RX ファミリ
SCIFA クロック同期式シングルマスタ制御モジュール
Firmware Integration Technology

R01AN2280JJ0109
Rev.1.09
2016.09.30

要旨

本アプリケーションノートではRX ファミリ ルネサス FIFO 内蔵シリアルコミュニケーションインタフェース（以下、SCIFA と略す）のクロック同期式シリアル通信を使用したクロック同期式シングルマスタ制御モジュールとその使用方法を説明します。本モジュールは、Firmware Integration Technology （以下、FIT と略す）を使ったクロック同期式シングルマスタ制御モジュールです。以降、本モジュールを SCIFA FIT モジュールと称します。また、同様に FIT を使った他の機能制御モジュールを FIT モジュールもしくは“機能名”FIT モジュールと表します。

ポート制御によるスレーブデバイスセレクト制御を付加することにより、SPI モード・シングルマスタ制御が可能です。

SCIFA FIT モジュールは、シングルマスタ基本制御方法を実現したものです。SCIFA FIT モジュールを使用し、スレーブデバイスを制御するためのソフトウェアを作成してください。

対象デバイス

対応 MCU

RX64M グループ

RX71M グループ

動作確認に使用したデバイス

ルネサス エレクトロニクス製 R1EX25xxx シリーズ Serial EEPROM 16Kbit

Macronix International 社製 MX25/66L family serial NOR Flash Memory 32Mbit

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合わせて変更し、十分評価してください。

なお、以降では、対象デバイスが、複数グループ MCU であるため、説明の都合上、“RX ファミリ MCU”として記述しています。

関連ドキュメント

- Firmware Integration Technology ユーザーズマニュアル (R01AN1833JU)
- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685JU)
- e^2 studio に組み込む方法 Firmware Integration Technology (R01AN1723JU)
- CS+に組み込む方法 Firmware Integration Technology (R01AN1826JJ)

目次

1. 概要

RX ファミリ MCU 内蔵の SCIFA を使用し、クロック同期式制御を行います。ポート制御によるスレーブデバイスセレクト制御を付加することにより、SPI モード・シングルマスタ制御が可能です。

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、機能概略を示します。

- マスタデバイスを RX ファミリ MCU とする SCIFA を使ったクロック同期式シングルマスタ用ブロック型デバイスドライバ
- SCIFA のクロック同期式シリアル通信機能を使用。ユーザ設定した 1 チャネルもしくは複数チャネルの制御が可能
- 異なるチャネルからリエントラントが可能
- スレーブデバイスセレクト制御を未サポート
  別途ポート制御によるスレーブデバイスセレクト制御の組み込みが必要です。
- ビッグエンディアン／リトルエンディアンでの動作が可能
- MSB ファーストフォーマットで転送
- ソフトウェア転送のみをサポート
  DMAC転送もしくはDTC転送を使用する場合、別途DMAC転送もしくはDTC転送のプログラムが必要です。

表 1-1　使用する周辺機器と用途

周辺機器	用途
SCIFA	クロック同期式（3 線式）シリアル：1 もしくは複数チャネル（必須）
Port	スレーブデバイスセレクト制御信号用：使用デバイス数分のポートが必要（必須） ただし、SCIFA FIT モジュールでは扱いません。

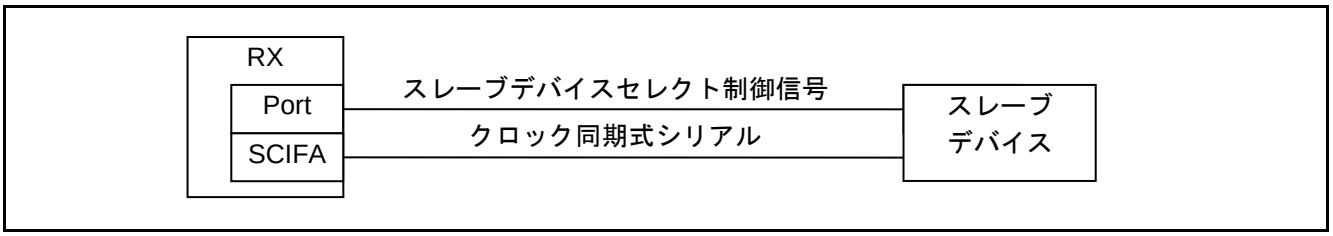


図 1-1　使用例

1.1 SCIFA FIT モジュール

SCIFA FIT モジュールは、他の FIT モジュールと組み合わせることにより、組み込みが容易になります。

また、SCIFA FIT 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。SCIFA FIT モジュールの組み込み方については、「2.9 モジュールの追加方法」「2.10 SCIFA 以外の周辺機能とモジュール」を参照してください。

1.2 API の概要とメモリサイズ

1.2.1 API の概要

表 1-2に SCIFA FIT モジュールに含まれる API 関数を示します。

表 1-2 API 関数

関数名	説明
R_SCIFA_SMstr_Open()	ドライバの初期化処理
R_SCIFA_SMstr_Close()	ドライバの終了処理
R_SCIFA_SMstr_Control()	ドライバのコントロール（ビットレート）設定処理
R_SCIFA_SMstr_Write()　注１	シングルマスタ送信処理
R_SCIFA_SMstr_Read()　注１	シングルマスタ受信処理
R_SCIFA_SMstr_WriteRead()　注１	シングルマスタ送受信（全二重通信）処理
R_SCIFA_SMstr_Get_BuffRegAddress()	FTDR レジスタ及び FRDR レジスタアドレス取得処理
R_SCIFA_SMstr_Int_Txif_Ier_Clear()	TXIF 送信割り込み要求禁止処理
R_SCIFA_SMstr_Int_Rxif_Ier_Clear()	RXIF 受信割り込み要求禁止処理
R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set()	DMAC／DTC 送信完了フラグ設定処理
R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set()	DMAC／DTC 受信完了フラグ設定処理
R_SCIFA_SMstr_GetVersion()	ドライバのバージョン情報取得処理
R_SCIFA_SMstr_Set_LogHdlAddress()	LONGQ FIT モジュールのハンドラアドレス設定処理
R_SCIFA_SMstr_Log()	LONGQ FIT モジュールを使ったエラーログ取得処理
R_SCIFA_SMstr_1ms_Interval()　注２	インターバルタイマカウント処理

注１：SCIFA 制御の高速化のために、FTDR レジスタ及び FRDR レジスタを 32 ビットアクセスします。送信／受信データ格納バッファポインタを指定する場合、開始アドレスを 4 バイト境界に合わせてください。

注２：DMAC 転送もしくは DTC 転送を使用する場合、タイムアウト検出のため、ハードウェアタイマやソフトウェアタイマ等を使って、1ms 間隔でコールする必要があります。

1.2.2　動作環境とメモリサイズ

(1)　RX64M の場合

表 1-3に動作確認条件、表 1-4に SCIFA FIT モジュール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
使用マイコン	RX64M グループ（プログラム ROM 4MB／RAM 512KB）
動作周波数	ICLK：120MHz、PCLKA：120MHz
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 e2 studio V3.1.0.24
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 V.2.01.00
	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
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.1.08
使用ボード	R0K50564MSxxxBE（Renesas Starter Kit for RX64M）

表 1-4　必要メモリサイズ

使用メモリ	サイズ	備考
ROM	3,288 バイト （リトルエンディアン）	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c 上記の動作確認条件による
RAM	22 バイト （リトルエンディアン）	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c 上記の動作確認条件による
最大使用ユーザスタック	76 バイト	
最大使用割り込みスタック	4 バイト	DMAC 転送もしくは DTC 転送を設定時のみ

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。

(2) RX71M の場合

表 1-5に動作確認条件、表 1-6に SCIFA FIT モジュール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 1-5 動作確認条件

項目	内容
使用マイコン	RX71M グループ（プログラム ROM 4MB／RAM 512KB）
動作周波数	ICLK：240MHz、PCLKA：120MHz
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 e2 studio V3.1.2.09
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 V.2.01.00
	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
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.1.08
使用ボード	R0K50571MSxxxBE（Renesas Starter Kit for RX71M）

表 1-6 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	3,288 バイト （リトルエンディアン）	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c 上記の動作確認条件による
RAM	22 バイト （リトルエンディアン）	r_scifa_smstr.c r_scifa_smstr_target.c r_scifa_smstr_target_dev_port.c 上記の動作確認条件による
最大使用ユーザスタック	76 バイト	
最大使用割り込みスタック	4 バイト	DMAC 転送もしくは DTC 転送を設定時のみ

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。

1.3　関連アプリケーションノート

SCIFA FIT モジュール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。

- RX Family LONGQ Module Using Firmware Integration Technology（R01AN1880EU）
- RX ファミリ DMA コントローラ DMACA 制御モジュール Firmware Integration Technology（R01AN2063JJ）
- RX Family DTC モジュール Firmware Integration Technology（R01AN1819JJ）
- RX ファミリ コンペアマッチタイマ (CMT)モジュール Firmware Integration Technology（R01AN1856JU）
- RX ファミリ EEPROM アクセス クロック同期式制御モジュール Firmware Integration Technology（R01AN2325JJ）
- RX Family General Purpose Input/Output Driver Module Using Firmware Integration Technology（R01AN1721EU）
- RX Family Multi-Function Pin Controller Module Using Firmware Integration Technology（R01AN1724EU）
- RX ファミリ Serial Flash memory アクセス クロック同期式制御モジュール Firmware Integration Technology（R01AN2662JJ）

1.4　ハードウェア設定

1.4.1　ハードウェア構成例

図 1-2に接続図を示します。なお、高速で動作させた場合を想定し、各信号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やコンデンサの付加を検討してください。

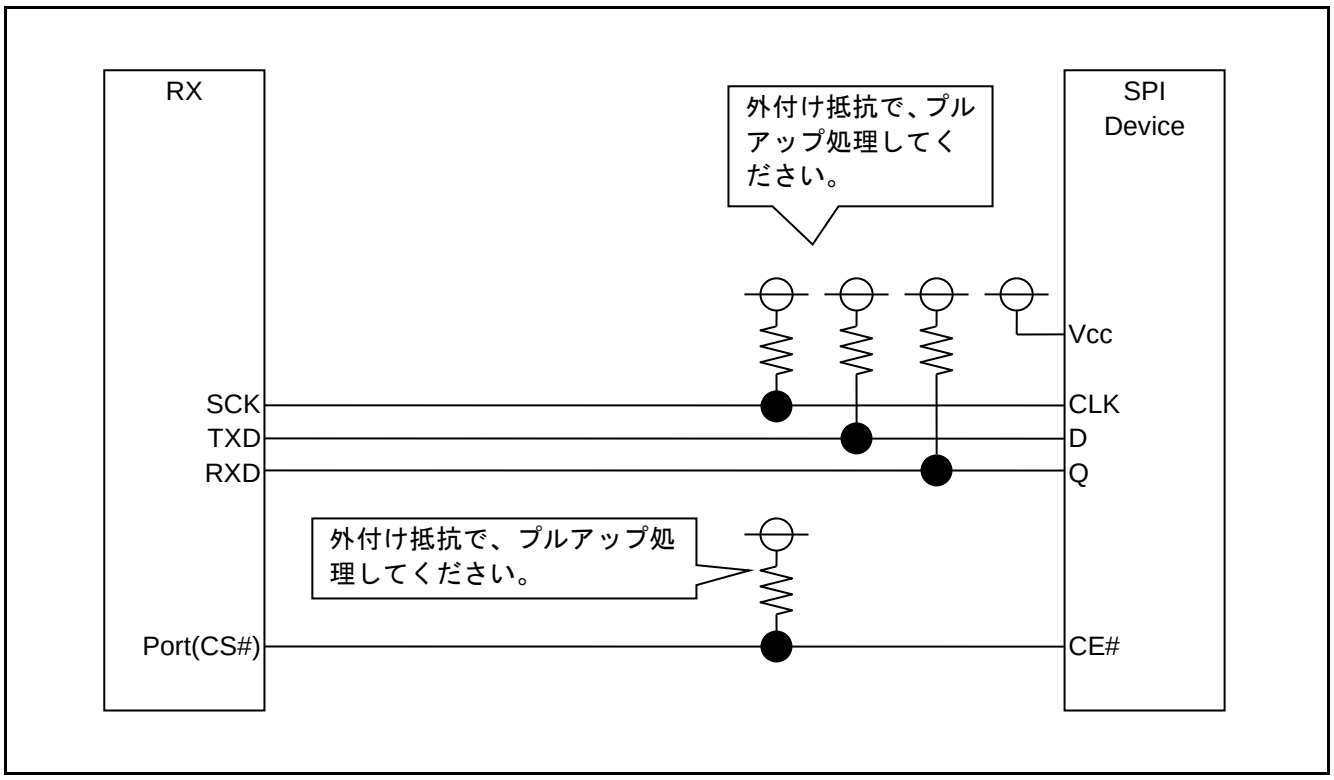


図 1-2　RX ファミリ MCU SCIFA と SPI スレーブデバイスの接続例

1.4.2　使用端子一覧

表 1-7に、使用端子と機能を示します。

表 1-7　使用端子と機能

端子名	入出力	内容
SCK	出力	クロック出力
TXD	出力	マスタデータ出力
RXD	入力	マスタデータ入力
Port（図 1-2の Port(CS#)）	出力	スレーブデバイスセレクト出力 ただし、SCIFA FIT モジュールでは、扱いません。

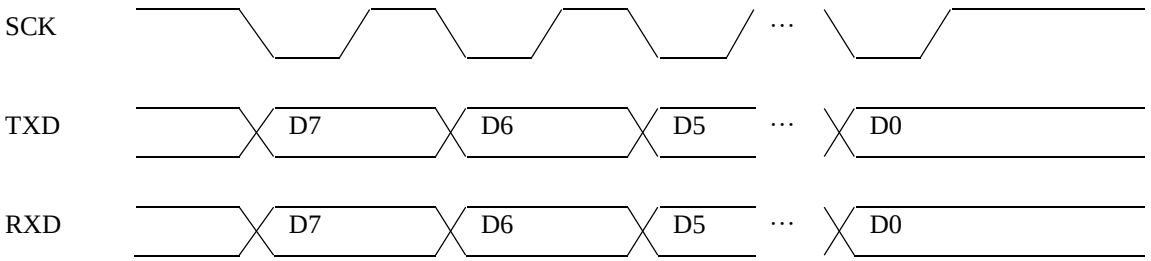
1.5　ソフトウェア説明

1.5.1　動作概要

SCIFA のクロック同期式シリアル通信機能を使って、内部クロックを使用したクロック同期式シングルマスタ制御（シングルマスタ送信、シングルマスタ受信、シングルマスタ送受信）を実現します。

1.5.2　制御可能なスレーブデバイス

制御可能なスレーブデバイスは、図 1-3に示す SPI モード 3（CPOL=1、CPHA=1）をサポートしたスレーブデバイスです。



・MCU->スレーブデバイスの送信時：転送クロックの立ち下がりで送信データ出力開始

・スレーブデバイス->MCU の受信時：転送クロックの立ち上がりで受信データの入力取り込み

・MSB ファーストでの転送

転送を行っていないときの SCK 端子のレベルは、”H”です。

図 1-3　制御可能なスレーブデバイスのタイミング

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。

1.5.3　スレーブデバイスの CE#端子制御

SCIFA FIT モジュールは、スレーブデバイスの CE#端子を制御しません。スレーブデバイスを制御する場合、別途、スレーブデバイスの CE#端子の制御を追加してください。

制御方法としては、MCU の Port に接続し MCU 汎用ポート出力で制御してください。

また、スレーブデバイスの CE#（MCU の Port(CS#)）信号の立ち下がりから、スレーブデバイスの CLK（MCU の SCK）信号の立ち下がりまでの時間（スレーブデバイスの CE#セットアップ時間）を設けてください。

同様に、スレーブデバイスの CLK（MCU の SCK）信号の立ち上がりから、スレーブデバイスの CE#（MCU の Port(CS#)）信号の立ち上がりまでの時間（スレーブデバイスの CE#ホールド時間）を設けてください。

スレーブデバイスのデータシートを確認して、システムに応じたソフトウェア・ウェイト時間を設定してください。

1.5.4　ソフトウェア構成

図 1-4にソフトウェア構成を示します。

SCIFA FIT モジュールを使用して、スレーブデバイスを制御するためのソフトウェアを作成してください。

なお、スレーブデバイス制御のためのソフトウェア例を用意していますので、入手してください。

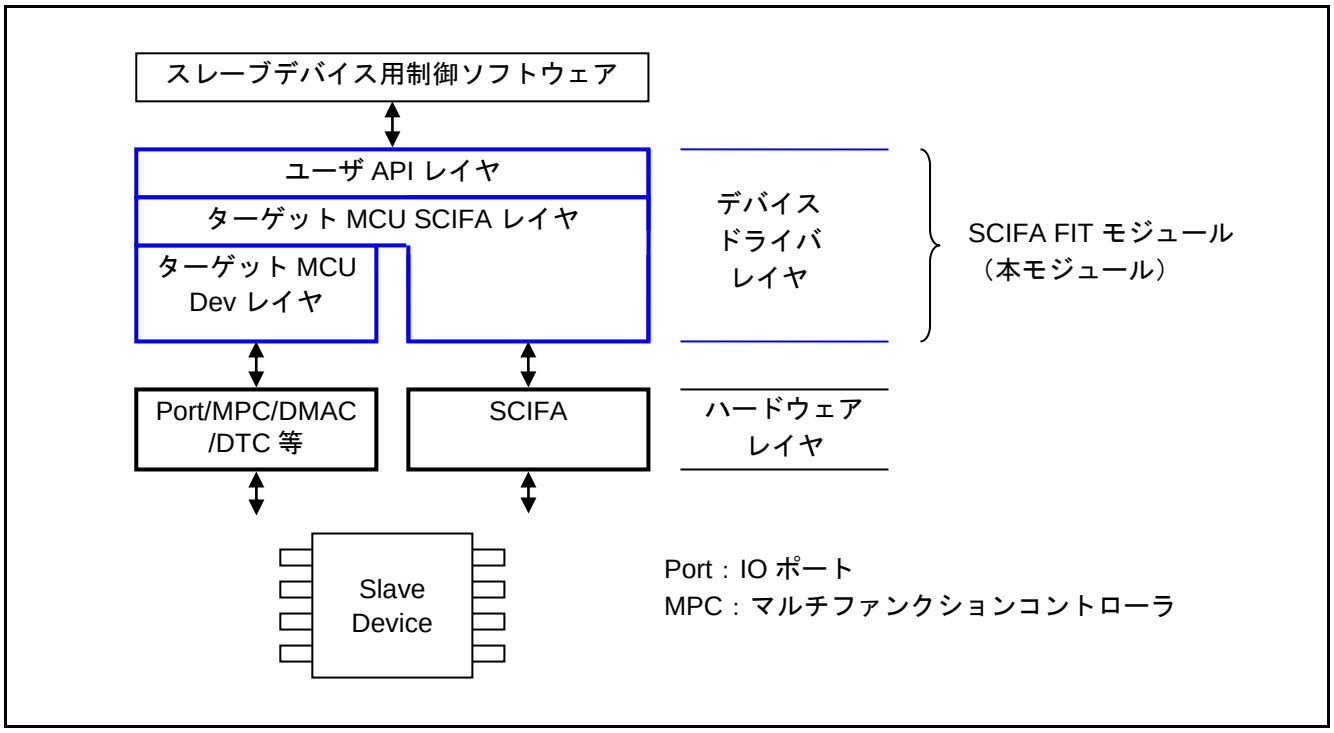


図 1-4　ソフトウェア構成

(a)　**ユーザ API レイヤ**（r_scifa_smstr.c）

SCIFA クロック同期式シングルマスタ制御部で、MCU や SCIFA の仕様に依存しない部分です。

また、DMAC 制御もしくは DTC 制御に必要な DMAC／DTC の転送起動設定処理が含まれています。DMAC FIT モジュールもしくは DTC FIT モジュールと組み合わせて使用できます。

(b)　**ターゲット MCU SCIFA レイヤ**（r_scifa_smstr_target.c）

SCIFA のリソース制御部分です。

チャネル数や SCIFA の仕様の違い等を MCU 毎に提供します。

(c)　**ターゲット MCU Dev レイヤ**（r_scifa_smstr_target_dev_port.c）

SCIFA 以外の IO ポート（以下、GPIO と略す）／マルチファンクションピンコントローラ（以下、MPC と略す）等の制御部分です。GPIO FIT モジュールと MPC FIT モジュールの使用が可能です。他の FIT モジュール等を使用してシステムに合わせて組み込みが必要です。

(d)　**スレーブデバイス用制御ソフトウェア**

例として、「RX ファミリ EEPROM アクセス クロック同期式モジュール Firmware Integration Technology（R01AN2325JJ）」を参照してください。この Serial EEPROM 制御ソフトウェアには、FIT モジュールとの合わせ込みのためのドライバ I/F 関数（r_eeprom_spi_drvif_devX.c：X=0or1）があります。

1.5.5　データバッファと送信／受信データの関係

SCIFA FIT モジュールは、ブロック型デバイスドライバであり、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。

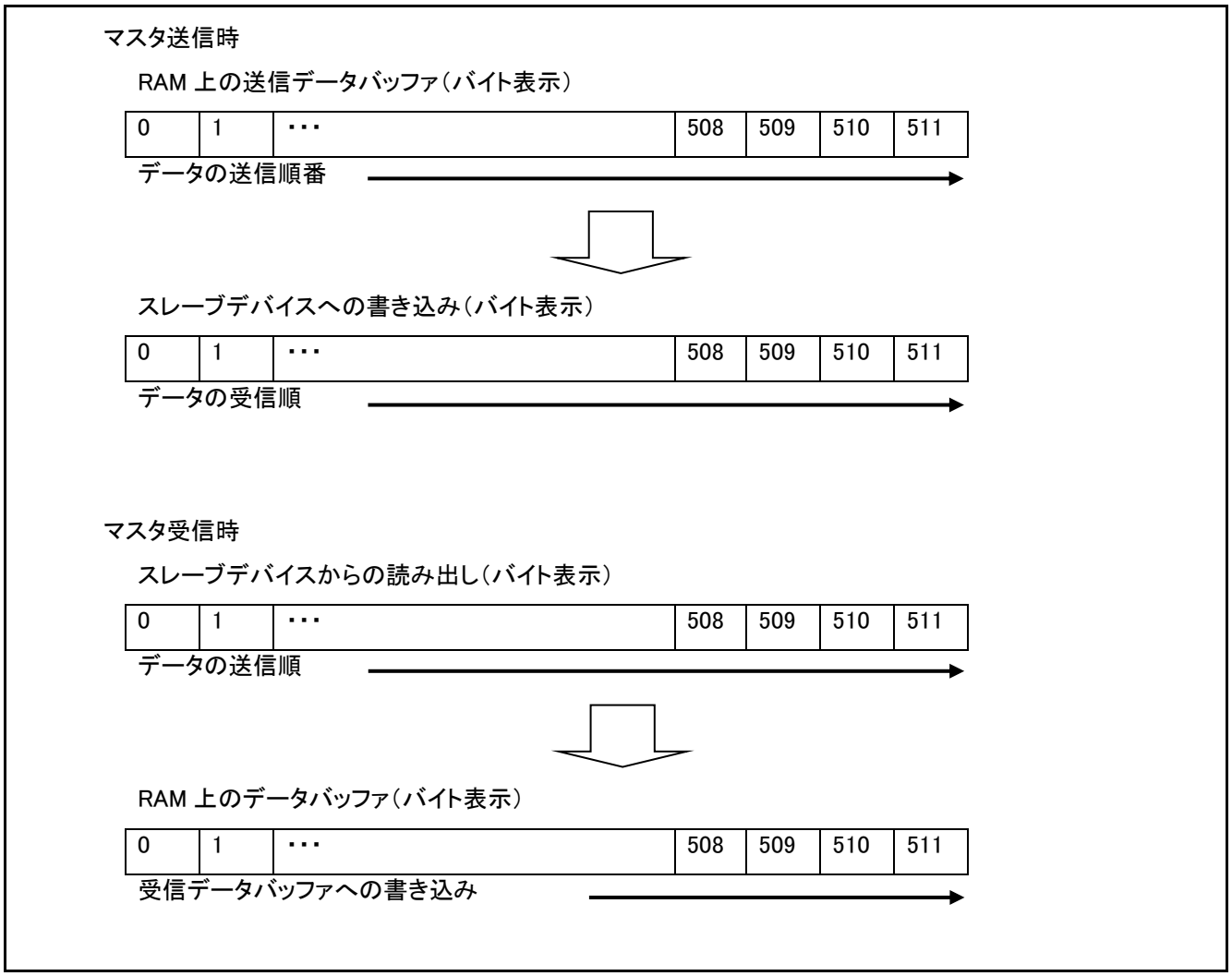


図 1-5　データバッファと送信／受信データの関係

1.5.6 状態遷移図

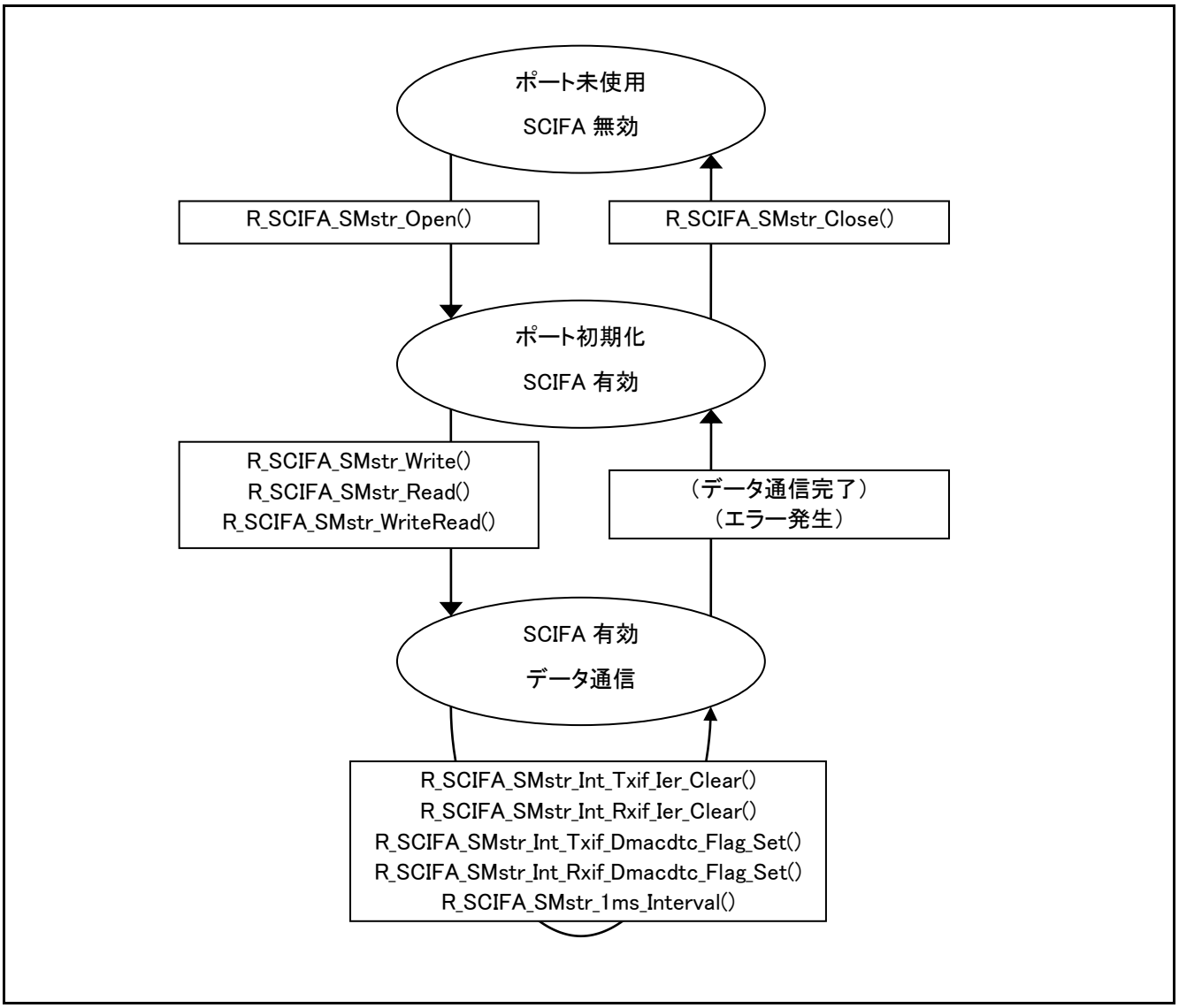


図 1-6 状態遷移図

2. API 情報

SCIFA FIT モジュールの API は、ルネサスの API 命名基準に従っています。

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- SCIFA

2.2 ソフトウェアの要求

このドライバは以下のパッケージに依存しています。

- r_bsp
- r_cgc_rx（クロック発生回路 CGC FIT モジュールが必要な RX ファミリ MCU を使用する場合のみ）
- r_dmaca_rx（DMACA FIT モジュールを用いて、DMAC 転送を使用する場合のみ）
- r_dtc_rx（DTC FIT モジュールを用いて、DTC 転送を使用する場合のみ）
- r_cmt_rx（DMAC 転送もしくは DTC 転送を使用し、かつコンペアマッチタイマ CMT FIT モジュールを使用する場合のみ）
  他タイマやソフトウェアタイマで代用できます。
- r_gpio_rx（GPIO, MPC FIT モジュールを用いて、GPIO を制御する場合のみ）
- r_mpc_rx（GPIO, MPC FIT モジュールを用いて、MPC を制御する場合のみ）

2.3 サポートされているツールチェイン

SCIFA FIT モジュールは、「1.2.2」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_scifa_smstr_rx_if.h に記載しています。

ビルド毎の構成オプションは、r_scifa_smstr_rx_config.h と r_scifa_smstr_rx_pin_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_scifa_smstr_rx_if.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

SCIFA FIT 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_scifa_smstr_rx_config.h と r_scifa_smstr_rx_pin_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_scifa_smstr_rx_config.h*	
#define SCIFA_SMSTR_CFG_USE_FIT ※デフォルト値は“有効”	SCIFA FIT モジュールの BSP 環境での使用有無を選択できます。 無効にした場合、r_bsp 等の FIT モジュール制御を無効します。また、別途、処理を組み込む必要があります。詳細は「2.11 FIT モジュール以外の環境で使用する場合の組み込み方法」を参照ください。 有効にした場合、r_bsp 等の FIT モジュール制御を有効にします。
#define SCIFA_SMSTR_CFG_CHx_INCLUDED ※チャネル 8 のデフォルト値は“有効” ※“x”はチャネル番号	該当チャネルを使用するかを選択できます。 無効にした場合、該当チャネルに関する処理をコードから省略します。 有効にした場合、該当チャネルに関する処理をコードに含めます。
#define SCIFA_SMSTR_CFG_LONGQ_ENABLE ※デフォルトは“無効”	デバッグ用のエラーログ取得処理を使用するか選択できます。 無効にした場合、処理をコードから省略します。 有効にした場合、処理をコードに含めます。 使用するためには、別途 LONGQ FIT モジュールが必要です。
#define SCIFA_SMSTR_CFG_CHx_INT_TXIF_LEVEL ※チャネル 8 のデフォルト値は“10” ※“x”はチャネル番号	DMAC 転送もしくは DTC 転送を使用する場合に送信 FIFO データエンプティ割り込み（TXIF）の割り込みレベルを設定してください。
#define SCIFA_SMSTR_CFG_CHx_INT_RXIF_LEVEL ※チャネル 8 のデフォルト値は“10” ※“x”はチャネル番号	DMAC 転送もしくは DTC 転送を使用する場合に受信 FIFO データフル割り込み（RXIF）の割り込みレベルを設定してください。

Configuration options in *r_scifa_smstr_rx_pin_config.h*	
#define R_SCIFA_SMSTR_CFG_SCIFAx_SCKx_PORT ※チャネル 8 のデフォルト値は“‘C’” ※“x”はチャネル番号	SCIFA の SCK 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_SCIFA_SMSTR_CFG_SCIFAx_SCKx_BIT ※チャネル 8 のデフォルト値は“‘5’” ※“x”はチャネル番号	SCIFA の SCK 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_SCIFA_SMSTR_CFG_SCIFAx_TXDx_PORT ※チャネル 8 のデフォルト値は“‘C’” ※“x”はチャネル番号	SCIFA の TXD 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_SCIFA_SMSTR_CFG_SCIFAx_TXDx_BIT ※チャネル 8 のデフォルト値は“‘7’” ※“x”はチャネル番号	SCIFA の TXD 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_SCIFA_SMSTR_CFG_SCIFAx_RXDx_PORT ※チャネル 8 のデフォルト値は“‘C’” ※“x”はチャネル番号	SCIFA の RXD 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_SCIFA_SMSTR_CFG_SCIFAx_RXDx_BIT ※チャネル 8 のデフォルト値は“‘6’” ※“x”はチャネル番号	SCIFA の RXD 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。

2.7 引数

API 関数の引数である構造体を示します。この構造体は API 関数のプロトタイプ宣言とともに r_scifa_smstr_rx_if.h で記載されています。

```
typedef struct
{
    uint32_t  data_cnt;                 /* Number of data (byte unit)        */
    uint8_t * p_tx_data;                /* Pointer to transmit data buffer   */
    uint8_t * p_rx_data;                /* Pointer to receive data buffer    */
    scifa_smstr_tranmode_t tran_mode;   /* Data transfer mode                */
} scifa_smstr_info_t;
```

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_scifa_smstr_rx_if.h で記載されています。

```
typedef enum e_scifa_smstr_status
{
    SCIFA_SMSTR_SUCCESS    = 0,         /* Successful operation              */
    SCIFA_SMSTR_ERR_PARAM  = -1,        /* Parameter error                   */
    SCIFA_SMSTR_ERR_HARD   = -2,        /* Hardware error                    */
    SCIFA_SMSTR_ERR_OTHER  = -7         /* Other error                       */
} scifa_smstr_status_t;
```

2.9　モジュールの追加方法

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

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

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

FIT プラグインを使用して FIT モジュールを追加するには以下の方法があります。

1.「FIT Configurator」を使用する。
lib ファイルパスの自動設定等、プラグイン機能が強化された最新の方法です。本方法の使用を推奨します。手順は、アプリケーションノート「RX ファミリ RX Driver Package Ver.1.10 (R01AN3345JJ)」の「4.3.2 FIT プラグインで FIT モジュールをインストールする」を参照してください。

2.従来の「FITプラグイン」を使用する。
手順は、アプリケーションノート「e2 studio に組み込む方法 Firmware Integration Technology (R01AN1723JU)」の「3. FIT プラグインを使用して FIT モジュールをプロジェクトに追加する方法」を参照してください。

2.10 SCIFA 以外の周辺機能とモジュール

SCIFA FIT モジュールは、SCIFA、GPIO FIT モジュール、MPC FIT モジュール以外に、以下の周辺機能およびモジュールを制御します。

- DMA コントローラ（以下、DMAC と略す）
- データトランスファコントローラ（以下、DTC と略す）
- コンペアマッチタイマ（以下、CMT と略す）　DMAC 転送もしくは DTC 転送使用時に必要
- ロングキュー（以下、LONGQ と略す）－ソフトウェアモジュール

LONGQ 以外の制御は FIT モジュールを使用していません。そのため、FIT モジュールを使用する環境で動作させる場合、SCIFA 以外の周辺機能の制御処理を FIT モジュールに置き換えることを推奨します。

対象のソースコードは“r_scifa_smstr_target_dev_port.c”に含まれます。

2.10.1 DMAC／DTC

DMAC 転送もしくは DTC 転送を使用する場合の制御方法を説明します。

SCIFA FIT モジュールでは、ICU.IERm.IENj ビットセットによる DMAC／DTC の転送起動、および転送完了待ちを行います。その他の DMAC レジスタもしくは DTC レジスタへの設定は DMAC FIT モジュールもしくは DTC FIT モジュールを使用するか、ユーザ独自で処理を作成してください。

なお、DMAC 転送設定の場合、DMAC 転送が完了した際の ICU.IERm.IENj ビットのクリア、および転送完了フラグのクリアはユーザが行う必要があります。

表 2-1に示す制御関数を使って、各々の処理を実現してください。

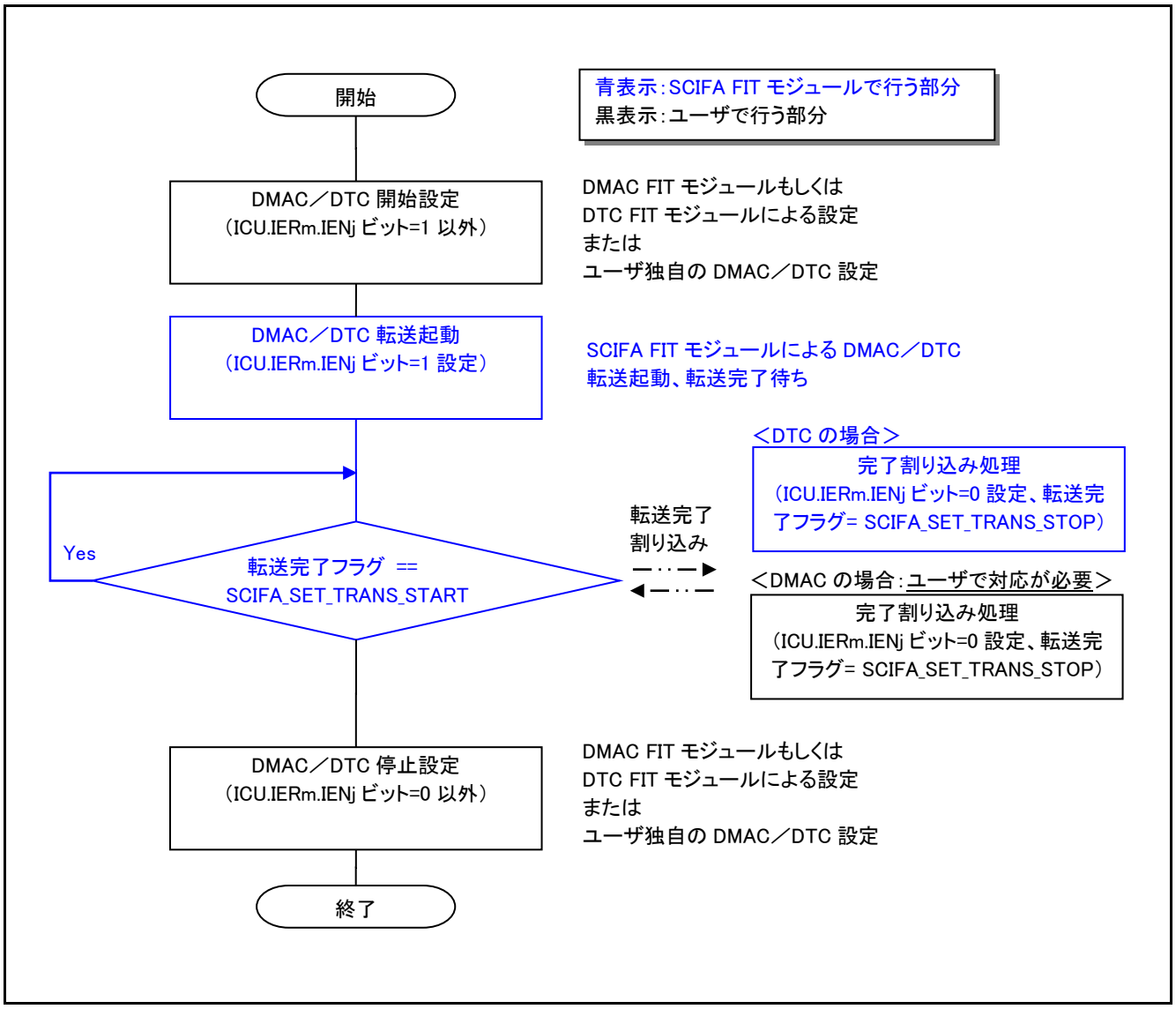


図 2-1　DMAC 転送および DTC 転送設定時の処理

DMAC／DTC 制御に関連する制御関数と処理内容について表 2-1に示します。

データ送信完了待ち処理 r_scifa_smstr_tx_dmacdtc_wait()、およびデータ受信完了待ち処理 r_scifa_smstr_rx_dmacdtc_wait()は、1ms タイマを用いて完了待ちを行います。そのため、事前にユーザシステムにて CMT 等を用いて 1ms タイマを起動してください。そして、コールバック関数等を用いて 1ms 毎に R_SCIFA_SMstr_1ms_Interval()をコールしてください。

表 2-1　制御関数と処理内容

関数名	処理内容
r_scifa_smstr_txif_isrX()	SCIFA channel “X” TXIF 割り込みハンドラ処理（X はチャネル番号）
r_scifa_smstr_rxif_isrX()	SCIFA channel “X” RXIF 割り込みハンドラ処理（X はチャネル番号）
r_scifa_smstr_tx_dmacdtc_wait()	DMAC／DTC 送信完了待ち処理
r_scifa_smstr_rx_dmacdtc_wait()	DMAC／DTC 受信完了待ち処理
r_scifa_smstr_int_txif_init()	TXIF 割り込み初期化処理
r_scifa_smstr_int_rxif_init()	RXIF 割り込み初期化処理
r_scifa_smstr_int_txif_ier_set()	TXIF 割り込みの ICU.IERm.IENj ビットをセット
r_scifa_smstr_int_rxif_ier_set()	RXIF 割り込みの ICU.IERm.IENj ビットをセット
R_SCIFA_SMstr_Int_Txif_Ier_Clear()	TXIF 割り込みの ICU.IERm.IENj ビットをクリア
R_SCIFA_SMstr_Int_Rxif_Ier_Clear()	RXIF 割り込みの ICU.IERm.IENj ビットをクリア
R_SCIFA_SMstr_Int_Txif_Dmacdtc_flag_Set()	送信用の DMAC／DTC 転送完了フラグを設定
R_SCIFA_SMstr_Int_Rxif_Dmacdtc_flag_Set()	受信用の DMAC／DTC 転送完了フラグを設定
R_SCIFA_SMstr_1ms_Interval()	各チャネルの内部タイマカウンタをインクリメント

2.10.2　CMT

DMAC 転送もしくは DTC 転送を使用する場合に必要です。転送タイムアウト検出目的で使用します。

2.10.3　LONGQ

エラーログ取得機能で使用する FIT モジュールです。

SCIFA FIT モジュールに LONGQ FIT モジュールを使用した制御例が含まれています。SCIFA FIT モジュールのコンフィギュレーションオプションのデフォルトは、エラーログ取得機能無効設定です。「2.6 コンパイル時の設定」を参照してください。

(1)　R_LONGQ_Open()の設定

LONGQ FIT モジュールの R_LONGQ_Open()の引数 ignore_overflow を“1”に設定してください。これによりエラーログバッファは、リングバッファとして使用することが可能です。

(2)　制御手順

R_SCIFA_SMstr_Open()をコールする前に、以下の関数を順番にコールしてください。

1. R_LONGQ_Open()
2. R_SCIFA_SMstr_Set_LogHdlAddress()

2.11　FIT モジュール以外の環境で使用する場合の組み込み方法

r_bsp 等の FIT モジュールを使用しない環境下で動作させる場合、以下を実施してください。

#r_scifa_smstr_rx_config.h の「#define SCIFA_SMSTR_CFG_USE_FIT」を無効にしてください。

#r_scifa_smstr_rx_if.h の#include "platform.h"をコメントアウトしてください。

#r_scifa_smstr_rx_if.h に以下のヘッダファイルをインクルードしてください。

```
#include “iodefine.h”
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
#include <machine.h>
```

#r_scifa_smstr_rx_if.h に「#define BSP_MCU_RXxxx（xxx は MCU 名、英字は大文字）」を定義してください。例えば RX64M であれば BSP_MCU_RX64M としてください。

#r_scifa_smstr_rx_if.h に以下の enum 定義を追加してください。また、以下の#define 定義を追加してください。「BSP_ICLK_HZ」にはシステムクロック（ICLK）の値、「BSP_PCLKA_HZ」には周辺モジュールクロック（PCLKA）の値を設定してください。なお、これらの定義は他の FIT モジュールの定義と重複する可能性があります。定義の先頭に「#ifndef SMSTR_WAIT」「#define SMSTR_WAIT」を記述し、最後に「#endif」を記述してください。

```
#ifndef SMSTR_WAIT
#define SMSTR_WAIT
typedef enum
{
    BSP_DELAY_MICROSECS = 1000000,
    BSP_DELAY_MILLISECS = 1000,
    BSP_DELAY_SECS = 1
} bsp_delay_units_t;

#define BSP_ICLK_HZ  (120000000)     /* ICLK=120MHz */
#define BSP_PCLKA_HZ (120000000)     /* PCLKA=120MHz */
#endif      /* #ifndef SMSTR_WAIT */
```

3. API 関数

3.1 R_SCIFA_SMstr_Open()

SCIFA FIT モジュールの API を使用する際に、最初に使用する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Open(
    uint8_t channel,
    uint8_t br_data
)
```

Parameters

channel

SCIFA チャネル番号

br_data

SCIFA Bit Rate Register(BRR) 設定値

Return Values

SCIFA_SMSTR_SUCCESS	*/* 正常終了した場合 */*
SCIFA_SMSTR_ERR_PARAM	*/* パラメータ異常の場合 */*
SCIFA_SMSTR_ERR_OTHER	*/* 他タスクが SCIFA リソース取得済の場合 */*

Properties

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

Description

引数 channel で指定したチャネル番号の SCIFA レジスタを初期化します。
引数 br_data で指定した値を Bit Rate Register(BRR)に設定します。使用する MCU のユーザーズマニュアルハードウェア編を参考にして、動作環境に適した br_data を設定してください。
正常終了時には、SCIFA のモジュールストップ状態は解除され、SCK 端子、TXD 端子は汎用出力ポート H 出力状態、RXD 端子は汎用入力ポート状態になります。
また、引数 channel で指定したチャネル番号の SCIFA リソースを占有します。リソースを解放するためには R_SCIFA_SMstr_Close()をコールしてください。
通信中は本関数をコールしないでください。通信中にコールした場合の通信は保証しません。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;
uint8_t              br_data;

channel  = 8;
br_data  = 1;
ret = R_SCIFA_SMstr_Open(channel, br_data);
```

Special Notes:

本関数では GPIO と MPC の制御により、各端子の機能を汎用入出力ポートに設定します。本関数を呼び出す前に、各端子が他の周辺機能を使用していないか確認してください。

3.2 R_SCIFA_SMstr_Close()

使用中の SCIFA FIT モジュールのリソースを開放する際に使用する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Close(
    uint8_t channel
)
```

Parameters

channel

SCIFA チャネル番号

Return Values

SCIFA_SMSTR_SUCCESS	*/* 正常終了した場合 */*
SCIFA_SMSTR_ERR_PARAM	*/* パラメータ異常の場合 */*

Properties

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

Description

引数 channel で指定したチャネル番号の SCIFA をモジュールストップ状態に設定します。
正常終了時には、SCK 端子、TXD 端子は汎用出力ポート H 出力状態、RXD 端子は汎用入力ポート状態になります。
また、引数 channel で指定したチャネル番号の SCIFA リソースを解放します。再度通信を行う場合、R_SCIFA_SMstr_Open()をコールしてください。
通信中は本関数をコールしないでください。通信中にコールした場合の通信は保証しません。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;

channel = 8;
ret = R_SCIFA_SMstr_Close(channel);
```

Special Notes:

本関数では GPIO と MPC の制御により、各端子の機能を汎用入出力ポートに設定します。本関数を呼び出す前に、各端子が他の周辺機能を使用していないか確認してください。
本関数コール後の SCK 端子、TXD 端子はリセット後の状態（汎用入力ポート状態）とは異なります。必要に応じて、端子設定を見直してください。

3.3 R_SCIFA_SMstr_Control()

ビットレートを変更する際に使用する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Control(
    uint8_t channel,
    uint8_t br_data
)
```

Parameters

channel

SCIFA チャネル番号

br_data

Bit Rate Register(BRR) 設定値

Return Values

SCIFA_SMSTR_SUCCESS	*/* 正常終了した場合 */*
SCIFA_SMSTR_ERR_PARAM	*/* パラメータ異常の場合 */*

Properties

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

Description

引数 channel で指定したチャネル番号の SCIFA のビットレートを変更します。
引数 br_data で指定した値を Bit Rate Register(BRR)に設定します。使用する MCU のユーザーズマニュアル ハードウェア編を参考にして、動作環境に適した br_data を設定してください。
通信中は本関数をコールしないでください。通信中にコールした場合の通信は保証しません。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;
uint8_t              br_data;

channel  = 8;
br_data  = 1;
ret = R_SCIFA_SMstr_Open(channel, br_data);

br_data  = 3;
ret = R_SCIFA_SMstr_Control(channel, br_data);
```

Special Notes:

なし

3.4 R_SCIFA_SMstr_Write()

データ送信する際に使用する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Write(
    uint8_t channel,
    scifa_smstr_info_t * p_scifa_smstr_info
)
```

Parameters

channel

SCIFA チャネル番号

** p_scifa_smstr_info*

SCIFA 情報構造体

data_cnt

設定可能範囲は 1～4,294,967,295 です。0 を設定した場合、エラーを返します。また、DMAC 転送もしくは DTC 転送を指定する場合、設定値は 8 の倍数としてください。

*p_tx_data

送信データ格納バッファのアドレスを設定してください。DMAC 転送もしくは DTC 転送を指定する場合、バッファのアドレスは 4 バイトの境界値としてください。

*p_rx_data

未使用

tran_mode

転送モードを設定してください。ただし、DMAC 転送もしくは DTC 転送を指定する場合、別途 DMAC 転送もしくは DTC 転送のプログラムが必要です。

SCIFA_SMSTR_SW　：ソフトウェア転送
SCIFA_SMSTR_DMAC　：DMAC 転送
SCIFA_SMSTR_DTC　：DTC 転送

Return Values

SCIFA_SMSTR_SUCCESS　*/* 正常終了した場合 */*
SCIFA_SMSTR_ERR_PARAM　*/* パラメータ異常の場合 */*
SCIFA_SMSTR_ERR_HARD　*/* ハードウェア異常の場合 */*

Properties

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

Description

引数 channel で指定したチャネル番号の SCIFA を使って、データを送信します。
引数 tran_mode で DMAC 転送もしくは DTC 転送を指定した場合､転送可能なバイト数は 8 の倍数の値です。そうでない場合、エラー終了となり転送は行いません。

Reentrant
異なるチャネルからリエントラントは可能です。

Example
```
#define DATA_CNT (uint32_t)(4)

uint8_t               buf[DATA_CNT];
scifa_smstr_status_t  ret = SCIFA_SMSTR_SUCCESS;
uint8_t               channel;
scifa_smstr_info_t    tx_info;

channel = 8;
tx_info.data_cnt      = DATA_CNT;
tx_info.p_tx_data     = &buf[0];
tx_info.tran_mode     = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_Write(channel, &tx_info);
```

Special Notes:
DMAC 転送もしくは DTC 転送を指定する場合、以下の点にご注意ください。
・別途、DMAC FIT モジュール／DTC FIT モジュール／タイマモジュール（CMT FIT モジュール等）を入手してください。
・バッファのアドレスは 4 バイトの境界値としてください。
・転送データ数として 8 の倍数を指定しコールしてください。転送データ数として 1～7 の端数が発生する場合、別途ソフトウェア転送を指定しコールしてください。
・データ送信完了待ち処理 r_scifa_smstr_tx_dmacdtc_wait()はタイマを使用します。本関数をコールする前に CMT 等を用いて 1ms タイマを起動してください。そして、1ms 毎に R_SCIFA_SMstr_1ms_Interval()をコールしてください。
・本関数をコールする前に DMAC もしくは DTC を起動可能状態に設定してください。
・DMAC を起動可能状態に設定せず tran_mode に SCIFA_SMSTR_DMAC を設定して本関数をコールした場合、DMAC 転送は行われません。戻り値は SCIFA_SMSTR_ERR_HARD が返ります。
・DTC を起動可能状態に設定せず tran_mode に SCIFA_SMSTR_DTC を設定して本関数をコールした場合、DTC 転送は行われません。戻り値は SCIFA_SMSTR_ERR_HARD が返ります。

3.5 R_SCIFA_SMstr_Read()

データ受信する際に使用する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Read(
    uint8_t channel,
    scifa_smstr_info_t * p_scifa_smstr_info
)
```

Parameters

channel

SCIFA チャネル番号

** p_scifa_smstr_info*

SCIFA 情報構造体

data_cnt

設定可能範囲は 1～4,294,967,295 です。0 を設定した場合、エラーを返します。また、DMAC 転送もしくは DTC 転送を指定する場合、設定値は 8 の倍数としてください。

*p_tx_data

未使用

*p_rx_data

受信データ格納バッファのアドレスを設定してください。DMAC 転送もしくは DTC 転送を指定する場合、バッファのアドレスは 4 バイトの境界値としてください。

tran_mode

転送モードを設定してください。ただし、DMAC 転送もしくは DTC 転送を指定する場合、別途 DMAC 転送もしくは DTC 転送のプログラムが必要です。

SCIFA_SMSTR_SW　：ソフトウェア転送
SCIFA_SMSTR_DMAC　：DMAC 転送
SCIFA_SMSTR_DTC　：DTC 転送

Return Values

SCIFA_SMSTR_SUCCESS　　*/* 正常終了した場合 */*
SCIFA_SMSTR_ERR_PARAM　　*/* パラメータ異常の場合 */*
SCIFA_SMSTR_ERR_HARD　　*/* ハードウェア異常の場合 */*

Properties

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

Description

引数 channel で指定したチャネル番号の SCIFA を使って、データを受信します。
引数 tran_mode で DMAC 転送もしくは DTC 転送を指定した場合､転送可能なバイト数は 8 の倍数の値です。そうでない場合、エラー終了となり転送は行いません。

Reentrant
異なるチャネルからリエントラントは可能です。

Example
```
#define DATA_CNT (uint32_t)(4)

uint8_t               buf[DATA_CNT];
scifa_smstr_status_t  ret = SCIFA_SMSTR_SUCCESS;
uint8_t               channel;
scifa_smstr_info_t    rx_info;

channel = 8;
rx_info.data_cnt      = DATA_CNT;
rx_info.p_rx_data     = &buf[0];
rx_info.tran_mode     = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_Read(channel, &rx_info);
```

Special Notes:
DMAC 転送もしくは DTC 転送を指定する場合、以下の処理を追加してください。
・別途、DMAC FIT モジュール／DTC FIT モジュール／タイマモジュール（CMT FIT モジュール等）を入手してください。
・バッファのアドレスは 4 バイトの境界値としてください。
・転送データ数として 8 の倍数を指定しコールしてください。転送データ数として 1～7 の端数が発生する場合、別途ソフトウェア転送を指定しコールしてください。
・データ受信完了待ち処理 r_scifa_smstr_rx_dmacdtc_wait()はタイマを使用します。本関数をコールする前に CMT 等を用いて 1ms タイマを起動してください。そして、1ms 毎に R_SCIFA_SMstr_1ms_Interval()をコールしてください。
・本関数をコールする前に DMAC もしくは DTC を起動可能状態に設定してください。
・DMAC を起動可能状態に設定せず tran_mode に SCIFA_SMSTR_DMAC を設定して本関数をコールした場合、DMAC 転送は行われません。戻り値は SCIFA_SMSTR_ERR_HARD が返ります。
・DTC を起動可能状態に設定せず tran_mode に SCIFA_SMSTR_DTC を設定して本関数をコールした場合、DTC 転送は行われません。戻り値は SCIFA_SMSTR_ERR_HARD が返ります。

3.6 R_SCIFA_SMstr_WriteRead()

データ送受信（全二重通信）する際に使用する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_WriteRead(
    uint8_t channel,
    scifa_smstr_info_t * p_scifa_smstr_info
)
```

Parameters

channel

SCIFA チャネル番号

** p_scifa_smstr_info*

SCIFA 情報構造体

data_cnt

設定可能範囲は 1～4,294,967,295 です。0 を設定した場合、エラーを返します。また、DMAC 転送もしくは DTC 転送を指定する場合、設定値は 8 の倍数としてください。

*p_tx_data

送信データ格納バッファのアドレスを設定してください。DMAC 転送もしくは DTC 転送を指定する場合、バッファのアドレスは 4 バイトの境界値としてください。

*p_rx_data

受信データ格納バッファのアドレスを設定してください。DMAC 転送もしくは DTC 転送を指定する場合、バッファのアドレスは 4 バイトの境界値としてください。

tran_mode

転送モードを設定してください。送受信共に指定した転送モードが設定されます。ただし、DMAC 転送もしくは DTC 転送を指定する場合、別途 DMAC 転送もしくは DTC 転送のプログラムが必要です。

SCIFA_SMSTR_SW　：ソフトウェア転送.
SCIFA_SMSTR_DMAC　：DMAC 転送
SCIFA_SMSTR_DTC　：DTC 転送

Return Values

SCIFA_SMSTR_SUCCESS　*/* 正常終了した場合 */*
SCIFA_SMSTR_ERR_PARAM　*/* パラメータ異常の場合 */*
SCIFA_SMSTR_ERR_HARD　*/* ハードウェア異常の場合 */*

Properties

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

Description

引数 channel で指定したチャネル番号の SCIFA を使って、全二重によるデータを送受信します。
引数 tran_mode で DMAC 転送もしくは DTC 転送を指定した場合､転送可能なバイト数は 8 の倍数の値です。そうでない場合、エラー終了となり転送は行いません。

Reentrant
異なるチャネルからリエントラントは可能です。

Example
```
#define DATA_CNT (uint32_t)(4)

uint8_t               tx_buf[DATA_CNT];
uint8_t               rx_buf[DATA_CNT];
scifa_smstr_status_t  ret = SCIFA_SMSTR_SUCCESS;
uint8_t               channel;
scifa_smstr_info_t    trx_info;

channel = 8;
trx_info.data_cnt     = DATA_CNT;
trx_info.p_tx_data    = &tx_buf[0];
trx_info.p_rx_data    = &rx_buf[0];
trx_info.tran_mode    = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_WriteRead(channel, &trx_info);
```

Special Notes:
DMAC 転送もしくは DTC 転送を指定する場合、以下の処理を追加してください。
・別途、DMAC FIT モジュール／DTC FIT モジュール／タイマモジュール（CMT FIT モジュール等）を入手してください。
・バッファのアドレスは 4 バイトの境界値としてください。
・転送データ数として 8 の倍数を指定しコールしてください。転送データ数として 1～7 の端数が発生する場合、別途ソフトウェア転送を指定しコールしてください。
・データ受信完了待ち処理 r_scifa_smstr_rx_dmacdtc_wait()は、タイマを使用します。本関数をコールする前に CMT 等を用いて 1ms タイマを起動してください。そして、1ms 毎に R_SCIFA_SMstr_1ms_Interval() をコールしてください。
・DMAC を起動可能状態に設定せず tran_mode に SCIFA_SMSTR_DMAC を設定して本関数をコールした場合、DMAC 転送は行われません。戻り値は SCIFA_SMSTR_ERR_HARD が返ります。
・DTC を起動可能状態に設定せず tran_mode に SCIFA_SMSTR_DTC を設定して本関数をコールした場合、DTC 転送は行われません。戻り値は SCIFA_SMSTR_ERR_HARD が返ります。

3.7 R_SCIFA_SMstr_Get_BuffRegAddress()

トランスミット FIFO データレジスタ（FTDR）とレシーブ FIFO データレジスタ（FRDR）のアドレスを取得する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Get_BuffRegAddress(
    uint8_t channel,
    uint32_t * p_ftdr_adr,
    uint32_t * p_frdr_adr
)
```

Parameters

channel

SCIFA チャネル番号

** p_ftdr_adr*

FTDR のアドレス格納用ポインタ。格納先のアドレスを設定してください。

** p_frdr_adr*

FRDR のアドレス格納用ポインタ。格納先のアドレスを設定してください。

Return Values

SCIFA_SMSTR_SUCCESS	*/* 正常終了した場合 */*
SCIFA_SMSTR_ERR_PARAM	*/* パラメータ異常の場合 */*

Properties

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

Description

DMAC もしくは DTC の転送先／転送元のアドレスを設定する場合等にご使用ください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint32_t              reg_buff_ftdr;
uint32_t              reg_buff_frdr;
scifa_smstr_status_t  ret = SCIFA_SMSTR_SUCCESS;
uint8_t               channel;

channel = 8;
ret = R_SCIFA_SMstr_Get_BuffRegAddress(channel, &reg_buff_ftdr, &reg_buff_frdr);
```

Special Notes:

なし

3.8 R_SCIFA_SMstr_Int_Txif_Ier_Clear()

送信バッファエンプティ割り込み（TXIF）の ICU.IERm.IENj ビットをクリアします。

Format
void R_SCIFA_SMstr_Int_Txif_Ier_Clear(
    uint8_t channel
)

Parameters
channel
SCIFA チャネル番号

Return Values
なし

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

Description
DMAC の転送完了時に発生する TXIF 割り込みハンドラ内で、割り込みを禁止する時に使用してください。

Reentrant
異なるチャネルからリエントラントは可能です。

Example
```
DMA_Handler_W()
{
    R_SCIFA_SMstr_Int_Txif_Ier_Clear(8);
    R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes:
ソフトウェア転送や DTC 転送時には使用しないでください。転送を破壊する可能性があります。

3.9 R_SCIFA_SMstr_Int_Rxif_Ier_Clear()

受信バッファフル割り込み（RXIF）の ICU.IERm.IENj ビットをクリアします。

Format
```
void R_SCIFA_SMstr_Int_Rxif_Ier_Clear(
    uint8_t channel
)
```

Parameters
channel
SCIFA チャネル番号

Return Values
なし

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

Description
DMAC の転送完了時に発生する RXIF 割り込みハンドラ内で、割り込みを禁止する時に使用してください。

Reentrant
異なるチャネルからリエントラントは可能です。

Example
```
DMA_Handler_R()
{
    R_SCIFA_SMstr_Int_Rxif_Ier_Clear(8);
    R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes:
ソフトウェア転送や DTC 転送時には使用しないでください。転送を破壊する可能性があります。

3.10 R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set()

データ送信用 DMAC もしくは DTC 転送完了フラグを設定する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set(
    uint8_t channel,
    scifa_smstr_trans_flg_t flg
)
```

Parameters

channel

SCIFA チャネル番号

flg

フラグ。以下を設定してください。

SCIFA_SET_TRANS_STOP　：DMAC もしくは DTC 転送完了

（SCIFA_SET_TRANS_START　：DMAC もしくは DTC 転送開始・・・ユーザによる設定禁止）

Return Values

SCIFA_SMSTR_SUCCESS　　*/* 正常終了した場合 */*

SCIFA_SMSTR_ERR_PARAM　　*/* パラメータ異常の場合 */*

Properties

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

Description

DMAC の転送完了時に発生する TXIF 割り込みハンドラ内で、SCIFA_SET_TRANS_STOP をセットしてください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
DMA_Handler_W()
{
    R_SCIFA_SMstr_Int_Txif_Ier_Clear(8);
    R_SCIFA_SMstr_Int_Txif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes:

ソフトウェア転送や DTC 転送時には使用しないでください。転送を破壊する可能性があります。

3.11 R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set()

データ受信用 DMAC もしくは DTC 転送完了フラグを設定する関数です。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set(
    uint8_t channel,
    scifa_smstr_trans_flg_t flg
)
```

Parameters

channel

SCIFA チャネル番号

flg

フラグ。以下を設定してください。

SCIFA_SET_TRANS_STOP　：DMAC もしくは DTC 転送完了

（SCIFA_SET_TRANS_START　：DMAC もしくは DTC 転送開始・・・ユーザによるこの設定は禁止）

Return Values

SCIFA_SMSTR_SUCCESS　*/* 正常終了した場合 */*

SCIFA_SMSTR_ERR_PARAM　*/* パラメータ異常の場合 */*

Properties

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

Description

DMAC の転送完了時に発生する RXIF 割り込みハンドラ内で、SCIFA_SET_TRANS_STOP をセットしてください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
DMA_Handler_R()
{
    R_SCIFA_SMstr_Int_Rxif_Ier_Clear(8);
    R_SCIFA_SMstr_Int_Rxif_Dmacdtc_Flag_Set(8, SCIFA_SET_TRANS_STOP);
}
```

Special Notes:

ソフトウェア転送や DTC 転送時には使用しないでください。転送を破壊する可能性があります。

3.12 R_SCIFA_SMstr_GetVersion()

ドライバのバージョン情報を取得する際に使用する関数です。

Format
uint32_t R_SCIFA_SMstr_GetVersion(void)

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_scifa_smstr_rx_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。

Reentrant
異なるチャネルからリエントラントは可能です。

Example
```
uint32_t version;
version = R_SCIFA_SMstr_GetVersion();
```

Special Notes:
なし

3.13 R_SCIFA_SMstr_Set_LogHdlAddress()

LONGQ FIT モジュールのハンドラアドレスを設定する関数です。エラーログ取得処理を使用する場合、コールしてください。

Format

```
scifa_smstr_status_t R_SCIFA_SMstr_Set_LogHdlAddress(
    uint32_t user_long_que
)
```

Parameters

user_long_que
LONGQ FIT モジュールのハンドラアドレスを設定してください。

Return Values

SCIFA_SMSTR_SUCCESS */* 正常終了した場合 */*

Properties

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

Description

LONGQ FIT モジュールのハンドラアドレスを SCIFA FIT モジュールに設定します。
LONGQ FIT モジュールを使用し、エラーログを取得するための準備処理です。R_SCIFA_SMstr_Open()をコールする前に処理を実行してください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
#define ERR_LOG_SIZE (16)
#define SCIFA_USER_LONGQ_IGN_OVERFLOW    (1)

scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint32_t             MtlLogTbl[ERR_LOG_SIZE];
longq_err_t          err;
longq_hdl_t          p_SCIFA_user_long_que;
uint32_t             long_que_hndl_address;


/* Open LONGQ module. */
err = R_LONGQ_Open(&MtlLogTbl[0],
                   ERR_LOG_SIZE,
                   SCIFA_USER_LONGQ_IGN_OVERFLOW,
                   &p_SCIFA_user_long_que
);

long_que_hndl_address = (uint32_t)p_SCIFA_user_long_que;
ret = R_SCIFA_SMstr_Set_LogHdlAddress(long_que_hndl_address);
```

Special Notes:

別途 LONGQ FIT モジュールを組み込んでください。また、r_scifa_smstr_rx_config.h の#define SCIFA_SMSTR_CFG_LONGQ_ENABLE を有効にしてください。

3.14 R_SCIFA_SMstr_Log()

エラーログを取得する関数です。エラー発生時、ユーザ処理を終了する直前にコールしてください。

Format

```
uint32_t R_SCIFA_SMstr_Log(
    uint32_t flg,
    uint32_t fid,
    uint32_t line
)
```

Parameters

flg

0x00000001（固定値）を設定してください。

fid

0x0000003f（固定値）を設定してください。

line

0x0001ffff（固定値）を設定してください。

Return Values

```
0                                   /* 正常終了した場合 */
1                                   /* 異常終了した場合 */
```

Properties

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

Description

エラーログを取得する関数です。
LONGQ FIT モジュールを使用し、エラーログ取得終了処理を行います。エラー発生時、ユーザ処理を終了する直前にコールしてください。
エラーログのバッファサイズ設定方法については、LONGQ FIT モジュールを参照してください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
#define  USER_DRIVER_ID    (0x00000001)
#define  USER_LOG_MAX      (0x0000003f)
#define  USER_LOG_ADR_MAX  (0x00001fff)

uint8_t              buf[DATA_CNT];
scifa_smstr_status_t ret = SCIFA_SMSTR_SUCCESS;
uint8_t              channel;
scifa_smstr_info_t   tx_info;

channel = 8;
tx_info.data_cnt     = DATA_CNT;
tx_info.p_tx_data    = &buf[0];
tx_info.tran_mode    = SCIFA_SMSTR_SW;
ret = R_SCIFA_SMstr_Write(channel, &tx_info);

if (SCIFA_SMSTR_SUCCESS != ret)
{
   /* Set last error log to buffer. */
   R_SCIFA_SMstr_Log(
      USER_DRIVER_ID,
      USER_LOG_MAX,
      USER_LOG_ADR_MAX
   );

   R_SCIFA_SMstr_Close(channel);
}
```

Special Notes:

別途 LONGQ FIT モジュールを組み込んでください。また、r_scifa_smstr_rx_config.h の#define SCIFA_SMSTR_CFG_LONGQ_ENABLE を有効にしてください。

3.15 R_SCIFA_SMstr_1ms_Interval()

関数が呼ばれる毎に内部タイマカウンタをインクリメントします。

Format
void R_SCIFA_SMstr_1ms_Interval(void)

Parameters
なし

Return Values
なし

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

Description
DMAC もしくは DTC 転送完了待ち時に内部タイマカウンタをインクリメントします。

Reentrant
異なるチャネルからリエントラントは可能です。

Example
```
void r_cmt_callback (void * pdata)
{
    uint32_t channel;

    channel = (uint32_t)pdata;
    if (channel == gs_cmt_channel)
    {
        R_SCIFA_SMstr_1ms_Interval();
    }
}
```

Special Notes:
タイマ等を使用して本関数を 1ms 毎にコールしてください。
上記、Example は 1ms 毎に発生するコールバック関数で本関数をコールする例です。

4. 端子設定

Power on Reset 後、および API 関数実行後の端子の状態を示します。

本モジュールは「1.5.2 制御可能なスレーブデバイス」に示すとおり、SPI モード 3（CPOL=1、CPHA=1）をサポートします。ハードウェア構成によらず、**Power on Reset 後はユーザ側で GPIO 制御を行い、SCK 端子と TXD 端子を H 出力状態にしてください。**

また、R_SCIFA_SMstr_Close()後の SCK 端子と TXD 端子の状態は GPIO H 出力です。必要に応じて端子設定を見直してください。

表 4-1　関数実行後の端子の状態

関数名	SCK 端子（注１）	TXD 端子	RXD 端子（注２）
（Power on Reset 後）	GPIO 入力状態	GPIO 入力状態	GPIO 入力状態
R_SCIFA_SMstr_Open()前	GPIO H 出力状態 **ユーザ側で設定**	GPIO H 出力状態 **ユーザ側で設定**	GPIO 入力状態
R_SCIFA_SMstr_Open()後	GPIO H 出力状態 本モジュールで設定	GPIO H 出力状態 本モジュールで設定	GPIO 入力状態 本モジュールで設定
R_SCIFA_SMstr_Close()後	GPIO H 出力状態 本モジュールで設定	GPIO H 出力状態 本モジュールで設定	GPIO 入力状態 本モジュールで設定

注１：SCK 端子は、メモリカード接続の場合、外付け抵抗によるプルアップ処理が推奨されていません。そのため、Power on Reset 後、GPIO H 出力状態にしてください。

注２：RXD 端子は、外付け抵抗でプルアップ処理してください。「1.4.1 ハードウェア構成例」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
[e² studio] RX ファミリ C/C++コンパイラ CC-RX V.2.01.00 ユーザーズマニュアル RX ビルド編
(R20UT2747JJ0100)
[e² studio] RX ファミリ C/C++コンパイラ CC-RX V2.01.00 ユーザーズマニュアル RX コーディング編
(R20UT2748JJ0100)
[e² studio] RX ファミリ C/C++コンパイラ CC-RX V2.01.00 ユーザーズマニュアル メッセージ編
(R20UT2749JJ0100)
(最新版をルネサス エレクトロニクスホームページから入手してください。)

ホームページとサポート窓口

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

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

改訂記録	RX ファミリ　アプリケーションノート　SCIFA クロック同期式シングルマスタモジュール　Firmware Integration Technology

Rev.	発行日	改訂内容	
		ページ	ポイント
1.08	2014.12.26	—	初版発行
1.09	2016.09.30	7	1.3 関連アプリケーションノート ● DTC モジュール：元は RX Family DTC Module Using Firmware Integration Technology（R01AN1819EJ）であった。 ● RX ファミリ Serial Flash memory アクセス クロック同期式制御モジュール Firmware Integration Technology（R01AN2662JJ）を追加
		14	2.6 コンパイル時の設定 表 r_scifa_smstr_rx_pin_config.h #define 名：元は以下のとおりであった。 #define SCIFA_SCK_CHx_SMSTR_CFG_PORTNO #define SCIFA_SCK_CHx_SMSTR_CFG_BITNO #define SCIFA_TXD_CHx_SMSTR_CFG_PORTNO #define SCIFA_TXD_CHx_SMSTR_CFG_BITNO #define SCIFA_RXD_CHx_SMSTR_CFG_PORTNO #define SCIFA_RXD_CHx_SMSTR_CFG_BITNO
		16	2.9 モジュールの追加方法 最新の方法に記述を変更した。
		42	4.端子設定　元は 2.12 端子の状態であった。

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/