RX ファミリ

SRC モジュール Firmware Integration Technology

要旨

本アプリケーションノートは、Firmware Integration Technology (FIT)を使用した SRC (Sampling Rate Converter)モジュールについて説明します。本モジュールは SRC を使用して、PCM データのサンプリングレート変換を行います。以降、本モジュールを SRC モジュールと称します。

対象デバイス

- RX64M グループ
- RX71M グループ

本アプリケーションノートを他のマイコンへ適用する場合、そのマイコンの仕様にあわせて変更し、十分評価してください。

対象コンパイラ

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

各コンパイラの動作確認内容については 4.1 動作確認環境を参照してください。

関連ドキュメント

Firmware Integration Technology ユーザーズマニュアル(R01AN1833)
ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)

目次

1. 概要 3
1.1 SRC モジュールとは 3
1.2 SRC モジュールの概要 3
1.3 API の概要 3
1.4 処理例 4

2. API 情報 5
2.1 ハードウェアの要求 5
2.2 ソフトウェアの要求 5
2.3 サポートされているツールチェーン 5
2.4 使用する割り込みベクタ 5
2.5 ヘッダファイル 6
2.6 整数型 6
2.7 コンパイル時の設定 6
2.8 コードサイズ 7
2.9 引数 8
2.10 戻り値 8
2.11 FIT モジュールの追加方法 9
2.12 for 文、while 文、do while 文について 10

3. API 関数 11
R_SRC_Open () 11
R_SRC_Close () 12
R_SRC_Start () 13
R_SRC_Stop () 15
R_SRC_Write () 16
R_SRC_Read () 18
R_SRC_CheckFlush () 20
R_SRC_GetVersion () 21

4. 付録 22
4.1 動作確認環境 22
4.2 トラブルシューティング 23

5. 参考ドキュメント 24

テクニカルアップデートの対応について 24

改訂記録 25

1. 概要

1.1 SRC 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、「2.11FIT モジュールの追加方法」を参照してください。

1.2 SRC モジュールの概要

SRC モジュールは SRC を使用したサンプリングレート変換を提供します。SRC モジュールは複数の関数からなります。それらを適切な手順で実行することでサンプリングレート変換を行うことができます。

なお、本書で ”モジュール” と表記した場合、SRC モジュールを示します。

1.3 API の概要

表 1.1 SRC モジュールは PCM データのサンプリングレート変換に使用します。
SRC は一般に PCM データの信号源（例：MP3 デコーダ）と SSI (Serial Sound Interface) の間に配置されます。サンプリングレートは PCM データの信号源により様々です。しかし SSI にとってサンプリングレートが一定でないことは好ましくありません。そこで SRC を使用することで SSI に入力する PCM データのサンプリングレートを一定にします。

表 1.1 に本モジュールに含まれる API 関数を示します。

表 1.1　API 関数一覧

関数	関数説明
R_SRC_Open ()	SRC をロックし、r_src_api_rx_config.h の設定に基づき初期化します。
R_SRC_Close ()	SRC のロックを解除します。
R_SRC_Start ()	4 つの引数に応じた設定を行い、サンプリングレート変換を開始します。入力、出力サンプリングレート、入力データと出力データのエンディアン形式を引数で指定します。
R_SRC_Stop ()	サンプリングレート変換を終了するためフラッシュ処理を開始します。なお R_SRC_Stop()実行後フラッシュ処理を完了するため、SRC から全データを読み出すまで R_SRC_Read()を繰り返し実行してください。
R_SRC_Write ()	サンプリングレート変換前の PCM データを SRC の入力データレジスタに書き込みます。書き込む PCM データのサンプル数と入力メモリのアドレスは引数により指定します。なおサンプリングレート変換中、R_SRC_Write()と R_SRC_Read() は繰り返し実行しなければなりません。また実行する回数は互いに異なります。
R_SRC_Read ()	サンプリングレート変換後の PCM データを SRC の出力データレジスタから読み出します。読み出す PCM データのサンプル数と出力メモリのアドレスは引数により指定します。なおサンプリングレート変換中、R_SRC_ Read ()と R_SRC_ Write () は繰り返し実行しなければなりません。また実行する回数は互いに異なります。
R_SRC_CheckFlush ()	フラッシュ処理が完了したかを確認します。
R_SRC_GetVersion ()	モジュールのバージョンを返します。

1.4 処理例

上表の 6 つの関数を使い、以下の手順でサンプリングレート変換を行うことができます。

サンプリングレート変換の基本的な手順;

1) R_SRC_Open() を実行し、SRC をロックし初期化する。
2) R_SRC_Start() を実行し、サンプリングレート変換を開始する。
3) R_SRC_Read() を繰り返し実行し、サンプリングレート変換後の PCM データを読み出す。
4) R_SRC_Write() を繰り返し実行し、サンプリングレート変換前の PCM データを書き込む。
5) R_SRC_Stop() を実行し、サンプリングレート変換を終了する。
6) R_SRC_Close() を実行し、SRC をロック解除する。

2. API 情報

本 FIT モジュールは、下記の条件で動作を確認しています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- SRC

2.2 ソフトウェアの要求

このドライバは以下の FIT モジュールに依存しています。

- ボードサポートパッケージ (r_bsp) v5.00 以上

2.3 サポートされているツールチェーン

本 FIT モジュールは「4.1 動作確認環境」に示すツールチェーンで動作確認を行っています。

2.4 使用する割り込みベクタ

マクロ定義 SRC_IEN が 1 の時、R_SRC_Start 関数を実行すると入力 FIFO エンプティ割り込みが有効になります。

マクロ定義 SRC_OEN が 1 の時、R_SRC_Start 関数を実行すると出力 FIFO フル割り込みが有効になります。

表 2.1 に本 FIT モジュールが使用する割り込みベクタを示します。

表 2.1 使用する割り込みベクター覧

デバイス	割り込みベクタ
RX64M RX71M	入力 FIFO エンプティ 割り込み（ベクタ番号:50） 出力 FIFO フル 割り込み（ベクタ番号:51）

2.5 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_src_api_rx_if.h に記載しています。

2.6 整数型

このドライバは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.7 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_src_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_src_rx_config.h	
SRC_IEN **Default value (0)**	SRCIDCTRL レジスタの IEN ビットを設定します。 (0) 入力 FIFO エンプティ割り込みを禁止する (1) 入力 FIFO エンプティ割り込みを許可する
SRC_OEN **Default value (0)**	SRCODCTRL レジスタの OEN ビットを設定します。 (0) 出力 FIFO フル割り込みを禁止する (1) 出力 FIFO フル割り込みを許可する
SRC_IFTG **Default value (3)**	SRCIDCTRL レジスタの IFTG ビットを設定します。 設定により入力 FIFO のデータ数が下記のとき、SRCSTAT レジスタの IINT ビットがセットされます。 (0) 0 (1) 2 以下 (2) 4 以下 (3) 6 以下
SRC_OFTG **Default value (3)**	SRCODCTRL レジスタの OFTG ビットを設定します。 設定により出力 FIFO のデータ数が下記のとき、SRCSTAT レジスタの OINT ビットがセットされます。 (0) 1 以上 (1) 4 以上 (2) 8 以上 (3) 12 以上
SRC_OCH **Default value (0)**	SRCODCTRL レジスタの OCH ビットを設定します。 (0) チャンネルを入れ替えない。 (1) チャンネルを入れ替える。

2.8　コードサイズ

本モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを下表に示します

ROM (コードおよび定数) と RAM (グローバルデータ) のサイズは、ビルド時の「2.7 コンパイル時の設定」のコンフィギュレーションオプションによって決まります。

下表の値は下記条件で確認しています。

モジュールリビジョン: r_src_api_rx rev1.13

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
(統合開発環境のデフォルト設定に”-lang = c99”オプションを追加)
GCC for Renesas RX 4.8.4.2018.1
(統合開発環境のデフォルト設定に”-std=gnu99”オプションを追加)
IAR C/C++ Compiler for Renesas RX version 4.10.1
(統合開発環境のデフォルト設定)

コンフィグレーションオプション: デフォルト設定

ROM、RAM およびスタックのコードサイズ

デバイス	分類	使用メモリ		
		Renesas Compiler	GCC	IAR Compiler
RX64M	ROM	**30922 バイト**	**36284 バイト**	**31953 バイト**
	RAM	**7420 バイト**	**7224 バイト**	**2100 バイト**
	スタック	**128 バイト**	**-**	**160 バイト**
RX71M	ROM	**30970 バイト**	**36372 バイト**	**32033 バイト**
	RAM	**7432 バイト**	**7236 バイト**	**2112 バイト**
	スタック	**128 バイト**	**-**	**160 バイト**

注 1　BSP を含んだサイズです。

2.9 引数

API 関数で用いられるパラメータは r_src_api_rx_if.h で定義されています。同ファイルはパブリックなインタフェースファイルであり、使用可能な値が定義されています。

2.10 戻り値

本モジュールの API 関数は 3 種類です。

R_SRC_Write() は ”int8_t”、R_SRC_Read() は “int32_t”、その他は全て “src_ret_t” 型です。なお R_SRC_Write() は、正の値で入力データレジスタに書き込んだ PCM データのサンプル数を返します。 また負の値でエラーまたはフラッシュ処理の状態を返します。負の戻り値の定義は ”src_ret_t” と同じです。同様に R_SRC_Read() も負の値を返します。

- int8_t
- int32_t
- src_ret_t

"src_ret_t" はファイル r_src_api_rx_if.h で列挙型として定義されています。

```
typedef enum {
    SRC_SUCCESS    =  0,  /* Function is finished successfully. */
    SRC_ERR_PARAM  = -1,  /* Function is finished unsuccessfully because of
                             incorrect argument. */
    SRC_ERR_UNLOCK = -2,  /* Function is finished unsuccessfully because SRC
                             peripheral is unlocked. */
    SRC_ERR_LOCKED = -3,  /* Function is finished unsuccessfully because SRC
                             peripheral is locked. */
    SRC_NOT_END    = -4,  /* Flush process is not completed. */
    SRC_END        = -5,  /* Flush process is completed. */
} src_ret_t;
```

2.11　FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。ただし、Smart Configurator は、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

(1)　e² studio 上で Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2)　e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加することができます。詳細は、アプリケーションノート「RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」を参照してください。

(3)　CS+上で Smart Configurator を使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版 Smart Configurator を使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(4)　CS+上で FIT モジュールを追加する場合
CS+上で、手動でユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」を参照してください。

2.12 for 文、while 文、do while 文について

本モジュールでは、レジスタの反映待ち処理等で for 文、while 文、do while 文（ループ処理）を使用しています。これらループ処理には、「WAIT_LOOP」をキーワードとしたコメントを記述しています。そのため、ループ処理にユーザがフェイルセーフの処理を組み込む場合は、「WAIT_LOOP」で該当の処理を検索できます。

以下に記述例を示します。

```
while 文の例：
/* WAIT_LOOP */
while(0 == SYSTEM.OSCOVFSR.BIT.PLOVF)
{
    /* The delay period needed is to make sure that the PLL has stabilized. */
}

for 文の例：
/* Initialize reference counters to 0. */
/* WAIT_LOOP */
for (i = 0; i < BSP_REG_PROTECT_TOTAL_ITEMS; i++)
{
    g_protect_counters[i] = 0;
}

do while 文の例：
/* Reset completion waiting */
do
{
    reg = phy_read(ether_channel, PHY_REG_CONTROL);
    count++;
} while ((reg & PHY_CONTROL_RESET) && (count < ETHER_CFG_PHY_DELAY_RESET)); /* WAIT_LOOP */
```

3. API 関数

R_SRC_Open ()

SRC をロックし r_src_api_rx_config.h の設定に基づき初期化します。この関数は他の API 関数を使用する前に実行される必要があります。

Format

src_ret_t R_SRC_Open (void);

Parameters

なし

Return Values

SRC_SUCCESS: */* 正常終了、SRC は設定された。 */*

SRC_ERR_LOCKED : */* 異常終了、SRC はすでにロックされていた。 */*

Properties

r_src_api_rx_if.h にプロトタイプ宣言されています。

Description

SRC を使用する前に必ず一度、本関数を実行してください。

以下の処理を実行します。

- SRC をロックします。
- SRC のモジュールストップ状態を解除します。
- SRC の I/O レジスタを初期化します。
- ファイル r_src_api_rx_config.h を参照し、SRC の I/O レジスタを設定します。
- フィルタ係数（r_src_api_rx_coef.h に記載）をダウンロードします。

Reentrant

- 可能

Special Notes:

なし

R_SRC_Close ()

SRC のロックを解除します。

Format
src_ret_t R_SRC_Close (void);

Parameters
なし

Return Values
SRC_SUCCESS:　　正常終了、SRC はロック解除された。

SRC_ERR_UNLOCK:　　異常終了、SRC はロックされていなかった。

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

Description
SRC の使用を終えるとき、本関数を実行してください。

以下の処理を実行します。

- SRC がロックされているか確認し、ロックされていない場合 SRC_ERR_UNLOCK を返します。
- SRC をロック解除します。
- SRC をモジュールストップ状態に設定します。

Reentrant
- 可能

Special Notes:
なし

R_SRC_Start ()

引数に応じたサンプリングレート変換を開始します。

Format

src_ret_t R_SRC_Start (src_ifs_t fsi, src_ofs_t fso, src_ied_t ied, src_oed_t oed)

Parameters

fsi

サンプリングレート変換前の PCM データのサンプリングレートを指定します。以下に示す定義 src_ifs_t のメンバから一つを選択してください。この定義はファイル r_src_api_rx_if.h に記述されています。

```
typedef enum
{
    SRC_IFS_8  = 0,       /* 8kHz     */
    SRC_IFS_11 = 1,       /* 11.02kHz */
    SRC_IFS_12 = 2,       /* 12kHz    */
    SRC_IFS_16 = 4,       /* 16kHz    */
    SRC_IFS_22 = 5,       /* 22.05kHz */
    SRC_IFS_24 = 6,       /* 24.0kHz  */
    SRC_IFS_32 = 8,       /* 32kHz    */
    SRC_IFS_44 = 9,       /* 44.1kHz  */
    SRC_IFS_48 = 10,      /* 48kHz    */
} src_ifs_t;
```

fso

サンプリングレート変換後の PCM データのサンプリングレートを指定します。以下に示す定義 src_ofs_t のメンバから一つを選択してください。この定義はファイル r_src_api_rx_if.h に記述されています。

```
typedef enum
{
    SRC_OFS_44 = 0,       /* 44.1kHz  */
    SRC_OFS_48 = 1,       /* 48kHz    */
    SRC_OFS_32 = 2,       /* 32kHz    */
    SRC_OFS_8  = 4,       /* 8kHz     */
    SRC_OFS_16 = 5,       /* 16kHz    */
} src_ofs_t;
```

ied

入力する PCM データのエンディアン形式を指定します。以下に示す定義 src_ied_t のメンバから一つを選択してください。この定義はファイル r_src_api_rx_if.h に記述されています。

```
typedef enum {
    SRC_IED_OFF = 0, /* Endian of input data is the same as endian of chip
                        configuration. */
    SRC_IED_ON  = 1, /* Endian of input data is different from endian chip
                        configuration. */
} src_ied_t;
```

oed

出力する PCM データのエンディアン形式を指定します。以下に示す定義 src_oed_t のメンバから一つを選択してください。この定義はファイル r_src_api_rx_if.h に記述されています。

```
typedef enum {
    SRC_OED_OFF = 0, /* Endian of output data is the same as endian of chip
                        configuration. */
    SRC_OED_ON  = 1, /* Endian of output data is different from endian chip
                        configuration. */
} src_oed_t;
```

Return Values

SRC_SUCCESS:　*正常終了、SRC はサンプリングレート変換を開始した。*

SRC_PARAM:　*パラメータが不正である。*

SRC_ERR_UNLOCK:　*SRC がロックされていない。*

SRC_NOT_END:　*フラッシュ処理が完了していない。*

Properties

r_src_api_rx_if.h にプロトタイプ宣言されています。

Description

サンプリングレート変換を開始するときに本関数を実行してください。

サンプリングレート変換を開始するため以下の処理を実行します。

- SRC がロックされているか確認し、ロックされていない場合 SRC_ERR_UNLOCK を返します。
- パラメータが正しいか確認し、正しくない場合 SRC_ERR_PARAM を返します。
- フラッシュ処理中か確認し、処理中なら SRC_NOT_END を返します。
- SRC の内部データをクリアします。
- *ifs* に応じサンプリングレート変換前のサンプリングレートを設定します。
- *ofs* に応じサンプリングレート変換後のサンプリングレートを設定します。
- *ied* に応じ入力 PCM データのエンディアン形式を設定します。
- *oed* に応じ出力 PCM データのエンディアン形式を設定します。
- r_src_api_rx_config.h の設定に応じ、入力データ FIFO エンプティ割り込みと出力データフル割り込みを許可または禁止に設定します。

サンプリングレート変換を開始します。

Reentrant

- 不可

Special Notes:

なし

R_SRC_Stop ()

サンプリングレート変換を終了するためフラッシュ処理を開始します。

Format
src_ret_t R_SRC_Stop (void)

Parameters
なし

Return Values
SRC_SUCCESS:　正常終了、要求は受けつけられた。

SRC_ERR_UNLOCK:　SRC がロックされていない。

SRC_NOT_END:　フラッシュ処理が完了していない。

SRC_END:　フラッシュ処理が完了している。

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

Description
サンプリングレート変換を終了するフラッシュ処理を開始するとき本関数を実行してください。

以下の処理を実行します。

- SRC がロックされているか確認し、ロックされていない場合 SRC_ERR_UNLOCK を返します。
- フラッシュ処理中か確認し、処理中なら SRC_NOT_END を返します。すでに完了している場合、SRC_END を返します。
- 入力データ FIFO エンプティ割り込みを禁止します。
- フラッシュ処理を開始します。

なお R_SRC_Stop() を実行したら、フラッシュ処理を完了させるため、SRC からすべてのデータを読み出すまで R_SRC_Read()を繰り返し実行しなければなりません。

Reentrant
- 不可

Special Notes:
なし

R_SRC_Write ()

サンプリングレート変換前の PCM データを SRC の入力データレジスタに書き込みます。

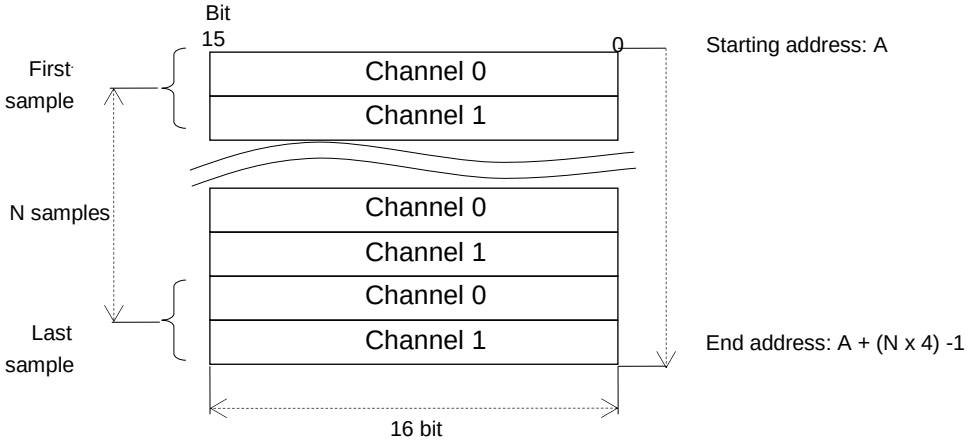
Format

int8_t R_SRC_Write (uint16_t * buf, uint32_t samples)

Parameters

buf

SRC の入力データレジスタに書き込む PCM データバッファの先頭アドレスを指定します。PCM データは下図のようにバッファに配置しなければなりません。また 2 つの 16 ビットの PCM データを 1 サンプルと定義します。



samples

SRC の入力データレジスタに書き込む PCM データのサンプル数を指定します。

Return Values

書き込みサンプル数： SRC の入力データレジスタに書き込んだ PCM データのサンプル数を示します。

SRC_PARAM: パラメータが不正である。

SRC_ERR_UNLOCK: SRC がロックされていない。

SRC_NOT_END: フラッシュ処理が完了していない。

SRC_END: フラッシュ処理が完了している。

Properties

r_src_api_rx_if.h にプロトタイプ宣言されています。

Description

SRC の入力データレジスタにサンプリングレート変換前の PCM データを書き込むために本関数を実行してください。書き込む PCM データのサンプル数とバッファメモリのアドレスは引数で指定します。

以下の処理を実行します。

- SRC がロックされているか確認し、ロックされていない場合 SRC_ERR_UNLOCK を返します。
- フラッシュ処理中か確認し、処理中なら SRC_NOT_END を返します。すでに完了している場合、SRC_END を返します。
- パラメータが正しいか確認し、正しくない場合 SRC_ERR_PARAM を返します。
- パラメータ *buff* と *samples* に応じ PCM データを入力データレジスタに書き込みます。なお入力 FIFO に空きがなければ書き込みをやめ書き込むことができたサンプル数を返します。

なおサンプリングレート変換中、R_SRC_Write()と R_SRC_Read() は繰り返し実行しなければなりません。また実行する回数は互いに異なります。

Reentrant

- 不可

Special Notes:

なし

R_SRC_Read ()

サンプリングレート変換後の PCM データを SRC の出力データレジスタから読み出します。

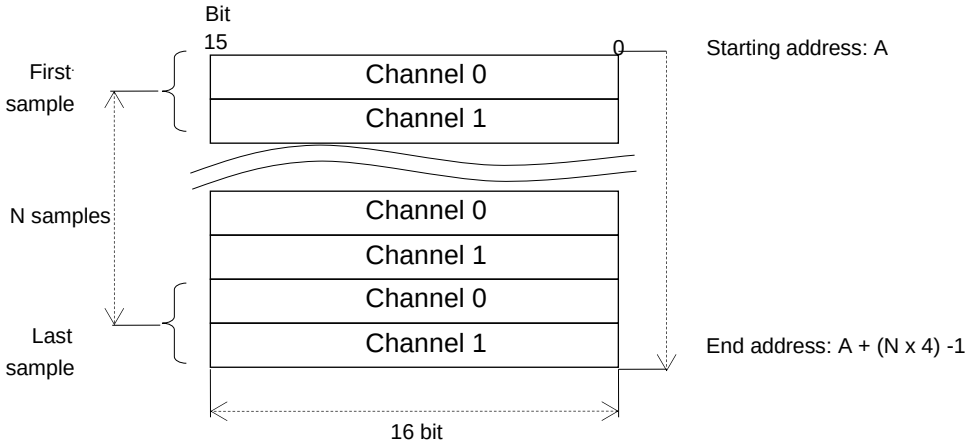
Format

int32_t R_SRC_Read (uint16_t * buf, uint32_t samples)

Parameters

buf

SRC の出力データレジスタから読み出す PCM データを格納する PCM データバッファの先頭アドレスを指定します。PCM データは下図のようにバッファに配置されます。また 2 つの 16 ビットの PCM データを 1 サンプルと定義します。



samples

SRC の出力データレジスタから読み出す PCM データのサンプル数を指定します。

Return Values

読み出しサンプル数 : SRC の出力データレジスタから読み出した PCM データのサンプル数を示します。

SRC_ERR_UNLOCK:SRC がロックされていない。

SRC_NOT_END: フラッシュ処理が完了していない。

SRC_END: フラッシュ処理が完了している。

Properties

r_src_api_rx_if.h にプロトタイプ宣言されています。

Description

SRC の出力データレジスタからサンプリングレート変換後の PCM データを読み出すために本関数を実行してください。読み出す PCM データのサンプル数とバッファメモリのアドレスは引数で指定します。

以下の処理を実行します。

- SRC がロックされているか確認し、ロックされていない場合 SRC_ERR_UNLOCK を返します。
- フラッシュ処理中か確認し、処理中なら SRC_NOT_END を返します。すでに完了している場合、SRC_END を返します。
- パラメータが正しいか確認し、正しくない場合 SRC_ERR_PARAM を返します。
- パラメータ *buf* と *samples* に応じ PCM データを出力データレジスタから読み出します。出力 FIFO が空ならば読み出しをやめ、読み出すことができたサンプル数を返します。

なおサンプリングレート変換中、R_SRC_ Read ()と R_SRC_ Write () は繰り返し実行しなければなりません。また実行する回数は互いに異なります。

Reentrant

- 不可

Special Notes:

なし

R_SRC_CheckFlush ()

フラッシュ処理が完了したかを確認します。

Format
src_ret_t R_SRC_CheckFlush (void)

Parameters
なし

Return Values
SRC_ERR_UNLOCK:　SRC がロックされていない。

SRC_NOT_END:　フラッシュ処理が完了していない。

SRC_END:　フラッシュ処理が完了している。

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

Description
フラッシュ処理が完了したかを確認するために本関数を実行してください。この関数は R_SRC_Write() と R_SRC_Read() を使用せず DMAC で SRC との PCM データ転送をする場合に使用します。

以下の処理を実行します。

・ SRC がロックされているか確認し、ロックされていない場合 SRC_ERR_UNLOCK を返します。

フラッシュ処理中か確認し、処理中なら SRC_NOT_END を返します。すでに完了している場合、SRC_END を返します。

Reentrant
- 不可

Special Notes:
なし

R_SRC_GetVersion ()

モジュールのバージョンを返します。

Format
uint32_t R_SRC_GetVersion (void)

Parameters
なし

Return Values
一つの 32 ビットの数値に格納されたメジャーとマイナーからなるバージョン番号。

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

Description
この関数は本モジュールのバージョンを返します。上位２バイトはメジャーバージョン番号、下位２バイトはマイナーバージョン番号を示します。

Reentrant
- 可能

Example
```
/* Retrieve the version number and convert it to a string. */

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

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

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

Special Notes:
なし

4. 付録

4.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 4.1 動作確認環境 (Rev.1.13)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.1
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.8.4.2018.01 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.1 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.13
使用ボード	Renesas Starter Kit+ for RX64M（型名：R0K50564MS800BE）

4.2　トラブルシューティング

(1) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「Could not open source file "platform.h"」エラーが発生します。

A：FIT モジュールがプロジェクトに正しく追加されていない可能性があります。プロジェクトへの追加方法をご確認ください。

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e^2 studio を使用している場合
  アプリケーションノート RX ファミリ e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

また、本 FIT モジュールを使用する場合、ボードサポートパッケージ FIT モジュール(BSP モジュール)もプロジェクトに追加する必要があります。BSP モジュールの追加方法は、アプリケーションノート「ボードサポートパッケージモジュール(R01AN1685)」を参照してください。

(2) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「This MCU is not supported by the current r_src_api_rx module.」エラーが発生します。

A：追加した FIT モジュールがユーザプロジェクトのターゲットデバイスに対応していない可能性があります。追加した FIT モジュールの対象デバイスを確認してください。

(3) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行すると「the value for macro マクロ名 must…」エラーが発生します。

A：“r_src_api_rx_config.h”ファイルの設定値が間違っている可能性があります。
“r_src_api_rx_config.h”ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

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 コンパイラ ユーザーズマニュアル（R20UT3248）
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデートの対応について

本モジュールは以下のテクニカルアップデートの内容を反映しています。

対応しているテクニカルアップデートはありません。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jul.11.14	—	初版発行
1.10	Sep.9.14	4	「2.2.2 メモリ使用量」をソフトウェバージョン 1.10 に合わせ変更。
		5	表 1.1 の SRC_OFTG の説明の誤記を修正.
		5..7	2.7 から 2.10 までの節番号の振り直し。
1.11	Dec.12.14	—	・ Added support for the RX71M Group.RX71M グループのサポートを追加。
		4	「2.3 ソフトウェアの要求」の r_bsp バージョンを v2.60 以上から v2.80 以上に変更。
	Apr.7.17	—	日本語版発行
1.12	Feb.1.19	—	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。
1.13	May.20.19	—	以下のコンパイラに対応 ・GCC for Renesas RX ・IAR C/C++ Compiler for Renesas RX
		1	「対象コンパイラ」の章を追加
		3	1.3「API の概要」の章を更新
		5	2.4「使用する割り込みベクタ」の章を追加
		7	2.8 「コードサイズ」章を追加
		10	2.12「for 文、while 文、do while 文について」の章を追加
		22	4.1 「動作確認環境」の章を追加

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)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。