコールしてください。*
PRSSRSNSR_IIC_ERR_NON_REPLY	*/* 未応答エラー発生中の場合 */* *→再度復帰処理関数をコールしてください。*
PRSSRSNSR_IIC_ERR_SDA_LOW_HOLD	*/* SDA Low ホールド状態の場合 */* *→スレーブデバイスが Low ホールドしていないか、マスタデバイスから Low 信号が出力されていないか等、システムの状態を確認してください。*
PRSSRSNSR_IIC_ERR_OTHER	*/* その他エラーの場合 */* *→ハードウェアエラーの可能性があります。復帰処理関数をコールしてください。* *もしくは、マスタデバイスから Low 信号が出力されていないか、SCL Low ホールド状態でないか等システムの状態を確認してください。*

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

通信エラーとなった場合に、復帰処理を行うことができます。
強制的に初期化したい場合、本関数をコールしてください。
本関数コール後、アイドル状態になります。

Special Notes:

I²C 内部リセットを行います。
リセット後、SDA が Low ホールドしている場合、SCL に疑似クロック（最大 9 クロック）を生成します。
I²C シングルマスタ制御ソフトウェアのマスタ送信モード（パターン 4）を使用し、スタートコンディションとストップコンディションを生成して、バスを解放します。
本処理実行後も通信に復帰できない場合、SDA が GND に固定されている等の異常が考えられます。

3.10　R_PRSSRSNSR_IIC_GetVersion

Format

uint32_t R_PRSSRSNSR_IIC_GetVersion(
　　viod
)

Parameters

なし

Return Values

バージョン番号　　*上位 2 バイト：メジャーバージョン、下位 2 バイト：マイナーバージョン*

Properties

r_prssrsnsr_iic_if.h にプロトタイプ宣言されています。

Description

バージョン情報を返します。

Special Notes:

なし

4. 注意事項

4.1 RAM 初期化に関する注意事項

RL78 ファミリマイコンは、RAM パリティ・エラー検出機能（初期値：パリティ・エラー・リセット発生を許可）があります。

RAM パリティ・エラー・リセット発生を許可する場合、使用する RAM 領域をデータ読み出し前に初期化する必要があります。関数コール時のスタック割り付けを想定し、BSS 領域とスタック領域の RAM を初期化してください。

詳細は、使用するマイコンのユーザーズマニュアル ハードウェア編の「RAM パリティ・エラー検出機能」を参照してください。

4.2 圧力センサ制御ソフトウェア共通の注意事項

本制御ソフトウェアは、圧力センサへのアクセス API を提供するものであり、圧力センサの全ての機能を確認したものではありません。また、取得したセンサデータ値の妥当性を確認したものではありません。

圧力センサのデータシートを参照し、各圧力センサの機能を利用してください。

4.3　ミツミ電機社製 MMR931XA 制御例

参照したデータシートは Rev.3 版です。

(1)　デモプログラム r_prssrsnsr_iic_mcu_demo_mmr931.c について

MMR9321XA を初期化し、センサデータを取得します。

(2)　電源投入後の待ちについて

電源投入後、圧力センサが Shutdown 状態に遷移するための待ち時間が必要です。ユーザアプリケーションで、システムに合ったタイムアウト時間を設定してください。

(3)　リセットコマンド送信後の待ちについて

リセットコマンド送信後、圧力センサが Shutdown 状態に遷移するための待ち時間が必要です。ユーザアプリケーションで、システムに合ったタイムアウト時間を設定してください。

(4)　アクティブコマンドとリザルトコマンドの送信後の待ちについて

ユーザアプリケーションで、システムに合ったタイムアウト時間を設定してください。

(5)　複数デバイス制御について

複数デバイスをサポートする場合、デバイス毎に割り込み入力端子を割り当ててください。

(6)　最大転送速度について

I^2C シングルマスタ制御ソフトウェアで設定してください。

標準モード・デバイスとファーストモード・デバイスが混載されるバスの場合を想定し、上位層のスレーブデバイス制御ソフトウェアでは設定できません。「1.4.2 最大転送速度」を参照してください。

(7)　電源条件について

圧力センサのデータシートを参照してください。

(8)　リザルトデータバッファポインタの受信データについて

受信データ（24 ビット）は、受信したデータ D23-D16、D15-D8、D7-D0 の順で格納されています

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
1.00	2016.08.31	ー	新規発行
1.01	2016.10.07	12	1.7 ファイル構成　表 1.7 ファイル構成　において、 フォルダ名を更新 注 1 を更新
		30	4.3 ミツミ電機社製 MMR931XA 制御例　において、 最新データシートの内容を反映

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/