お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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。
5. 輸出に際しては、「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。本資料に記載されている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の目的で使用しないでください。また、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に使用することができません。
6. 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。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されてい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生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   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、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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。

SH7730 グループ

SCIF クロック同期式シリアル通信 送受信設定例 (全二重通信)

要旨

この資料は FIFO 内蔵シリアルコミュニケーションインタフェース (SCIF) 機能のクロック同期式モードの送受信設定例 (全二重通信) を掲載しています。

動作確認デバイス

SH7730

目次

1. はじめに

1.1 仕様

- シリアルクロック出力側 SH7730 とシリアルクロック入力側 SH7730 の 2 つのマイコン間で全二重通信を行います。
- シリアルクロック出力側 SH7730 は、同期用のクロックを出力し、そのクロックに同期してデータを送信すると同時に、そのクロックに同期して、シリアルクロック入力側 SH7730 からのデータを受信します。
- シリアルクロック入力側 SH7730 は、同期用のクロックを入力し、そのクロックに同期してデータを送信すると同時に、そのクロックに同期して、シリアルクロック出力側 SH7730 からのデータを受信します。
- シリアルクロック出力側 SH7730 とシリアルクロック入力側 SH7730 共に、SCIF チャネル 0 (SCIF0) を、クロック同期式モードの送受信モードとして初期化します。
- シリアルクロック入力側 SH7730 の送受信準備完了後、シリアルクロック出力側 SH7730 は、送受信動作を開始します。シリアルクロック入力側 SH7730 の送受信準備完了の判定は、ポート E の状態で判定します。PTE1 = Low、PTE2 = High の状態になった際に送受信準備完了とします。
- シリアルクロック入力側 SH7730 とシリアルクロック出力側 SH7730 共に、送受信データ数、送信データを指定して送受信処理を行います。指定データ数送受信完了後、送受信動作を終了します。

1.2 使用機能

- SCIF チャネル 0

1.3 適用条件

- 評価ボード: アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
  外付けメモリ (エリア 0): NOR 型フラッシュメモリ 4M バイト
  Spansion 製 S29AL032D70TFI04
  (エリア 3): SDR-SDRAM 32 M バイト (16 M バイト ×2 個)
  Samsung 製 K4S281632F-UC75
- マイコン: SH7730 (R8A77301)
- 動作周波数:
  CPU クロック: 266.66MHz
  SuperHyway バスクロック: 133.33MHz
  バスクロック: 66.66MHz
  周辺クロック: 33.33MHz
- エリア 0 バス幅: 16 ビット固定 (MD3 端子 = Low レベル)
- クロック動作モード: モード 2 (MD0 端子 = Low レベル, MD1 端子 = High レベル)
- エンディアン: ビッグエンディアン (MD5 端子 = Low レベル)
- ツールチェーン: ルネサス テクノロジ製 SuperH RISC engine Standard Toolchain Ver.9.2.0.0
- コンパイルオプション: High-performance Embedded Workshop でのデフォルト設定
  (-cpu=sh4a -include="$(PROJDIR)¥inc" -object="$(CONFIGDIR)¥$(FILELEAF).obj"
  -debug -optimize=0 -gbr=auto -chgincpath -errorpath -global_volatile=0
  -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo)

1.4 関連アプリケーションノート

本資料の参考プログラムは, 「SH7730 グループ アプリケーションノート SH7730 初期設定例 (RJJ06B0864)」の設定条件で動作確認しています。そちらもあわせてご参照ください。

2. 応用例の説明

本応用例では、FIFO 内蔵シリアルコミュニケーションインタフェース (SCIF) を使用します。SCIF0 をクロック同期式モードの送受信モードで初期化し、全二重通信を行います。

シリアルクロック出力側 SH7730 とシリアルクロック入力側 SH7730 の 2 つのマイコン間で全二重通信を行います。

【注】 本応用例では、シリアルクロック出力側 SH7730 **用のプログラムとシリアルクロック入力**側 SH7730 **用のプログラムをそれぞれ用意しています。シリアルクロック出力側** SH7730 **制御時には、**#ifdef SERIAL_CLOCK_OUT_MCU **対応プログラムを、シリアルクロック入力側** SH7730 **制御時には、**#ifdef SERIAL_CLOCK_IN_MCU **対応プログラムをそれぞれ有効にします。**

2.1 使用機能の動作概要

SCIF のクロック同期式モードでは、クロックパルスに同期してデータ送受信を行います。クロックソースとして内部クロックまたは、SCK 端子より外部クロック入力の選択ができます。内部クロックを選択した場合は、同期クロックを SCK 端子から出力します。外部クロックを選択した場合は、同期クロックを SCK 端子から入力します。通信データフォーマットは、8 ビット長固定です。表 1にクロック同期式モードの概要を示します。SCIF ブロック図については、「SH7730 グループ ハードウェアマニュアル (RJJ09B0339) 22 章 図 22.1 SCIF のブロック図」を参照ください。

表1 SCIF (クロック同期式モード) の概要

項目	概要
チャネル数	4 チャネル（SCIF0～SCIF3）
クロックソース	内部クロック: Pϕ, Pϕ/4, Pϕ/16, Pϕ/64　Pϕ: 内蔵周辺クロック 外部クロック: SCK0～SCK3 端子入力クロック
データフォーマット	転送データ長: 8 ビット固定 転送順序: LSB ファースト固定
エラー検出	オーバランエラー
割り込み要求	送信 FIFO データエンプティ割り込み (TXI) 受信 FIFO データフル割り込み (RXI) ブレーク割り込み (オーバランエラー) (BRI)
その他	• 独立した送信部と受信部を備えているので、送信と受信を同時に行うことが可能 • 送信部および受信部ともに 16 段の FIFO バッファ構造になっているのでシリアルデータの連続送信、連続受信が可能 • 消費電力低減のために、未使用チャネルのクロック供給を停止させることが可能 • 送信および受信 FIFO データレジスタ内に格納されている有効データ数を検出可能

【注】 SCIF **についての詳細は、「**SH7730 **グループ ハードウェアマニュアル** (RJJ09B0339) 22 **章** FIFO **内蔵シリアルコミュニケーションインタフェース** (SCIF)**」の章を参照ください。**

2.2 クロック同期式モードの動作

SCIF のクロック同期式モードの動作については、「SH7730 グループ ハードウェアマニュアル (RJJ09B0339) 22 章 FIFO 内蔵シリアルコミュニケーションインタフェース (SCIF) 22.4.3 クロック同期式モードの動作」の章を参照ください。

2.3 全二重通信の動作概要

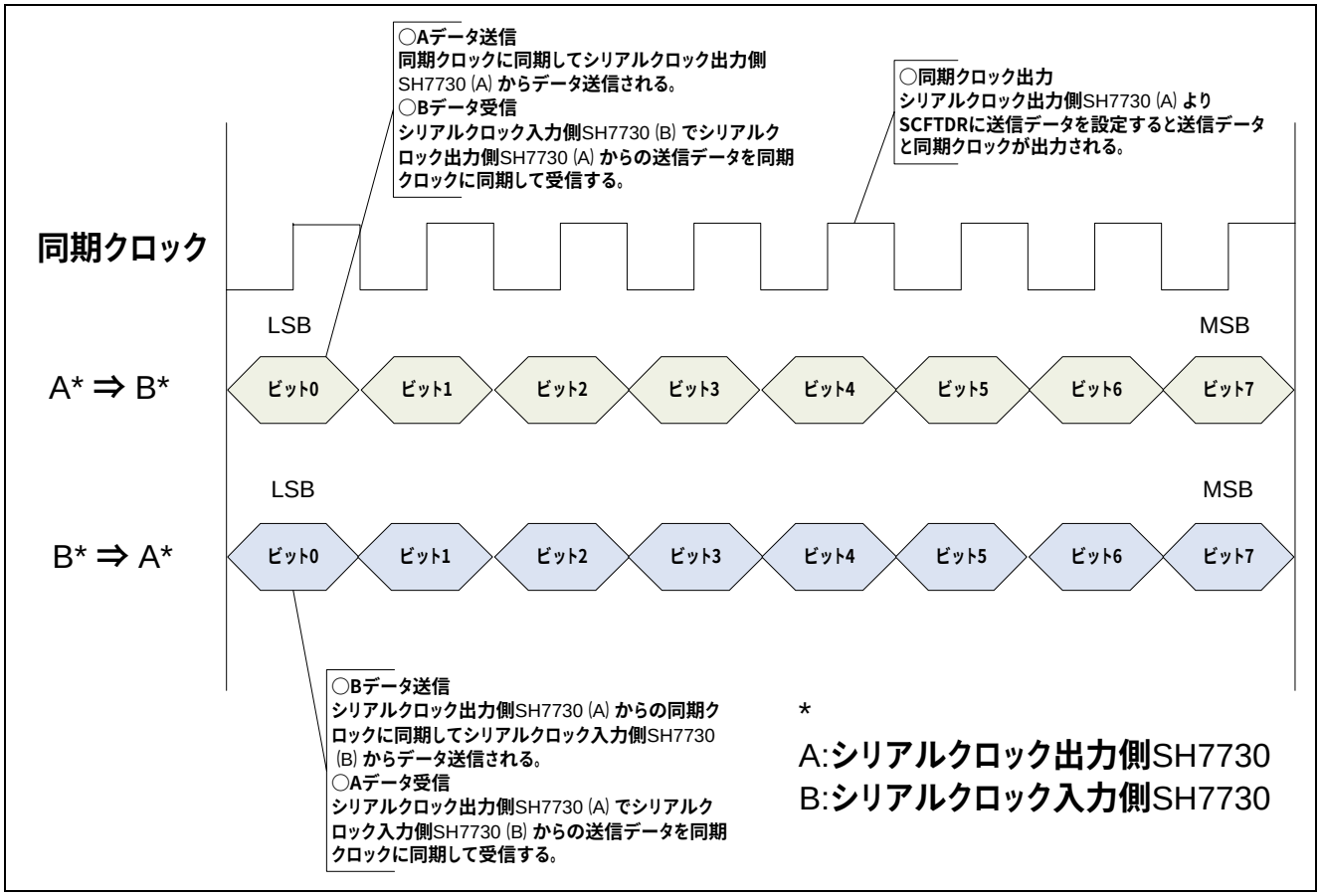


図1 全二重通信の動作概要

3. 参考プログラムについて

3.1 参考プログラム動作環境

以下のように、シリアルクロック出力側 SH7730 とシリアルクロック入力側 SH7730 を接続します。

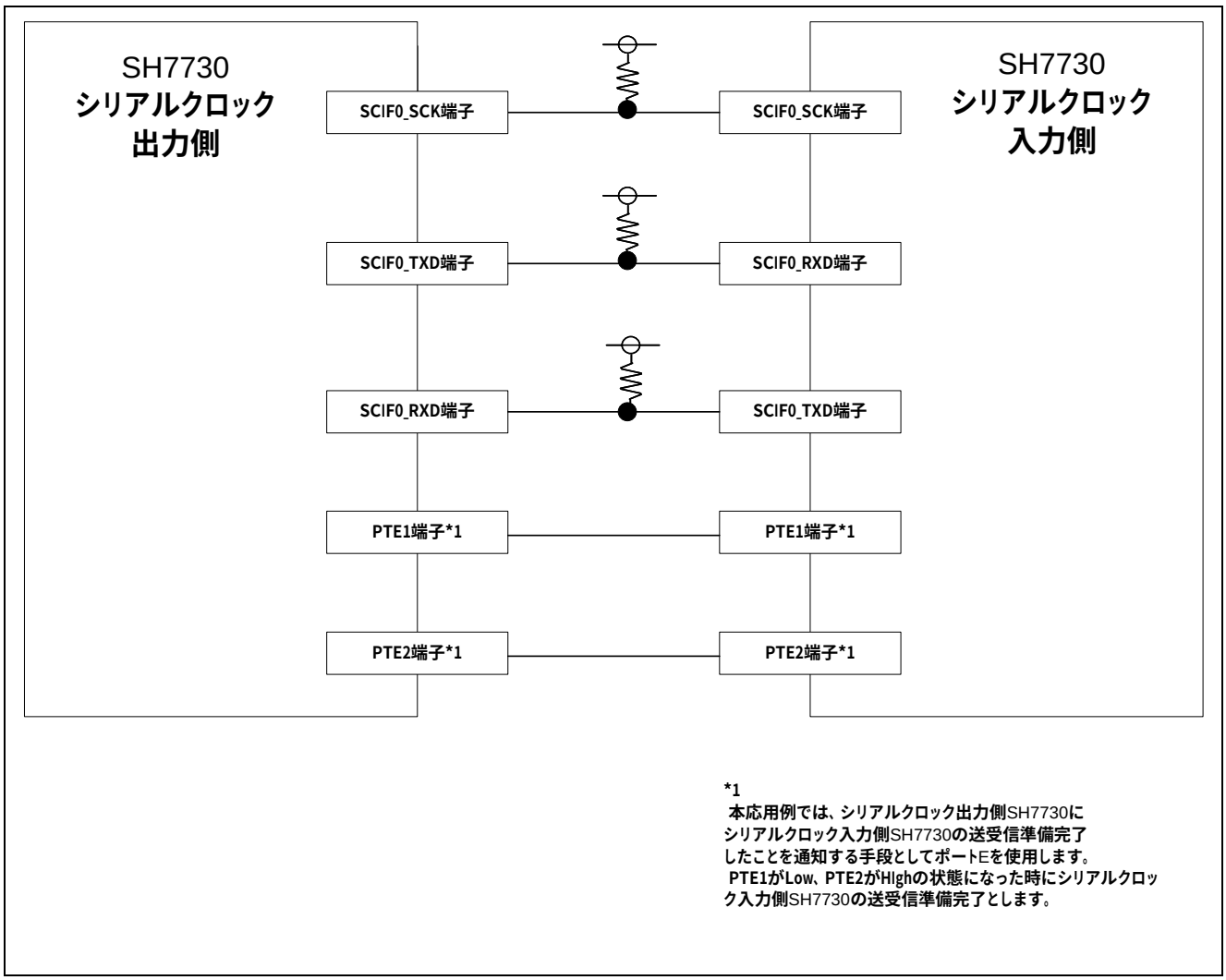


図2 参考プログラム動作環境

3.2 参考プログラムの動作概要

3.2.1 シリアルクロック出力側 SH7730

参考プログラムでは、主に以下のことを行います。

(1) シリアルクロック入力側 SH7730 が送受信準備完了まで待ちます。シリアルクロック入力側 SH7730 が送受信準備完了後、以下の処理を開始します。(PTE1 = Low、PTE2 = High の状態が、シリアルクロック入力側 SH7730 が送受信準備完了とします。)
(2) シリアルクロック入力側 SH7730 が送受信準備完了後、SCIF0 をクロック同期式の送受信モード、同期クロック出力として初期化します。
(3) 送信データを 0x00～0x23 (36 バイト) とします。
(4) SCIF0_SCK 端子から同期クロックの出力を行い、データ送受信を行います。
(5) 送信 FIFO (SCFTDR0) の空き数分、送信データを設定します。残りの送信データについては、送信 FIFO データエンプティ割り込み (TXI) 要求のタイミングで、送信 FIFO (SCFTDR0) に順次設定します。
(6) 同期クロックに同期して、シリアルクロック入力側 SH7730 から送信されるデータ*1 を順次受信し、送信 FIFO データエンプティ割り込み（TXI）要求のタイミング*2 で受信 FIFO に格納されているデータを受信データ格納用のデータ領域 (g_rcv_data[D_SCIF_DATA_SIZE]) に格納します。
(7) 指定した全送信データの送信が完了するまで待ちます。
同期クロックで送受信が行われるので送信が完了した時点で受信も完了しています。
(8) 受信 FIFO に残っている受信データを受信データ格納用のデータ領域 (g_rcv_data[D_SCIF_DATA_SIZE]) に格納します。
(9) 受信データ格納用のデータ領域 (g_rcv_data[D_SCIF_DATA_SIZE]) にシリアルクロック入力側 SH7730 から送信されるデータ*1 が格納されていることを確認します。

【注】 *1 **シリアルクロック入力側** SH7730 **から送信されるデータは、**0x23～0x00 (36 **バイト**) **とします。**
*2 **同期クロックによる送受信処理により、動作時の送信データ数と受信データ数は一致するため、本応用例では、送信** FIFO **データエンプティ割り込み** (TXI) **要求のタイミングで、受信** FIFO **に格納されているデータを取得します。**

表2 シリアルクロック出力側 SH7730 **の通信機能設定**

通信フォーマット	設定機能
通信モード	**クロック同期式**
使用チャネル	**チャネル** 0
割り込み	**送信** FIFO **データエンプティ割り込み** (TXI) **使用** ※SCSCR0.TIE=1 **設定** **優先度** 1
通信速度	100Kbps
データ長	8 **ビットデータ**
ビット順序	LSB **ファースト**
同期クロック	**同期クロック出力**
FIFO **データ数トリガ**	**レシーブ** FIFO **データ数トリガ**: 8 **※本参考プログラムでは、**#define **値** (D_SCIF_DATA_NUM_RCV_TRIGGER) **の変更により変更可能です。** **トランスミット** FIFO **データ数トリガ**: 8 **※本参考プログラムでは、**#define **値** (D_SCIF_DATA_NUM_SND_TRIGGER) **の変更により変更可能です。**

【注】 #ifdef SERIAL_CLOCK_OUT_MCU **に対応するプログラムが、シリアルクロック出力側** SH7730 **の制御プログラムとなります。**SERIAL_CLOCK_OUT_MCU **を有効にしてご使用ください。**

3.2.2 シリアルクロック入力側 SH7730

参考プログラムでは、主に以下のことを行います。

(1) SCIF0 をクロック同期式の送受信モード、同期クロック入力として初期化します。
(2) 送信データを 0x23～0x00 (36 バイト) とします。
(3) 送信 FIFO (SCFTDR0）の空き数分、送信データを設定します。
(4) シリアルクロック出力側 SH7730 に送受信準備完了したことを通知します。(PTE1 = Low、PTE2 = High にします。)
(5) SCIF0_SCK 端子で、シリアルクロック出力側 SH7730 からの同期クロックを入力し、データ送受信を行います。
(6) 同期クロックが入力された際に、同期クロックに同期して送信データが送信されます。
(7) 残りの送信データについては、送信 FIFO データエンプティ割り込み (TXI) 要求のタイミングで、送信 FIFO (SCFTDR0) に順次設定します。
(8) また、同期クロックに同期して、シリアルクロック出力側 SH7730 から送信されるデータ*1 を順次受信し、送信 FIFO データエンプティ割り込み (TXI) 要求のタイミング*2 で受信 FIFO に格納されているデータを受信データ格納用のデータ領域 (g_rcv_data[D_SCIF_DATA_SIZE]) に格納します。
(9) 指定した全送信データの送信が完了するまで待ちます。
同期クロックで送受信が行われるので送信が完了した時点で受信も完了しています。
(10) 受信 FIFO に残っている受信データを受信データ格納用のデータ領域 (g_rcv_data[D_SCIF_DATA_SIZE]) に格納します。
(11) 受信データ格納用のデータ領域 (g_rcv_data[D_SCIF_DATA_SIZE]) にシリアルクロック出力側 SH7730 から送信されるデータ*1 が格納されていることを確認します。

【注】 *1 **シリアルクロック出力側** SH7730 **から送信されるデータは、**0x00～0x23 (36 **バイト**) **とします。**
*2 **同期クロックによる送受信処理により、動作時の送信データ数と受信データ数は一致するため、本応用例では、送信** FIFO **データエンプティ割り込み** (TXI) **要求のタイミングで、受信** FIFO **に格納されているデータを取得します。**

表3　シリアルクロック入力側 SH7730 **の通信機能設定**

通信フォーマット	設定機能
通信モード	**クロック同期式**
使用チャネル	**チャネル** 0
割り込み	**送信** FIFO **データエンプティ割り込み** (TXI) **使用**　※SCSCR0.TIE=1 **設定** **優先度**　1
通信速度	100Kbps
データ長	8 **ビットデータ**
ビット順序	LSB **ファースト**
同期クロック	**同期クロック入力**
FIFO **データ数トリガ**	**レシーブ** FIFO **データ数トリガ**: 8 **※本参考プログラムでは、**#define **値** (D_SCIF_DATA_NUM_RCV_TRIGGER) **の変更により変更可能です。** **トランスミット** FIFO **データ数トリガ**: 8 **※本参考プログラムでは、**#define **値** (D_SCIF_DATA_NUM_SND_TRIGGER) **の変更により変更可能です。**

【注】 #ifdef SERIAL_CLOCK_IN_MCU **に対応するプログラムが、シリアルクロック入力側** SH7730 **の制御プログラムとなります。**SERIAL_CLOCK_IN_MCU **を有効にしてご使用ください。**

3.3 参考プログラムの処理フロー

ここでは、本参考プログラムの処理フローを説明します。

3.3.1 参考プログラムメイン処理

(1) シリアルクロック出力側 SH7730 制御プログラム

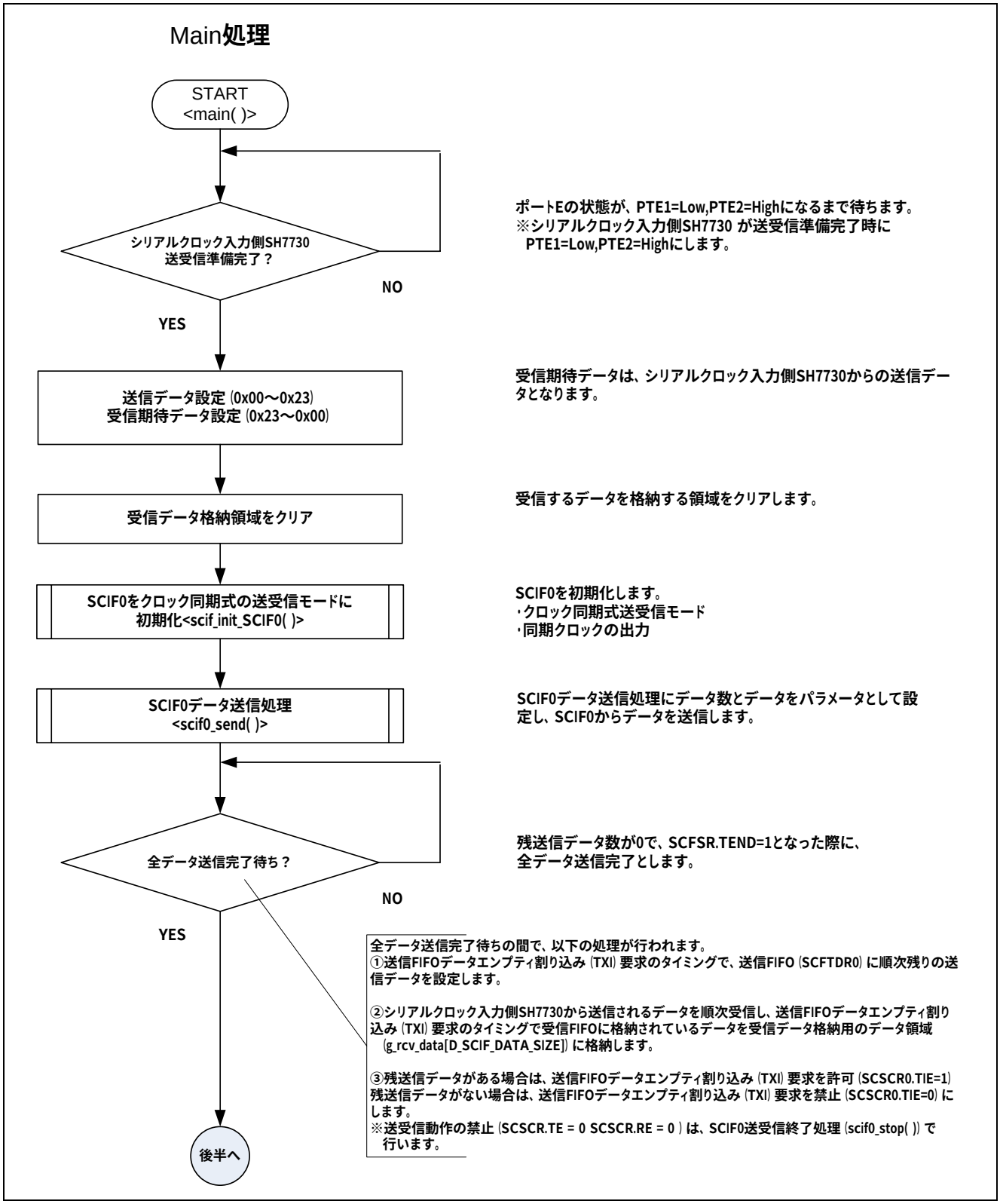


図3 シリアルクロック出力側 Main 処理フロー (前半)

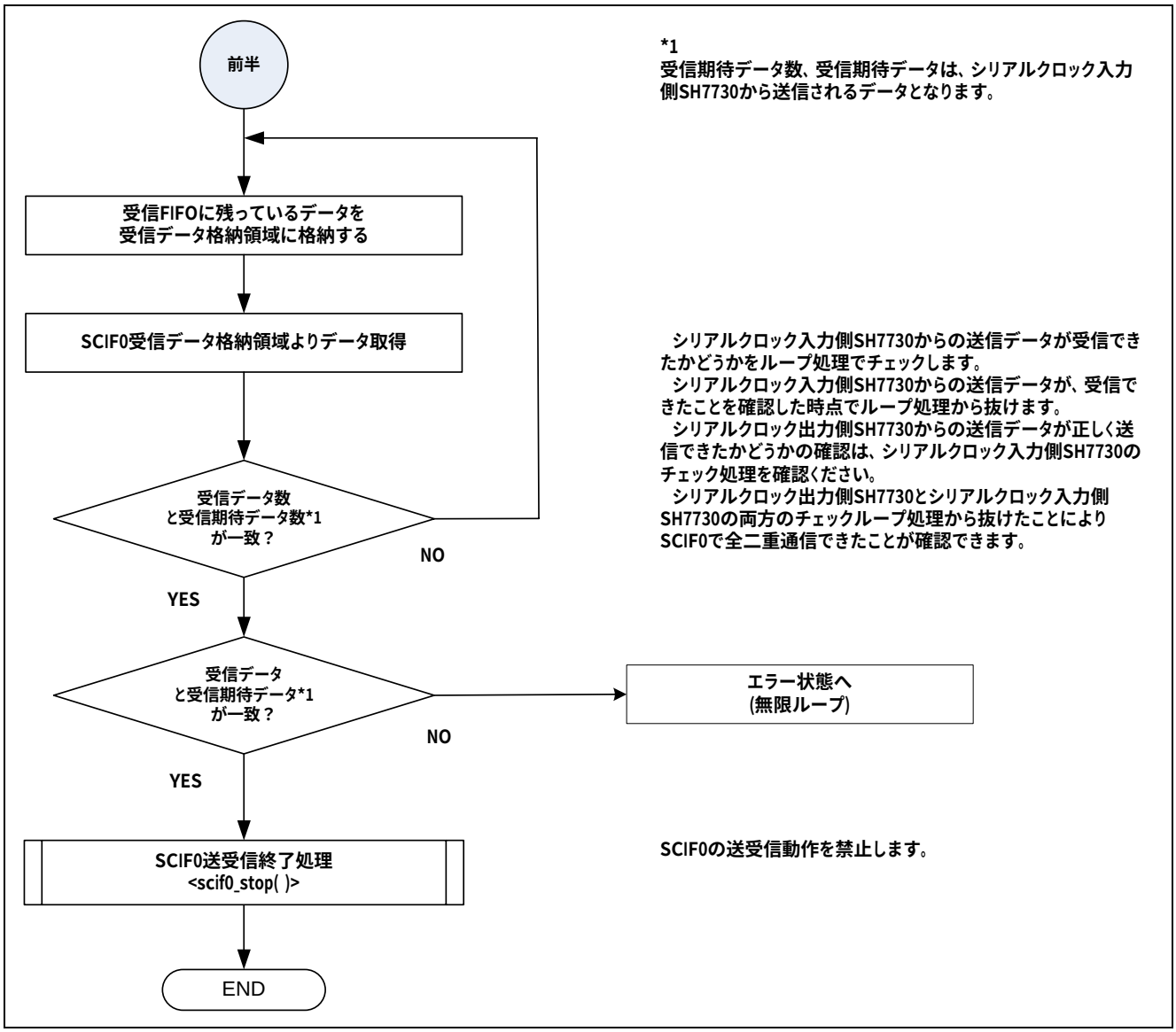


図4　シリアルクロック出力側 Main 処理フロー (後半)

(2) シリアルクロック入力側 SH7730 制御プログラム

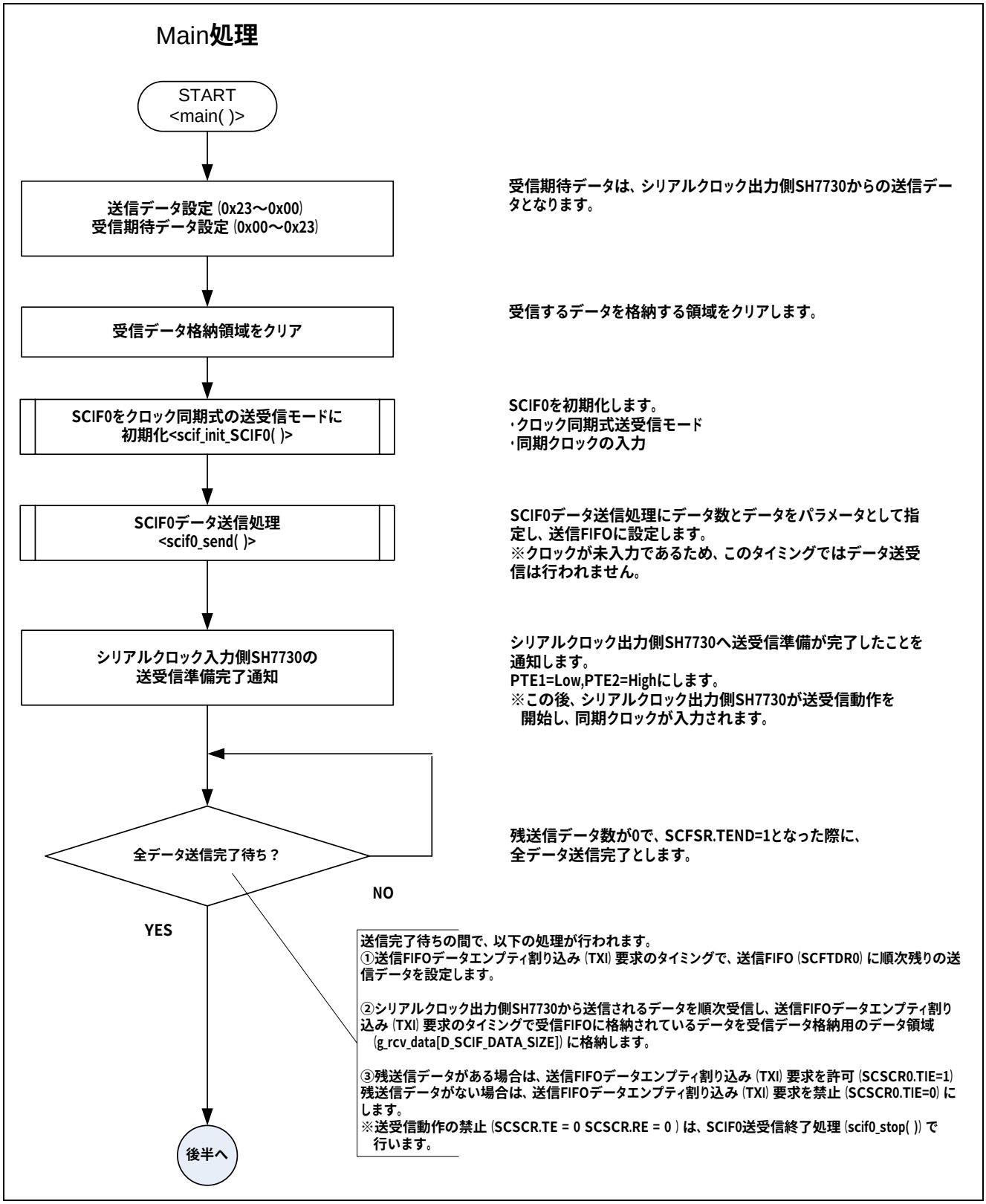


図5　シリアルクロック入力側 Main 処理フロー (前半)

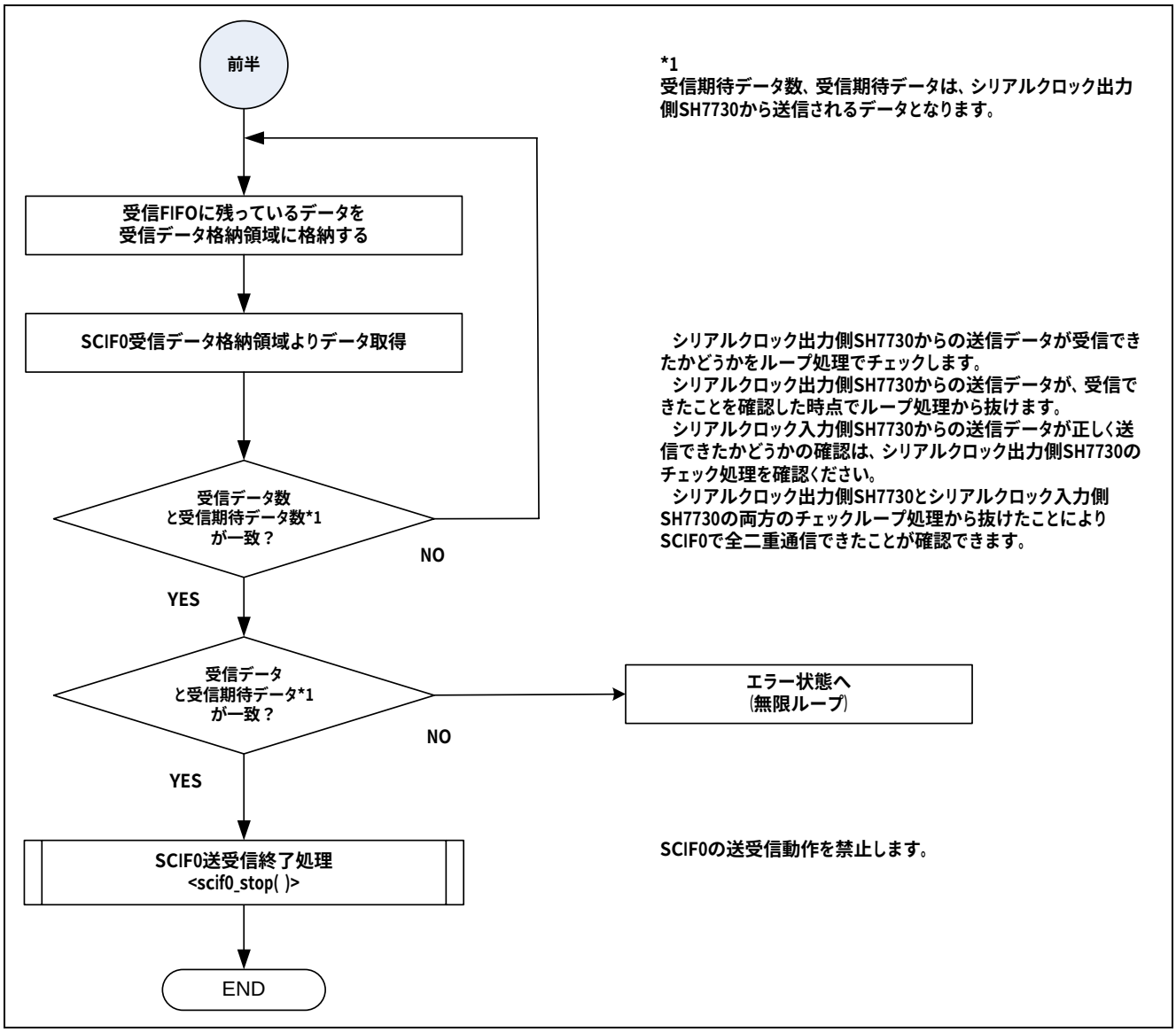


図6　シリアルクロック入力側 Main 処理フロー (後半)

3.3.2 SCIF0 初期化処理フロー

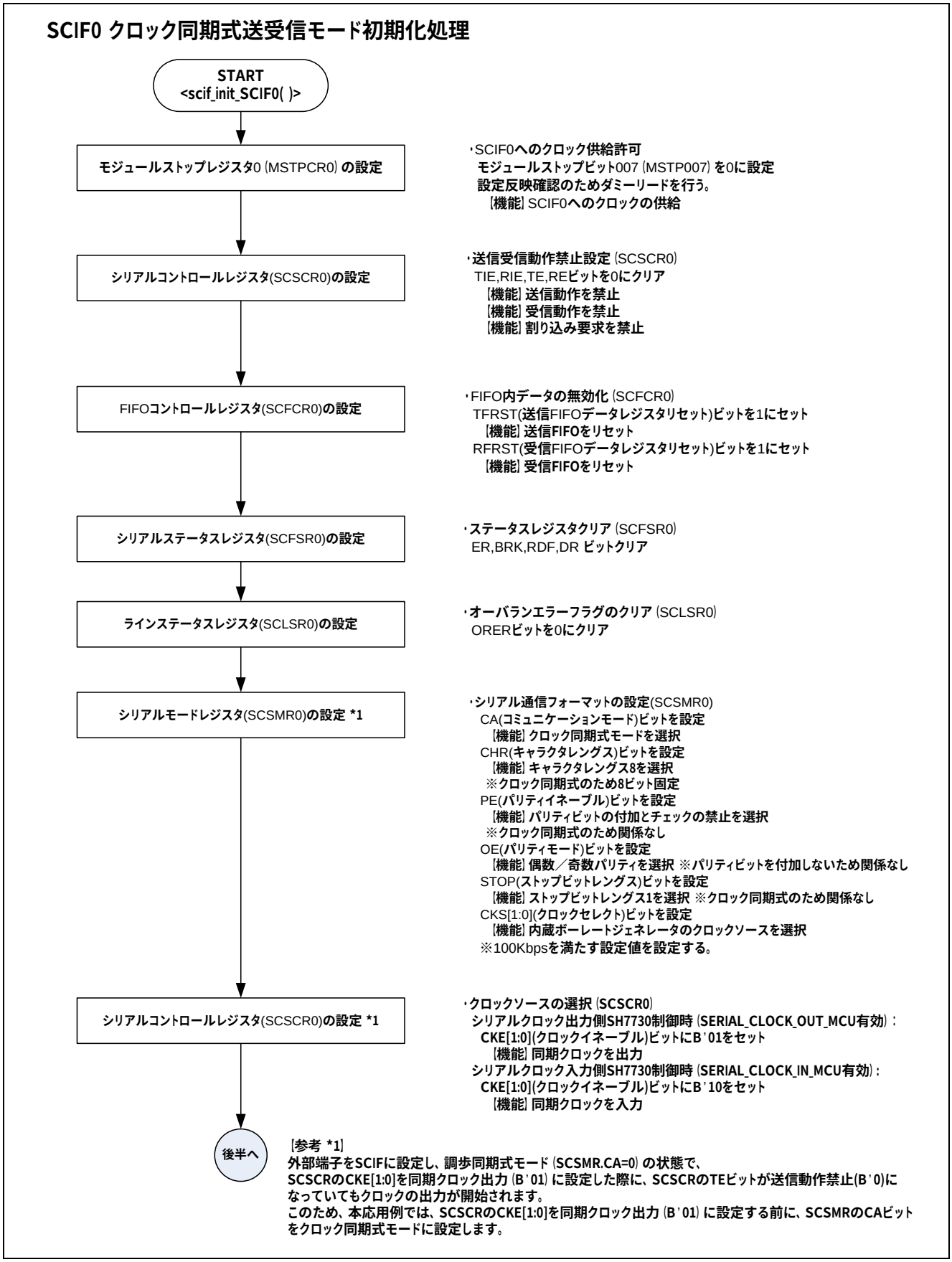
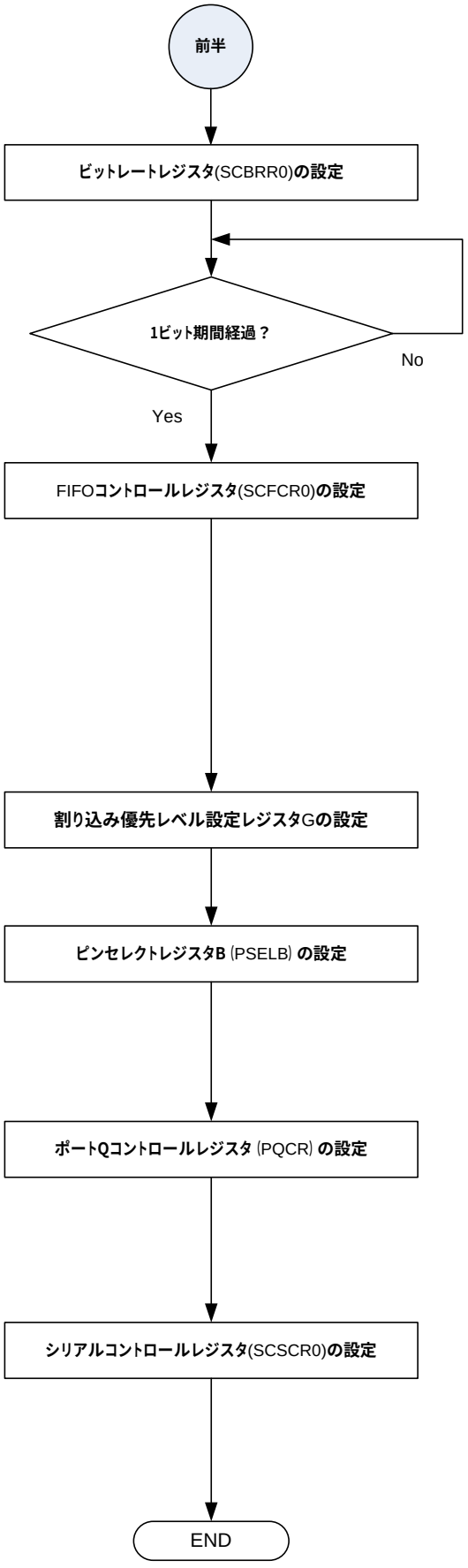


図7 SCIF0 初期化処理フロー (前半)



・通信速度の設定 (SCBRR0)
※100Kbpsを満たす設定値を設定する。

・1ビット期間経過待ち
SCBRR0設定後、SCSCR0のTEビットを設定する前に
少なくとも1ビット待ちます。

RTRG[1:0](レシーブFIFOデータ数トリガ)
【機能】受信トリガ数=8(B'10)を選択
TTRG[1:0](トランスミットFIFOデータ数トリガ)
【機能】送信トリガ数=8(B'00)を選択
MCE(モデムコントロールイネーブル)ビットを設定
【機能】モデムコントロール信号(CTS,RTS)無効を選択
TFRST(トランスミットFIFOデータレジスタリセット)ビットを0にクリア
【機能】リセット動作を禁止
RFRST(レシーブFIFOデータレジスタリセット)ビットを0にクリア
【機能】リセット動作を禁止
LOOP(ループバックテスト)ビットを0に設定
【機能】ループバックテストを禁止

IPRG.IPR0nの設定
【機能】SCIF0の割り込み優先レベルを1に設定

・マルチプレクス端子の機能をシリアル通信機能端子設定
PSB7ビットを0にセット
【機能】SCIF0_RXD端子選択
PSB8ビットを0にセット
【機能】SCIF0_TXD端子選択

・マルチプレクス端子の機能をシリアル通信機能端子設定
PQ0MDビットをB'00にセット
【機能】その他の機能選択
PQ1MDビットをB'00にセット
【機能】その他の機能選択
PQ2MDビットをB'00にセット
【機能】その他の機能選択

・送信動作許可設定 (SCSCR0)
RE(レシーブイネーブル)ビットを1にセット
【機能】受信動作を許可
TE(トランスミットイネーブル)ビットを1にセット
【機能】送信動作を許可

※TEビットとREビットを同時に1に設定してください。
※TIE (トランスミットインタラプトイネーブル)ビットはSCFTDRに
送信データを設定時に残送信データがあれば有効にします。

図8　SCIF0 初期化処理フロー (後半)

3.3.3 SCIF0 データ送信処理フロー

データ送信処理を開始する際に、パラメータに送信データ数、送信データを設定し SCIF0 データ送信処理をコールします (図 9)。

SCIF0 データ送信処理 (scif0_send( )) で、SCIF0 送信データ設定処理 (scif0_setdata( )) をコールします。

SCIF0 送信データ設定処理 (scif0_setdata( )) で、送信データが存在する場合に、SCFTDR0 データ設定処理 (scif_setdata_to_SCFTDR_ch0( )) をコールします (図 10)。SCFTDR0 データ設定処理 (scif_setdata_to_SCFTDR_ch0( )) で、送信 FIFO の空き数分、送信データを SCFTDR0 に設定します。この時に、送信データを SCFTDR0 に設定後、残送信データがある場合、送信 FIFO データエンプティ割り込み (TXI) 要求を許可にし、残送信データがない場合、送信 FIFO データエンプティ割り込み (TXI) 要求を禁止にします（図 11)。

送信 FIFO データエンプティ割り込み (TXI) 発生時にコールされる SCIF0 割り込み処理で、SCIF0 送信データ設定処理 (scif0_setdata( )) をコールし、残送信データを順次 SCFTDR0 に設定します (図 12)。

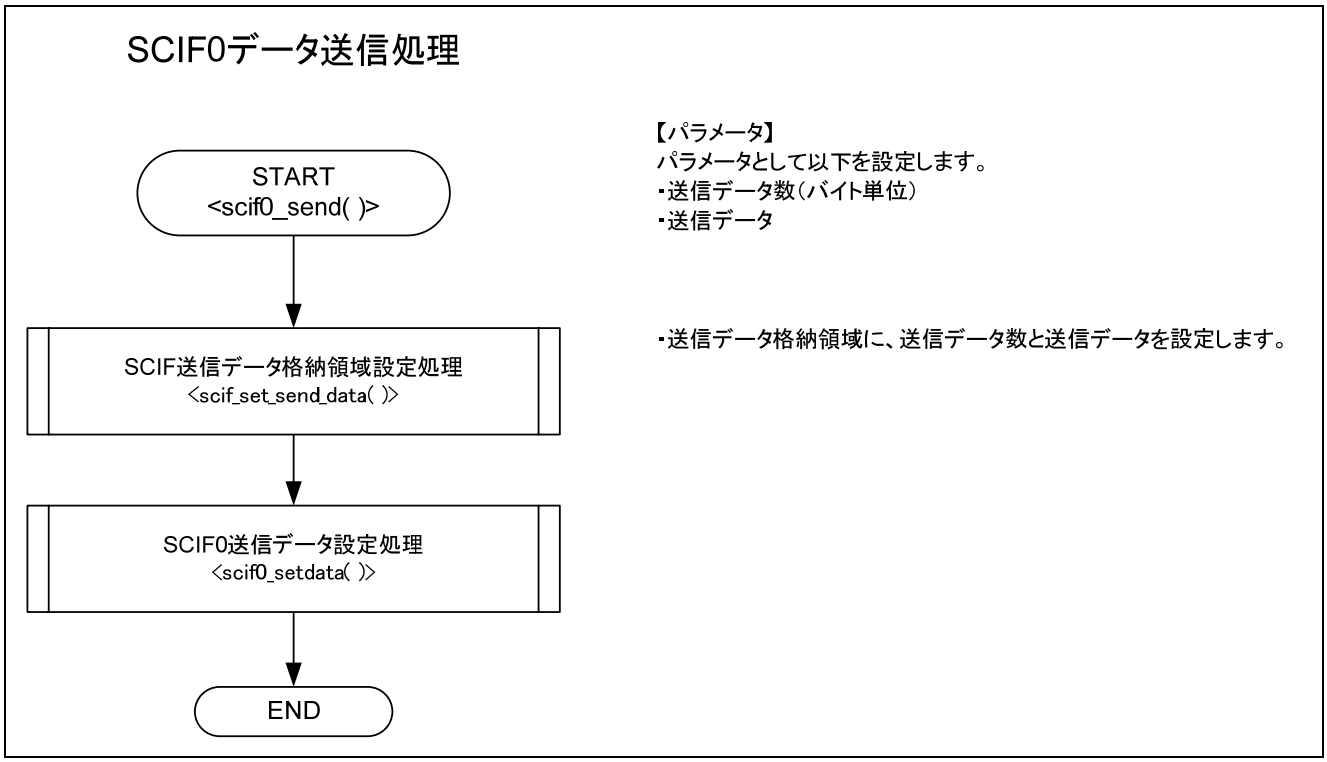


図9　SCIF0 データ送信処理フロー

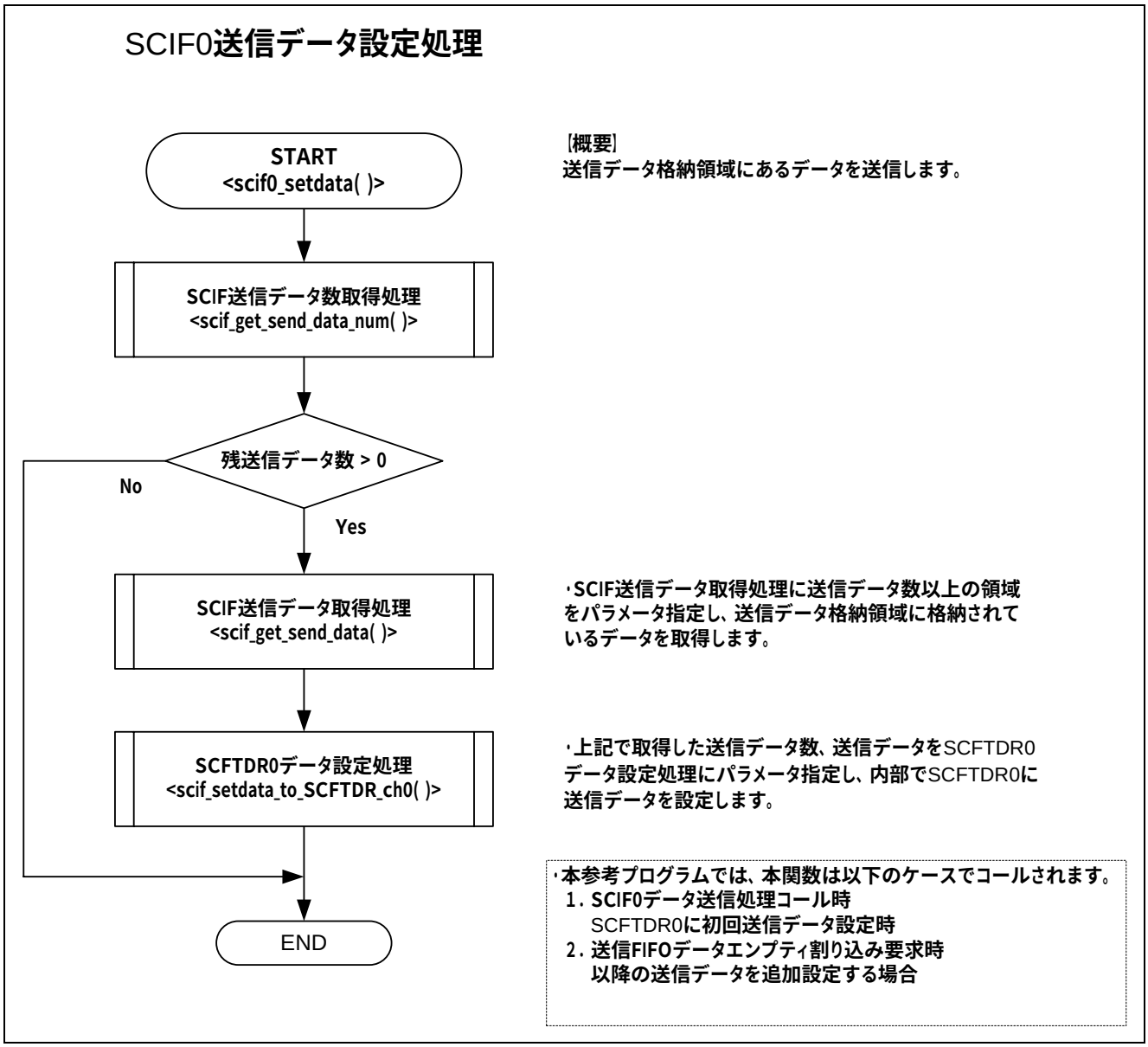


図10　SCIF0 送信データ設定処理フロー

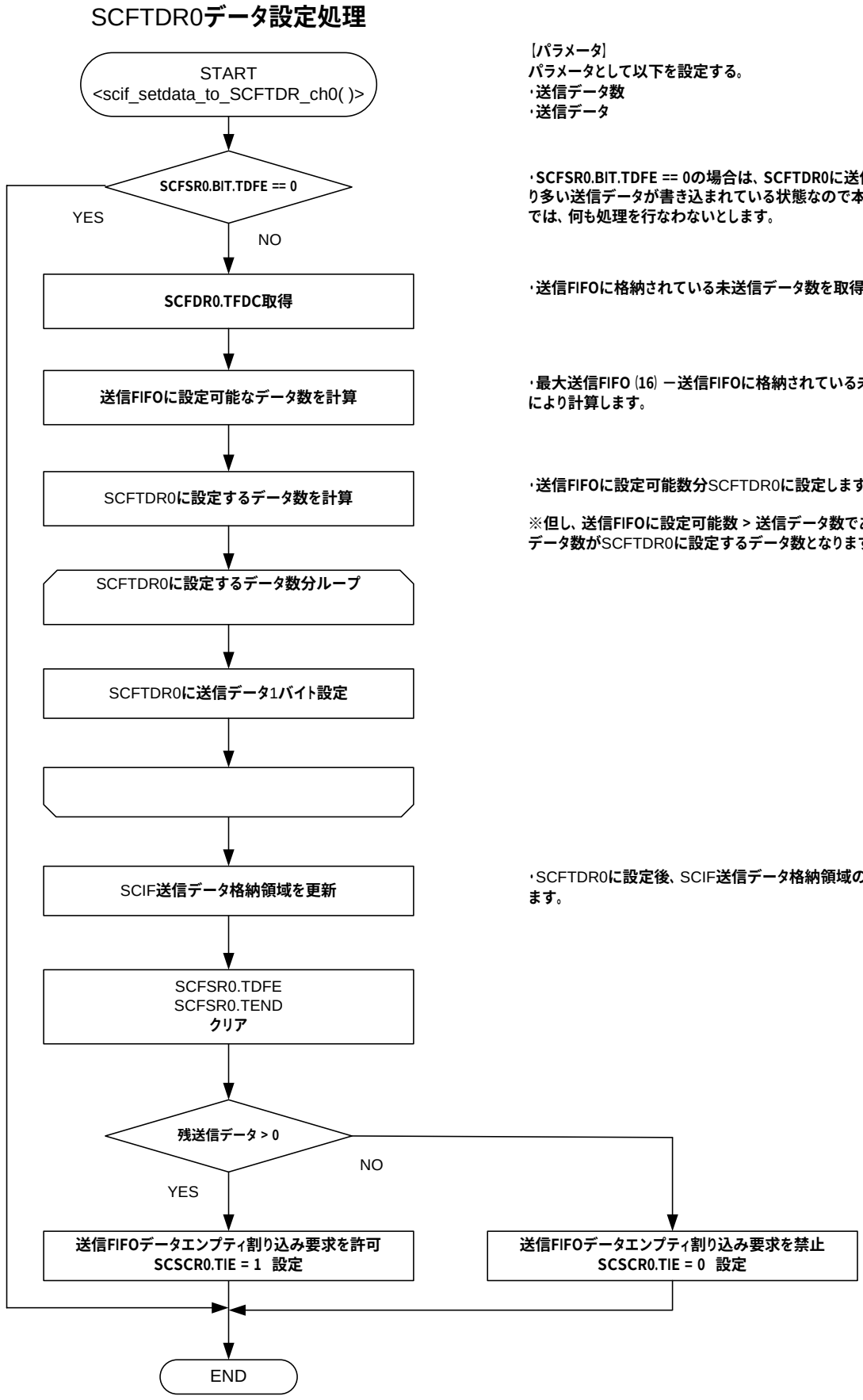
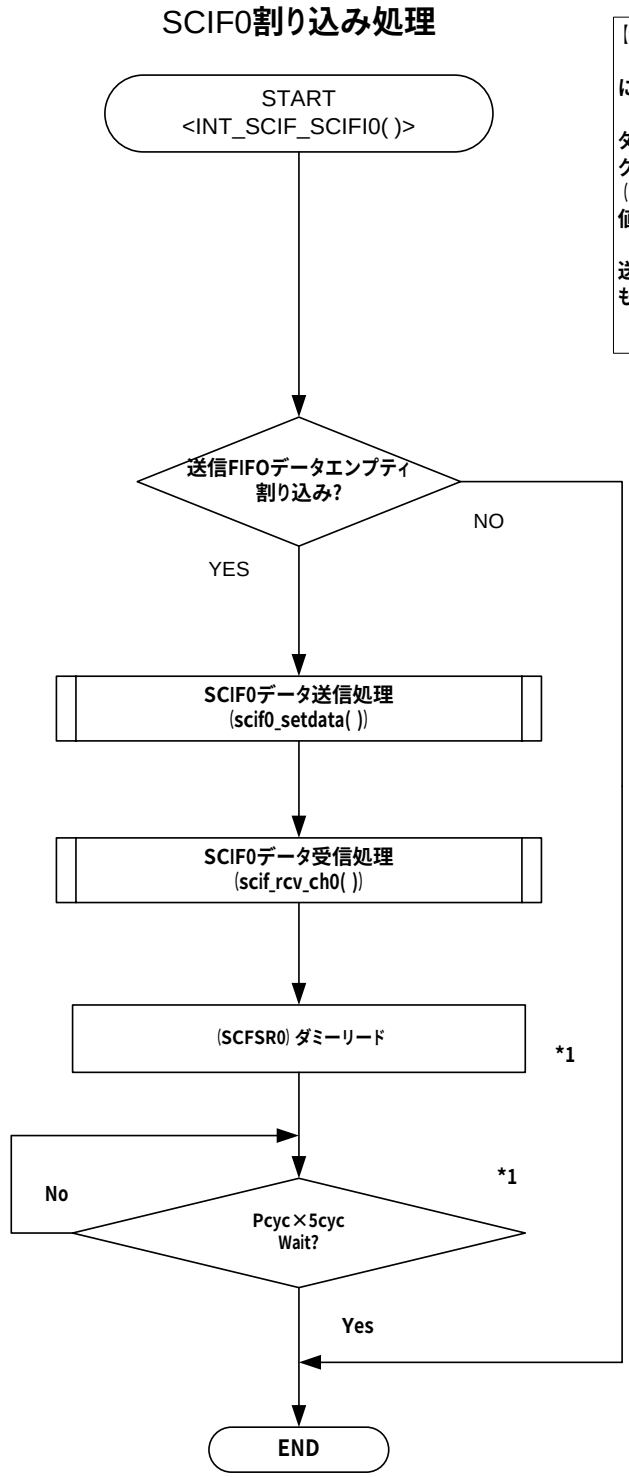


図11　SCFTDR0 データ設定処理フロー



[注]
シリアルクロック入力側では、同期クロック入力時に送信FIFO (SCFTDR) に送信データが設定されているようにしてください。
送信FIFOデータエンプティ割り込みで、送信FIFO (SCFTDR) に送信データを設定するタイミングが遅れて、送信FIFOが空の状態であると、シリアルクロック出力側にデータが送出されません。
(同期クロックに同期して受信する、シリアルクロック出力側にとっては不定値となります。)
また、シリアルクロック入力側で、同期クロックに同期して送信FIFOにある送信データを送信しないと、シリアルクロック出力側からのデータを受信しても受信FIFOにデータは格納されません。

同期クロックによる送受信処理により、送信データ数と受信データ数は一致するため、本応用例では、送信FIFOデータエンプティ割り込み (TXI) 要求のタイミングで、受信FIFOに格納されているデータを取得します。

送信データ格納領域に設定されているデータを順次送信します。

受信エラー状態を判定し、SCFRDR0より受信データを取得します。
※オーバランエラーが発生しないように送信処理後、すぐに受信処理を行ってください。

＊1 更新したはずの割り込み要因による割り込みの受け付けを避けるために、いったん当該フラグを含む内蔵モジュールのレジスタを読み出し、優先順位判定時間 (Pcyc×5) 待ってから本関数を抜けます。

図12　SCIF0 送受信割り込み処理フロー

3.3.4 SCIF0 データ受信処理フロー

SCIF0 割り込み処理 (INT_SCIF_SCIFI0( )) で、送信 FIFO データエンプティ割り込み時に SCIF0 データ受信処理 (scif_rcv_ch0( )) をコールします。

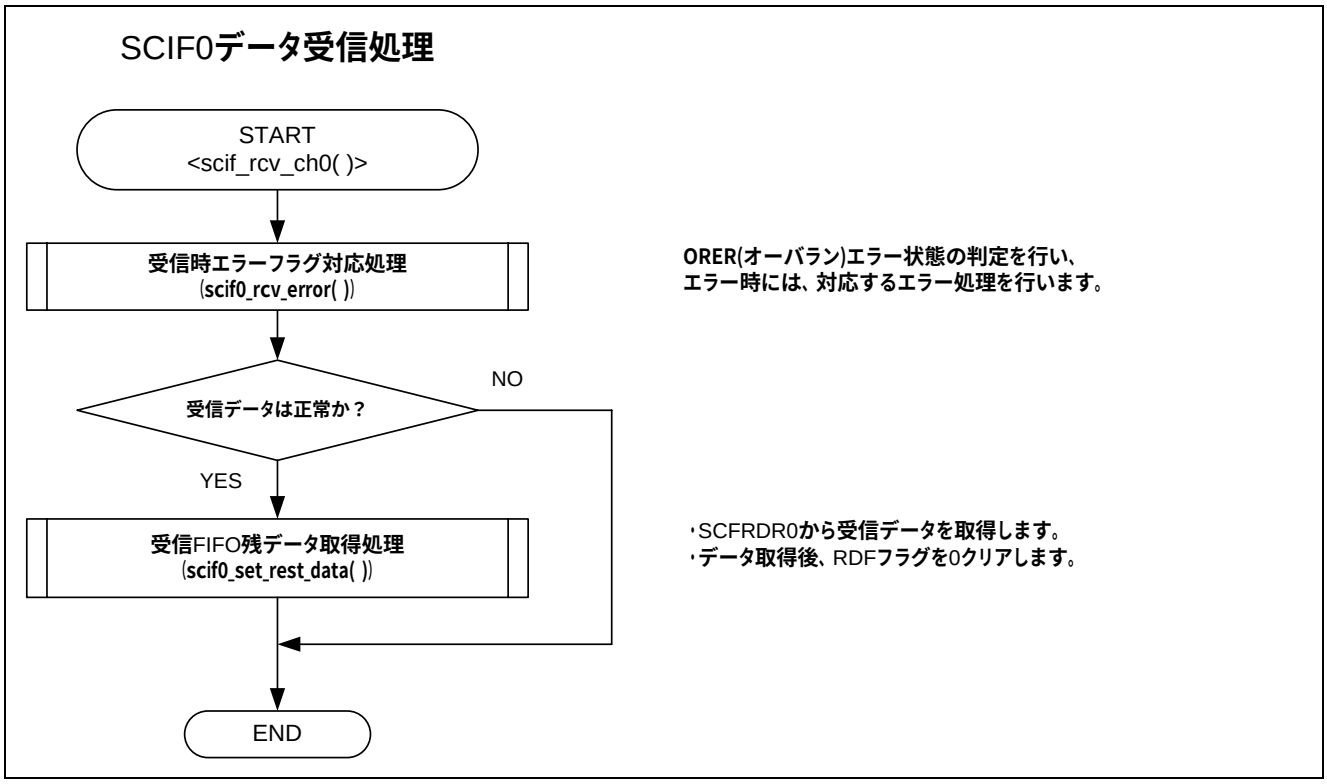


図13 SCIF0 データ受信処理フロー

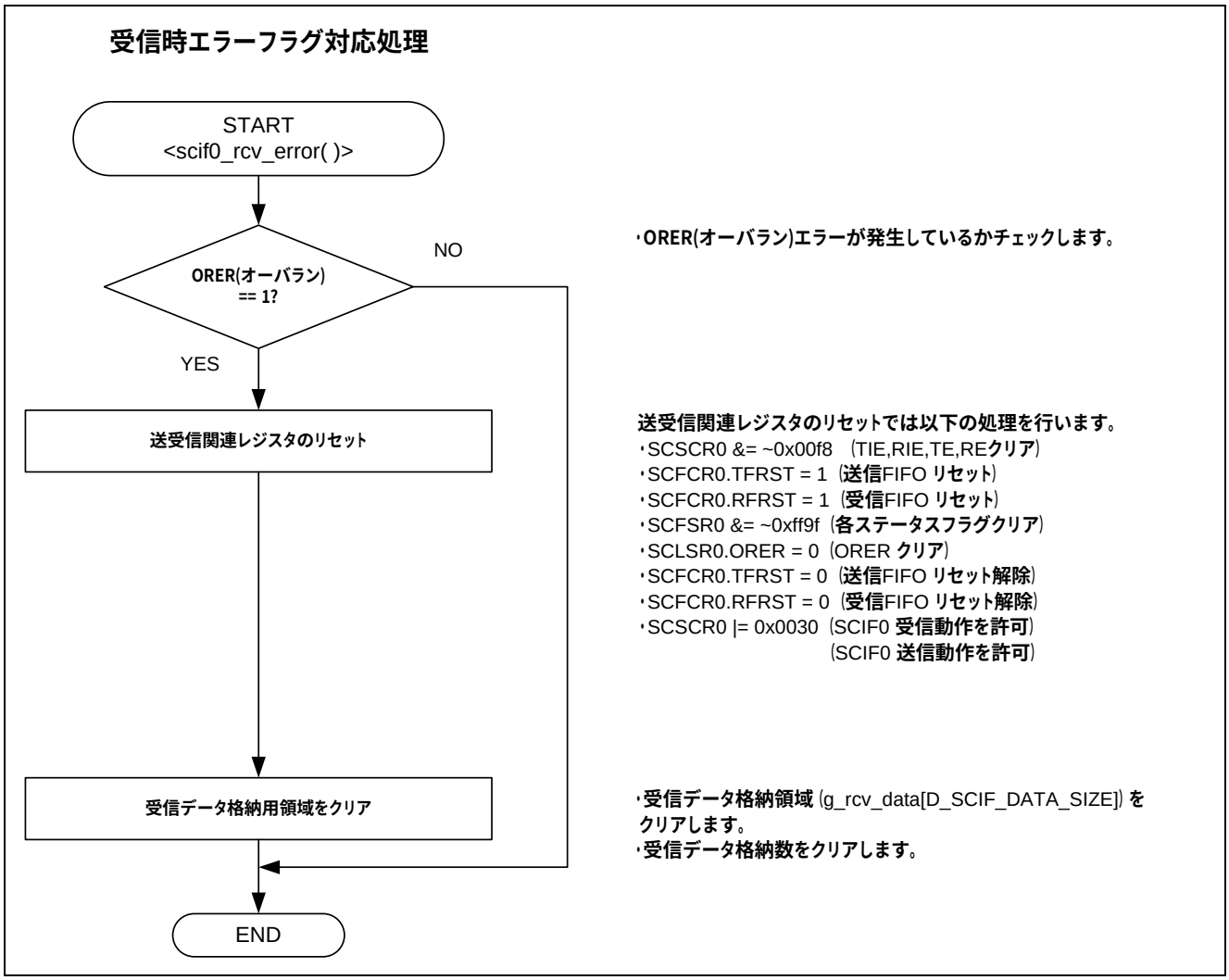


図14　SCIF0 受信時エラーフラグ対応処理フロー

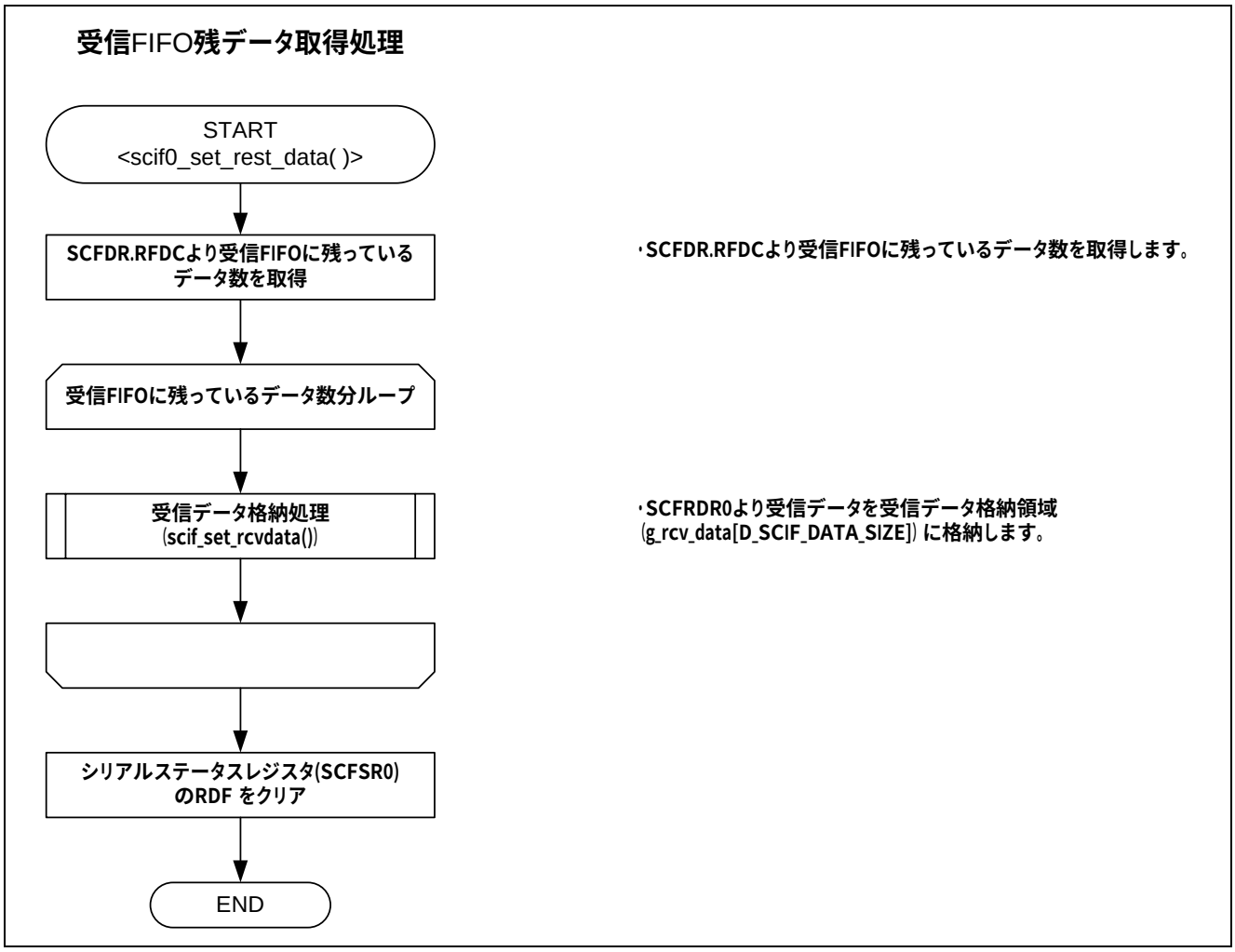


図15　受信 FIFO 残データ取得処理フロー

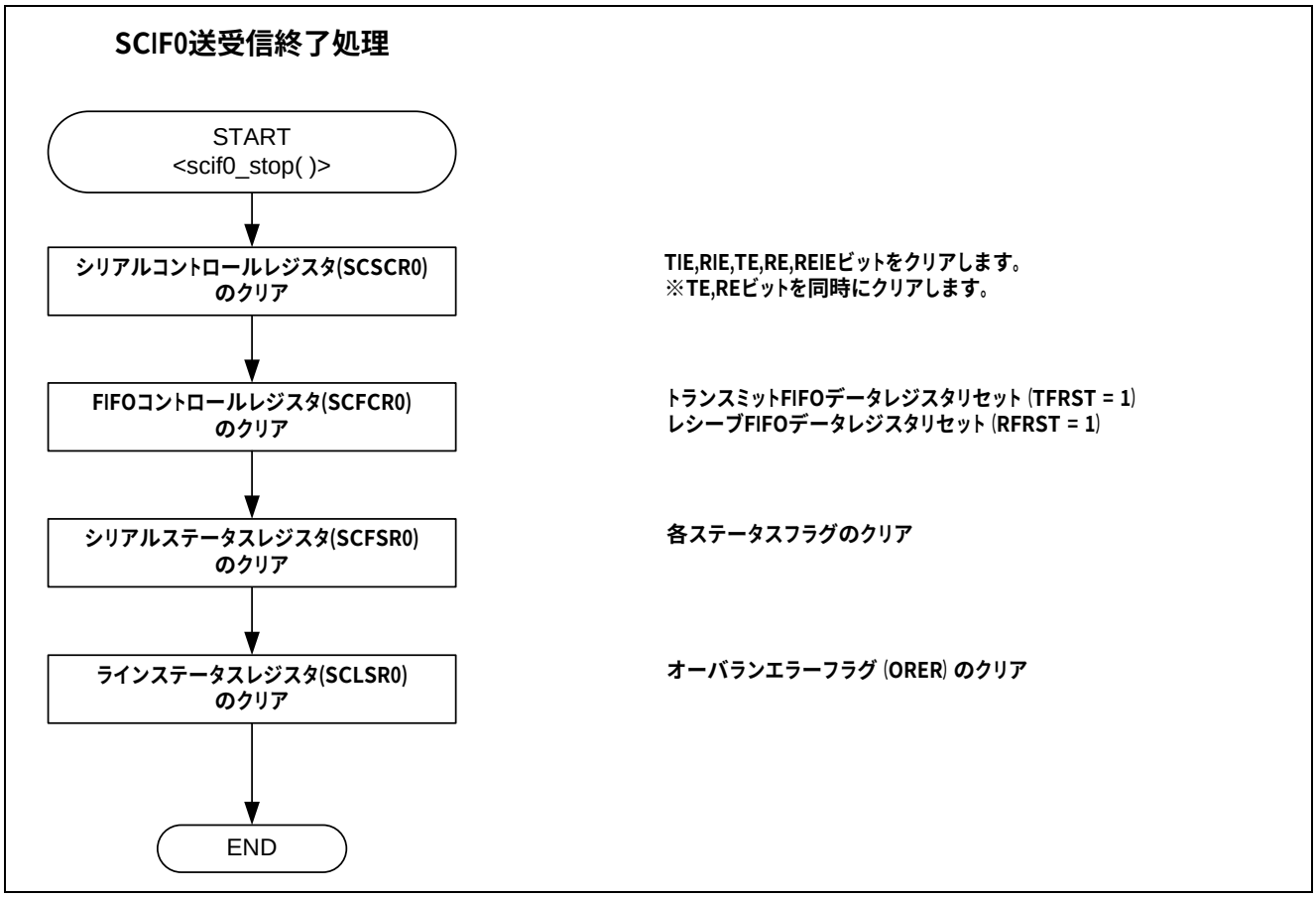


図16　SCIF0 送受信終了処理フロー

3.4 参考プログラムのインタフェース

本参考プログラムでは、SCIF でデータ送受信するために、主なインタフェースとして以下を作成します。

(1) SCIF0 初期化処理 (scif_init_SCIF0 ( ))

【概要】

SCIF0 をクロック同期式の送受信モードに初期化します。
レジスタの設定値に関しては、表 4を参照ください。
処理フローについては、図 7、図 8を参照ください。

【パラメータ】

なし。

(2) SCIF0 データ送信処理 (scif0_send ( ))

【概要】

SCIF0 よりデータ送信開始する際に使用します。
事前に SCIF0 初期化処理 (scif_init_SCIF0 ( )) をコールしておく必要があります。
処理フローについては、図 9を参照ください。

【処理内容】

1. 送信データ格納領域をクリア後、送信データ格納領域にパラメータで設定した送信データ数と送信データを設定します。ただし、送信データサイズが、D_SCIF_DATA_SIZE より大きい場合には設定を行いません。より大きいサイズのデータを送信したい場合は D_SCIF_DATA_SIZE の値を変更してください。

2. SCIF0 送信データ設定処理 (scif0_setdata( )) をコールします。

【パラメータ】

型	変数	内容
unsigned long	i_count	**送信データ数 (バイト単位)**
unsigned char*	i_ptr	**送信データ**

(3) SCIF0 送信データ設定処理 (scif0_setdata( ))

【概要】

送信データ格納領域に設定されている送信データを SCFTDR0 に設定する際にコールします。
SCIF0 データ送信処理 (scif0_send( )) (図 9) と送信 FIFO データエンプティ割り込み (TXI) 発生時にコールされる SCIF0 割り込み処理 (図 12) からコールされます。
事前に送信データ格納領域に送信データが設定されている必要があります。SCFTDR0 に送信データ設定後、送信データ格納領域に残送信データがある場合は、送信 FIFO データエンプティ割り込み (TXI) を有効にします。残送信データがない場合は、送信 FIFO データエンプティ割り込み (TXI) を無効にします。
処理フローについては、図 10を参照ください。

【処理内容】

1. 送信 FIFO の空き数分、送信データ格納領域に設定されている送信データを SCFTDR0 に設定します。
2. SCFTDR0 に送信データを設定後、送信データ格納領域に残送信データがある場合は送信 FIFO データエンプティ割り込み (TXI) 要求を許可にします。送信データ格納領域に残送信データがない場合は、送信 FIFO データエンプティ割り込み (TXI) を禁止にします。

【パラメータ】

なし。

(4) SCIF0 データ受信処理 (scif_rcv_ch0 ( ))

【概要】

SCIF1 よりデータ受信する際に使用します。
送信 FIFO データエンプティ割り込み (TXI) でコールされます。(図 12)
受信データを受信データ格納領域 (g_rcv_data[D_SCIF_DATA_SIZE]) に格納します。
※ただし受信したデータが D_SCIF_DATA_SIZE より大きくなる場合は、それ以降のデータは格納しません。
事前に SCIF0 初期化処理 (scif_init_SCIF0 ( )) をコールしておく必要があります。
処理フローについては、図 13を参照ください。

【処理内容】

1. 受信データのエラーチェックを行い、エラーの際にはおのおののエラーケースの処理を行います。(詳細は、受信時エラーフラグ対応処理 (scif0_rcv_error ( )) を参照ください。)
2. 1.の処理で正常データと判断された場合、受信 FIFO に設定されている分のデータを SCFRDR0 から取得します。

【パラメータ】

なし。

(5) 受信時エラーフラグ対応処理 (scif0_rcv_error ( ))

【概要】

SCIF0 データ受信処理 (scif_rcv_ch0 ( )) からコールされます。(図 13)
データ受信時の ORER (オーバラン) エラー状態を判定し、エラー時には D_SCIF_RET_NG を返します。
SCIF0 データ受信処理 (scif_rcv_ch0( )) では、本関数が、D_SCIF_RET_NG と返した際には、以降の SCIF0 データ受信処理での受信処理を行いません。
処理フローについては、図 14を参照ください。

【処理内容】

1. ORER (オーバラン) エラー発生時
・送受信関連のレジスタをリセットします。
・受信データ格納領域 (g_rcv_data[D_SCIF_DATA_SIZE]) をクリアします。

【パラメータ】

なし。

【戻り値】

受信時エラーフラグ状態正常　D_SCIF_RET_OK
受信時エラーフラグ状態異常　D_SCIF_RET_NG

(6) SCIF0 送受信終了処理 (scif0_stop ( ))

【概要】

SCIF0 の送受信処理を終了します。

【処理内容】

1. シリアルコントロールレジスタ (SCSCR) の TIE, RIE, TE, RE, REIE ビットをクリアします。
2. トランスミット FIFO データレジスタリセットのリセット動作を許可 (TFRST = 1) します。
3. レシーブ FIFO データレジスタリセットのリセット動作を許可 (RFRST = 1) します。
4. シリアルステータスレジスタ (SCFSR) の各ステータスフラグをクリアします。
5. ラインステータスレジスタ (SCLSR) のオーバランエラーフラグ (ORER) をクリアします。

【パラメータ】

なし。

【戻り値】

なし。

3.5 参考プログラムのレジスタ設定

表 4に SCIF0 の参考プログラムのレジスタ設定を示します。

表4 参考プログラムのレジスタ設定 (SCIF0)

レジスタ名	アドレス	設定値	機能と設定値
シリアルコントロールレジスタ (SCSCR0)	H'FFE00008	H'0000	• **初期化処理①** TE = "0": **送信動作を禁止** RE = "0": **受信動作を禁止** TIE = "0": **トランスミットインタラプトイネーブルを禁止** RIE = "0": **レシーブインタラプトイネーブルを禁止** REIE = "0": **レシーブエラーインタラプトイネーブルを禁止**
		H'0001(*1) or H'0002(*2)	• **初期化処理②** CKE[1:0] = "B'01": **内部クロック**/SCK **端子は出力端子** (*1 **の場合**) CKE[1:0] = "B'10": **外部クロック**/SCK **端子は入力端子** (*2 **の場合**)
		H'0031(*1) or H'0032(*2)	• **初期化処理③** TE = "1": **送信動作を許可** RE = "1": **受信動作を許可** **【注】** TE, RE **を同時に** 1 **にセットしてください。**
		H'00B1(*1) or H'00B2(*2)	• **送信処理時** TIE = "1": **トランスミットインタラプトイネーブルを許可**
		H'0031(*1) or H'0032(*2)	• **送信処理終了時** (**送信データ** = 0 **の場合**) TIE = "0": **トランスミットインタラプトイネーブルを禁止**
		H'0001(*1) or H'0002(*2)	• **送受信処理終了時** TE = "0": **送信動作を禁止** RE = "0": **受信動作を禁止** TIE = "1": **トランスミットインタラプトイネーブルを禁止** RIE = "1": **レシーブインタラプトイネーブルを禁止** **【注】** TE, RE **を同時に** 0 **にクリアしてください。**

(**次頁へ続く**)

レジスタ名	アドレス	設定値	機能と設定値
FIFO コントロールレジスタ (SCFCR0)	H'FFE00018	H'0006	• 初期化処理① TFRST = "1": トランスミット FIFO データレジスタリセット動作を許可 RFRST = "1": レシーブ FIFO データレジスタリセット動作を許可
		H'0080	• 初期化処理② TFRST = "0": トランスミット FIFO データレジスタリセット動作を禁止 RFRST = "0": レシーブ FIFO データレジスタリセット動作を禁止 TTRG[1:0] = "B'00": 送信 FIFO データ数のトリガ 送信トリガ数 = 8 ※#define により変更可能 RTRG[1:0] = "B'10": レシーブ FIFO データ数のトリガ 受信トリガ数 = 8 ※#define により変更可能 LOOP = "0": ループバックテストを禁止
シリアルステータスレジスタ (SCFSR0)	H'FFE00010	H'0060	• 初期化処理時 ER, BRK, RDF, DR ビットクリア
ラインステータスレジスタ (SCLSR0)	H'FFE00024	H'0000	• 初期化処理時 ORER ビットクリア
シリアルモードレジスタ (SCSMR0)	H'FFE00000	H'0080	• 初期化処理時 CA = "1": クロック同期式モード CHR = "0": 8 ビットデータ PE = "0": パリティビットの付加を禁止 CKS[1:0] = "0": Pφクロック ※#define により変更可能
ビットレートレジスタ (SCBRR0)	H'FFE00004	H'0052	• 初期化処理時 100Kbps 設定 ※#define により変更可能
ピンセレクトレジスタ B (PSELB)	H'A4050102	H'0000	• 初期化処理時 PSB7 = "0": SCIF0_RXD の選択 PSB8 = "0": SCIF0_TXD の選択
ポート Q コントロールレジスタ (PQCR)	H'A405001A	H'0000	• 初期化処理時 PQ0MD[1:0] = "0": その他の機能選択 PQ1MD[1:0] = "0": その他の機能選択 PQ2MD[1:0] = "0": その他の機能選択
割り込み優先レベル設定レジスタ G (IPRG)	H'A4080018	H'1000	• 初期化処理時 IPR0 = "B'0001": 優先レベル 1 設定

【注】 *1 シリアルクロック出力側 SH7730 の場合
*2 シリアルクロック入力側 SH7730 の場合

3.6 参考プログラムのマクロ定義

表 5に参考プログラムのマクロ定義を示します。

表5 参考プログラムで使用しているマクロ定義

マクロ定義	設定値	機能
D_SCIF_FIFO_NUM	16	FIFO 設定可能数: 16
D_SCIF_DATA_SIZE	50	送信・受信データ格納領域サイズ: 50 格納領域のサイズを変更できます。
SERIAL_CLOCK_OUT_MCU	—	シリアルクロック出力側 SH7730 の制御プログラムを有効にする場合は、この define を有効にしてください。
SERIAL_CLOCK_IN_MCU	—	シリアルクロック入力側 SH7730 の制御プログラムを有効にする場合は、この define を有効にしてください。
D_SCIF_DATA_NUM_SND_TRIGGER	8	トランスミット FIFO データ数トリガの設定値 ※設定値を以下 D_SCIF_SEND_FIFO_TRIGER0～8 から選択可能。 本参考プログラムでは D_SCIF_SEND_FIFO_TRIGER8 を設定
D_SCIF_DATA_NUM_RCV_TRIGGER	8	レシーブ FIFO データ数トリガの設定値 ※設定値を以下 D_SCIF_RCV_FIFO_TRIGER1～14 から選択可能。 本参考プログラムでは D_SCIF_RCV_FIFO_TRIGER8 を設定
D_SCIF_SEND_FIFO_TRIGER0	0	トランスミット FIFO データ数トリガ: 0
D_SCIF_SEND_FIFO_TRIGER2	2	トランスミット FIFO データ数トリガ: 2
D_SCIF_SEND_FIFO_TRIGER4	4	トランスミット FIFO データ数トリガ: 4
D_SCIF_SEND_FIFO_TRIGER8	8	トランスミット FIFO データ数トリガ: 8
D_SCIF_RCV_FIFO_TRIGER1	1	レシーブ FIFO データ数トリガ: 1
D_SCIF_RCV_FIFO_TRIGER2	2	レシーブ FIFO データ数トリガ: 2
D_SCIF_RCV_FIFO_TRIGER8	8	レシーブ FIFO データ数トリガ: 8
D_SCIF_RCV_FIFO_TRIGER14	14	レシーブ FIFO データ数トリガ: 14
D_SCIF_SEND_NO_END	0	送信データ送信中 【参考】 scif0_setdata_chk_end( )により送信処理状態が取得できます。
D_SCIF_SEND_END	1	送信データ送信完了 【参考】 scif0_setdata_chk_end( )により送信処理状態が取得できます。
D_SCIF_CBR_1K	0	ビットレート指定値: 1Kbps *
D_SCIF_CBR_2_5K	1	ビットレート指定値: 2.5Kbps *
D_SCIF_CBR_5K	2	ビットレート指定値: 5Kbps *
D_SCIF_CBR_10K	3	ビットレート指定値: 10Kbps *
D_SCIF_CBR_25K	4	ビットレート指定値: 25Kbps *
D_SCIF_CBR_50K	5	ビットレート指定値: 50Kbps *
D_SCIF_CBR_100K	6	ビットレート指定値: 100Kbps *
D_SCIF_CBR_250K	7	ビットレート指定値: 250Kbps *
D_SCIF_CBR_500K	8	ビットレート指定値: 500Kbps *

(次頁へ続く)

【注】 * ビットレート指定値のビットレートを満たす SCSMR の CKS[1:0]、SCBRR 設定値、1bit wait 用の値が、あらかじめ T_SCIF_CKS_SCBRR_SET_INFO gc_scif_cks_scbrr_tbl [ ]テーブルに格納されています。
テーブルの設定値は、Iϕ = 266.66MHz、Pϕ = 33.33MHz で動作する場合を前提としております。そのため、Iϕ、Pϕを変更する場合はテーブルの設定値も変更してください。
このテーブルに格納されている SCSMR の CKS[1:0]、SCBRR 設定値については、「SH7730 グループ ハードウェアマニュアル (RJJ09B0339) 22 章 FIFO 内蔵シリアルコミュニケーションインタフェース (SCIF) 22.3.8 ビットレートレジスタ (SCBRR)」を参照ください。

4. 参考プログラム例

(1) サンプルプログラムリスト "sh7730.c"

```
/***********************************************************************
* DISCLAIMER

* This software is supplied by Renesas Technology Corp. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Technology Corp. and is protected under
* all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
***********************************************************************/
/* Copyright (C) 2009. Renesas Technology Corp., All Rights Reserved.       */
/*""FILE COMMENT""*********** Technical reference data ***************
* System Name  : SH7730 Sample Program
* File Name    : sh7730.c
* Abstract     : SH7730 SCIF クロック同期式 全二重送受信設定例
* Version      : Ver 1.00
* Device       : SH7730
* Tool-Chain   : High-performance Embedded Workshop (Version 4.04.01.001)
*              : C/C++ Compiler Package for SuperH Family (V.9.02release00)
* OS           : None
* H/W Platform : アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
* Description  : SH7730 FIFO 内蔵シリアルコミュニケーション
*              : インターフェース(SCIF)
*              : クロック同期式全二重通信サンプルプログラム
* Operation    :
* Limitation   : 本参考プログラムは、シリアルクロック出力側 SH7730 と
*              : シリアルクロック入力側 SH7730 の２つのマイコンが必要となります。
*              : シリアルクロック出力側 SH7730 には、
*              : #ifdef SERIAL_CLOCK_OUT_MCU 対応プログラムを
*              : シリアルクロック入力側 SH7730 には、
*              : #ifdef SERIAL_CLOCK_IN_MCU 対応プログラムを
*              : それぞれに書き込みクロック同期式全二重通信を行います。
*              :
*              : SERIAL_CLOCK_OUT_MCU, SERIAL_CLOCK_IN_MCU の
*              : どちらか片方のみを有効にしてご使用ください。
*              :
*********************************************************************
* History      : 27.Feb.2009 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/

#include <machine.h>
#include "iodefine.h"
#include "scif.h"

```

```
  62    void main(void);
  63
  64    #ifdef SERIAL_CLOCK_OUT_MCU /* シリアルクロック出力側 SH7730 用 */
  65
  66    void Chk_scif_clock_in_mcu_prepare(void);
  67
  68    /*""FUNC COMMENT""************************************************
  69    * ID                    :
  70    * Outline               : シリアルクロック出力側 SH7730：サンプルプログラムメイン
  71    *                       : (クロック同期式シリアル送受信処理)
  72    * Include               :
  73    * Declaration           : void main(void)
  74    * Description           : シリアルクロック出力側 SH7730 の
  75    *                       : クロック同期式シリアル全二重通信例を示します。
  76    *                       :
  77    *                       : SCIF0 からデータを送受信し
  78    *                       : 正しくシリアルクロック入力側 SH7730（相手側）からの
  79    *                       : データが受信できていることを確認します。
  80    *                       :
  81    *                       : 1.シリアルクロック入力側 SH7730 の送受信準備
  82    *                       :   が完了するまで待ちます。
  83    *                       :   (PTE1=Low,PTE2=High になった際に準備完了とします。)
  84    *                       : 2.SCIF0 を送受信モードで初期化します。
  85    *                       : 3.SCIF0_SCK 端子より同期クロックを出力し、
  86    *                       :   SCIF0_TXD 端子よりデータを順次送信します。
  87    *                       : 4.SCIF0_SCK 端子からの同期クロックに同期し、
  88    *                       :   SCIF0_RXD 端子よりシリアルクロック入力側 SH7730
  89    *                       :   からのデータを順次受信します。
  90    *                       : 5.SCIF0 の受信データ格納領域のデータと
  91    *                       :   シリアルクロック入力側 SH7730 から送信された
  92    *                       :   データが等しいことを確認します。
  93    *                       :
  94    * Limitation            : シリアルクロック入力側 SH7730（相手側）に、
  95    *                       : 下記のプログラム
  96    *                       : （シリアルクロック入力側 SH7730：サンプルプログラムメイン）
  97    *                       : が書き込まれていることとします。
  98    *                       :
  99    *                       : シリアルクロック入力側 SH7730（相手側）送受信準備完了後
 100    *                       : データ送受信を開始します。
 101    *                       :
 102    * Argument              : none
 103    * Return Value          : none
 104    * Calling Functions     :
 105    *""FUNC COMMENT END""*******************************************/
 106    void main(void)
 107    {
 108      unsigned long    send_num = 36;             /* 送信データ数 */
 109      unsigned char    senddata[36];              /* 全送信データ格納領域   */
 110      unsigned char    checkdata[36];             /* 受信期待データ格納領域 */
 111
 112      unsigned char    *ptr;
 113      unsigned char    rcvdata[D_SCIF_DATA_SIZE]; /* 受信データ格納用   */
 114      unsigned long    rcv_num = 0;               /* 受信データ数格納用 */
 115      int              result = 0;                /* 比較結果格納用     */
 116      int              i;
 117
 118      memset(rcvdata, 0x00, sizeof(rcvdata));
 119
 120      /* データ設定 */
 121      for(i = 0; i < 36; i++)
 122      {
 123        senddata[i] = i;                      /* 送信データ設定（0x00～0x23）        */
 124        checkdata[i] = 35 - i;                /* 受信期待データ設定（0x23～0x00）    */
 125                                              /*（シリアルクロック入力側 SH7730 から   */
 126                                              /* 送信されるデータ）                 */
 127
```

```
128      }
129
130      /* 受信データ格納領域をクリアする */
131      scif_clear_rcvdata();
132
133
134      /* シリアルクロック入力側 SH7730 送受信準備完了待ち */
135      Chk_scif_clock_in_mcu_prepare();            /* 本関数で準備完了まで待ちます */
136
137
138      /* ==== SCIF0 をクロック同期式送受信モード初期化==== */
139      scif_init_SCIF0();
140
141      ptr = senddata;
142
143      /* ==== SCIF0 データ送信処理 ==== */
144      scif0_send(send_num, ptr);          /* 送信データ数 */
145                                        /* 送信データ */
146
147      /* データ送受信完了待ち */
148      /* 全データが送受信完了するまで待ちます */
149      while(D_SCIF_SEND_NO_END == scif0_setdata_chk_end());
150
151      /* 受信データ確認 */
152      while (1)
153      {
154        /* 受信 FIFO に残っているデータを取得する */
155        scif0_set_rest_data();
156
157        /* 受信データ格納領域よりデータを取得 */
158        scif_get_rcvdata(&rcv_num, rcvdata);
159
160        /* シリアルクロック入力側 SH7730 からの送信データ数 == 受信データ数となった時 */
161        if(send_num == rcv_num)
162        {
163            /* シリアルクロック入力側 SH7730 からの送信データと受信データを比較する */
164            result = memcmp(checkdata, rcvdata, rcv_num);
165
166            /* 一致した場合ループを抜ける */
167            if(result == 0)
168            {
169                /* SCIF 送受信終了処理 */
170                scif0_stop();
171                break;
172            }
173            /* エラーケース */
174            else
175            {
176                /* データが一致しない場合は無限ループ */
177                while(1);
178            }
179
180        }
181      }
182
183      /* ここまで処理が実行されたことにより全二重通信でクロック入力側 SH7730 から
184         送信されるデータを正しく受信できたことが確認できます */
185      while (1)
186      {
187
188      }
189    }
190
191    /*""FUNC COMMENT""************************************************
       * ID                    :
```

```
192    * Outline               : シリアルクロック入力側 SH7730 送受信準備完了待ち
193    * Include               :
194    * Declaration            : void Chk_scif_clock_in_mcu_prepare(void)
195    * Description            : シリアルクロック入力側 SH7730 送受信準備完了
196    *                       : 判定を行います。
197    *                       :
198    *                       : PTE1 が Low、PTE2 が High の信号を
199    *                       : シリアルクロック入力側 SH7730 から入力する
200    *                       : まで待ちます。
201    *                       : 3 回異なるタイミングで信号を確認し、
202    *                       : 3 回とも PTE1 が Low、PTE2 が High を確認した
203    *                       : 場合に送受信準備完了と判定します。
204    *                       :
205    * Argument              : none
206    * Return Value           : none
207    * Calling Functions       :
208    *""FUNC COMMENT END""*********************************************/
209    void Chk_scif_clock_in_mcu_prepare(void)
210    {
211      int      i;
212      int      count;
213
214      PFC.PECR.BIT.PE1MD = 2;     /* PTE1 ポート入力 */
215      PFC.PECR.BIT.PE2MD = 2;     /* PTE2 ポート入力 */
216
217      /* シリアルクロック入力側 SH7730（相手側）初期化完了待ち */
218      do
219      {
220        count = 0;
221
222        if((PORT.PEDR.BIT.B1 == 0) && (PORT.PEDR.BIT.B2 == 1))
223        {
224            count++;
225        }
226
227        for(i = 0; i < 1000; i ++);
228
229        if((PORT.PEDR.BIT.B1 == 0) && (PORT.PEDR.BIT.B2 == 1))
230        {
231            count++;
232        }
233
234        for(i = 0; i < 1000; i ++);
235
236        if((PORT.PEDR.BIT.B1 == 0) && (PORT.PEDR.BIT.B2 == 1))
237        {
238            count++;
239        }
240
241        if(count == 3)
242        {
243            break;
244        }
245
246      }while(1);
247
248    }
249
250    #endif /* SERIAL_CLOCK_OUT_MCU */
251
252    #ifdef SERIAL_CLOCK_IN_MCU /* シリアルクロック入力側 SH7730 用 */
253    /*""FUNC COMMENT""*************************************************
254    * ID                    :
255    * Outline                : シリアルクロック入力側 SH7730：サンプルプログラムメイン
256    *                       : (クロック同期式シリアル送受信処理)
```

```
257     * Include               :
258     * Declaration            : void main(void)
259     * Description            : シリアルクロック入力側 SH7730 の
260     *                        : クロック同期式シリアル全二重通信例を示します。
261     *                        :
262     *                        : シリアルクロック出力側 SH7730 からの
263     *                        : 同期クロックを入力し、そのクロック
264     *                        : に同期してデータ送受信を行います。
265     *                        :
266     *                        : 1.SCIF0 を送受信モードで初期化します。
267     *                        : 2.シリアルクロック出力側 SH7730 に送受信準備完了
268     *                        :   を通知します。(PTE1=Low,PTE2=High で送受信準備完了とします)
269     *                        : 3.シリアルクロック出力側 SH7730 からの
270     *                        :   同期クロックに同期し、
271     *                        :   SCIF0_TXD 端子よりデータを順次送信します。
272     *                        : 4.SCIF0_SCK 端子からの同期クロックに同期し、
273     *                        :   SCIF0_RXD 端子よりシリアルクロック出力側 SH7730
274     *                        :   からのデータを順次受信します。
275     *                        : 5.SCIF0 の受信データ格納領域のデータと
276     *                        :   シリアルクロック出力側 SH7730 から
277     *                        :   送信されたデータが等しいことを確認します。
278     *                        :
279     * Limitation              : シリアルクロック出力側 SH7730 に、
280     *                        : 上記のプログラム
281     *                        :（シリアルクロック出力側 SH7730：サンプルプログラムメイン）
282     *                        : が書き込まれていることとします。
283     *                        :
284     * Argument               : none
285     * Return Value            : none
286     * Calling Functions       :
287     *""FUNC COMMENT END""*********************************************/
288     void main(void)
289     {
290       unsigned long    send_num = 36;             /* 送信データ数 */
291       unsigned char    senddata[36];              /* 全送信データ格納用                */
292       unsigned char    checkdata[36];             /* 受信期待データ格納用              */
293
294       unsigned char    *ptr;
295       unsigned char    rcvdata[D_SCIF_DATA_SIZE]; /* 受信データ格納用   */
296       unsigned long    rcv_num = 0;               /* 受信データ数格納用 */
297       int              result = 0;                /* 比較結果格納用     */
298       int              i;
299
300       memset(rcvdata, 0x00, sizeof(rcvdata));
301
302       /* ==== ポート E 初期化 ==== */
303       PFC.PECR.BIT.PE1MD = 2;     /* PTE1 ポート入力 */
304       PFC.PECR.BIT.PE2MD = 2;     /* PTE2 ポート入力 */
305
306       /* データ設定 */
307       for(i = 0; i < 36; i++)
308       {
309         senddata[i] = 35 - i;                /* 送信データ設定（0x23～0x00）        */
310         checkdata[i] = i;                    /* 受信期待データ設定（0x00～0x23）    */
311                                              /*（シリアルクロック入力側 SH7730 から   */
312                                              /* 送信されるデータ）                */
313       }
314
315       /* 受信データ格納領域をクリアする */
316       scif_clear_rcvdata();
317
318       /* ==== SCIF0 をクロック同期式送受信モード初期化==== */
319       scif_init_SCIF0();
320
321       ptr = senddata;
```

```
322
323     /* ==== SCIF0 データ送信処理 ==== */
324     scif0_send(send_num, ptr);          /* 送信データ数 */
325                                       /* 送信データ */
326
327     /* シリアルクロック出力側 SH7730（相手側）へ送受信準備完了を通知 */
328     /* PTE1=Low,PTE2=High で送受信準備完了とします。 */
329     PFC.PECR.BIT.PE1MD = 1;     /* PTE1 ポート出力 */
330     PORT.PEDR.BIT.B1 = 0;       /* PTE1 ⇒ Low */
331
332     /* データ送信完了待ち */
333     /* 全データが送信完了するまで待ちます */
334     while(D_SCIF_SEND_NO_END == scif0_setdata_chk_end());
335
336     /* 受信データ確認 */
337     while (1)
338     {
339       /* 受信 FIFO に残っているデータを取得する */
340       scif0_set_rest_data();
341
342       /* 受信データ格納領域よりデータを取得 */
343       scif_get_rcvdata(&rcv_num, rcvdata);
344
345       /* 送信データ数 == 受信データ数となった時 */
346       if(send_num == rcv_num)
347       {
348           /* 受信期待データと受信データを比較する */
349           result = memcmp(checkdata, rcvdata, rcv_num);
350
351           /* 一致した場合ループを抜ける */
352           if(result == 0)
353           {
354               /* SCIF 送受信終了処理 */
355               scif0_stop();
356               break;
357           }
358           /* エラーケース */
359           else
360           {
361               /* データが一致しない場合は無限ループ */
362               while(1);
363           }
364
365       }
366     }
367
368     /* ここまで処理が実行されたことにより全二重通信でシリアルクロック出力側 SH7730 から
369        送信されるデータを正しく受信できたことが確認できます */
370     while (1)
371     {
372
373     }
374   }
375
376   #endif /* SERIAL_CLOCK_IN_MCU */
```

(2) サンプルプログラムリスト "scif.c"

```
/***********************************************************************
* DISCLAIMER

* This software is supplied by Renesas Technology Corp. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Technology Corp. and is protected under
* all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
***********************************************************************/
/* Copyright (C) 2009. Renesas Technology Corp., All Rights Reserved.       */
/*""FILE COMMENT""*********** Technical reference data ***************
* System Name  : SH7730 Sample Program
* File Name    : scif.c
* Abstract     : SH7730 SCIF クロック同期式 全二重送受信設定例
* Version      : Ver 1.00
* Device       : SH7730
* Tool-Chain   : High-performance Embedded Workshop (Version 4.04.01.001)
*              : C/C++ Compiler Package for SuperH Family (V.9.02release00)
* OS           : None
* H/W Platform : アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
* Description  : SH7730 FIFO 内蔵シリアルコミュニケーション
*              : インタフェース(SCIF)
*              : クロック同期式全二重通信サンプルプログラム
* Operation    :
* Limitation   :
*              :
*********************************************************************
* History      : 27.Feb.2009 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/

#include <machine.h>
#include <stdio.h>
#include <stdlib.h>
#include "iodefine.h"
#include "scif.h"

/*""VAL DEF""*/
/* ==== 送信データ格納用 ==== */
unsigned char  g_send_data[D_SCIF_DATA_SIZE];        /* 送信データ格納領域 */
unsigned long  g_send_count = 0;                     /* 送信データ数 */

/* ==== 受信データ格納用 ==== */
unsigned char  g_rcv_data[D_SCIF_DATA_SIZE];         /* 受信データ格納領域 */
unsigned long  g_rcv_count = 0;                      /* 受信データ数 */

/* ==== レジスタ(SCBRR、CKS)設定値テーブル ==== */
```

```
  65    /* 注意
  66       以下の設定値は Iφ=266.66MHz、Pφ=33.33MHz で動作している場合の設定値
  67       となります。Iφ、Pφを変更する場合は以下の設定値も変更してください
  68    */
  69    const static T_SCIF_CKS_SCBRR_SET_INFO gc_scif_cks_scbrr_tbl[] =
  70    {
  71      /*    SCBRR,  CKS,    1bitwait */
  72      { 129,    3,      266667},       /*   1Kbps */
  73      { 52,     3,      106667},       /* 2.5Kbps */
  74      { 104,    2,      53333},        /*   5Kbps */
  75      { 52,     2,      26667},        /*  10Kbps */
  76      { 83,     1,      10667},        /*  25Kbps */
  77      { 166,    0,      5333},         /*  50Kbps */
  78      { 82,     0,      2667},         /* 100Kbps */
  79      { 33,     0,      1067},         /* 250Kbps */
  80      { 16,     0,      533}           /* 500Kbps */
  81    };
  82
  83    /* ==== レジスタ(SCFCR.TTRG) 設定値テーブル ==== */
  84    const static T_SCIF_SCFCR_TTRG_SET gc_scif_ttrg_tbl[] =
  85    {
  86      { D_SCIF_SEND_FIFO_TRIGER8, 0x00},  /* トランスミット FIFO データ数トリガ = 8 */
  87      { D_SCIF_SEND_FIFO_TRIGER4, 0x01},  /* トランスミット FIFO データ数トリガ = 4 */
  88      { D_SCIF_SEND_FIFO_TRIGER2, 0x02},  /* トランスミット FIFO データ数トリガ = 2 */
  89      { D_SCIF_SEND_FIFO_TRIGER0, 0x03}   /* トランスミット FIFO データ数トリガ = 0 */
  90    };
  91
  92    /* ==== レジスタ(SCFCR.RTRG) 設定値テーブル ==== */
  93    const static T_SCIF_SCFCR_RTRG_SET gc_scif_rtrg_tbl[] =
  94    {
  95      { D_SCIF_RCV_FIFO_TRIGER1, 0x00},   /* レシーブ FIFO データ数トリガ = 1 */
  96      { D_SCIF_RCV_FIFO_TRIGER2, 0x01},   /* レシーブ FIFO データ数トリガ = 2 */
  97      { D_SCIF_RCV_FIFO_TRIGER8, 0x02},   /* レシーブ FIFO データ数トリガ = 8 */
  98      { D_SCIF_RCV_FIFO_TRIGER14, 0x03}   /* レシーブ FIFO データ数トリガ = 14 */
  99    };
 100    /*""VAL DEF END""*/
 101
 102    /* ==== プロトタイプ宣言 ==== */
 103    static void scif_setdata_to_SCFTDR_ch0(unsigned long i_num, unsigned char *i_pdata);
 104    static void scif0_reset_status(void);
 105    static int  scif0_rcv_error(void);
 106
 107    static void scif_set_rcvdata(unsigned char *i_pRcv_data);
 108    static unsigned char scif_get_clock_select(T_SCIF_bit_rate_type i_type_bps);
 109    static unsigned char scif_get_SCBRR(T_SCIF_bit_rate_type i_type_bps);
 110    static unsigned char scif_get_snd_trigger_num(T_SCIF_send_fifo_trigger i_trigger_num);
 111    static unsigned char scif_get_rcv_trigger_num(T_SCIF_rcv_fifo_trigger i_trigger_num);
 112    static void scif_wait_1bit(unsigned long i_bit_rate);
 113
 114    /*""FUNC COMMENT""************************************************
 115    * ID                    :
 116    * Outline                : SCIF0 初期化処理
 117    * Include                :
 118    * Declaration             : void scif_init_SCIF0(void)
 119    * Description             : SCIF0 を送受信モードで初期化します。
 120    *                       : SERIAL_CLOCK_OUT_MCU が有効時には、
 121    *                       : シリアルクロック出力側 SH7730 制御
 122    *                       : SERIAL_CLOCK_IN_MCU が有効時には、
 123    *                       : シリアルクロック入力側 SH7730 制御
 124    *                       : が行われます。
 125    *                       :
 126    * Limitation              : ボーレート設定値は内部クロックを使った
 127    *                       : 周辺モジュール用動作周波数Pφ=33.33MHz
 128    *                       : の場合です。
 129    *                       : 他のクロックを使用する場合はビットレート
```

```
130     *                       : 設定値を変更してください。
131     *                       : gc_scif_cks_scbrr_tbl[]の設定値を
132     *                       : 修正してください。
133     *                       :
134     * Argument               : none
135     * Return Value           : none
136     * Calling Functions      :
137     *"""FUNC COMMENT END"""*************************************************/
138     void scif_init_SCIF0(void)
139     {
140       unsigned long    dummy;
141
142       /* ==== SCIF0 の初期設定 ==== */
143       /* ==== モジュールストップレジスタの設定 ==== */
144       LOWP.MSTPCR0 &= ~0x00000080;
145
146       dummy = LOWP.MSTPCR0;    /* 設定反映確認のためダミリード */
147
148       /* ==== シリアルコントロールレジスタ(SCSCR)の設定 ==== */
149       SCIF0.SCSCR.WORD = 0x0000;      /* SCIF0 送／受信動作停止 */
150
151       /* ==== FIFO コントロールレジスタ(SCFCR)の設定 ==== */
152       SCIF0.SCFCR.BIT.TFRST = 1;      /* 送信 FIFO リセット */
153       SCIF0.SCFCR.BIT.RFRST = 1;      /* 受信 FIFO リセット */
154
155       /* ==== シリアルステータスレジスタ(SCFSR)の初期化 ==== */
156       SCIF0.SCFSR.WORD &= ~0xff9f;/* ER,BRK,FER,PER,RDF,DR ビットクリア */
157
158       /* ==== ラインステータスレジスタ(SCLSR)の設定 ==== */
159       SCIF0.SCLSR.BIT.ORER = 0;       /* ORER ビットクリア */
160
161       /* ==== シリアルモードレジスタ(SCSMR)の設定 ==== */
162       /* コミュニケーションモード ：クロック同期式 */
163       SCIF0.SCSMR.WORD = 0x0080;
164
165     #ifdef SERIAL_CLOCK_OUT_MCU    /* シリアルクロック出力側 SH7730 制御時 */
166
167       /* ==== シリアルコントロールレジスタ(SCSCR)の設定 ==== */
168       SCIF0.SCSCR.BIT.CKE = 1;     /* B'01:内部クロック 同期クロック出力 */
169
170     #else   /* シリアルクロック入力側 SH7730 制御時 */
171
172       /* ==== シリアルコントロールレジスタ(SCSCR)の設定 ==== */
173       SCIF0.SCSCR.BIT.CKE = 2;     /* B'10:外部クロック 同期クロック入力 */
174
175     #endif
176
177       /* ==== クロックセレクト : ビットレートの設定値により決定 ==== */
178       SCIF0.SCSMR.BIT.CKS = scif_get_clock_select(D_SCIF_SCBRR_100K);
179
180       /* ==== ビットレートレジスタ(SCBRR)の設定  : ビットレートの設定値により決定 ==== */
181       SCIF0.SCBRR = scif_get_SCBRR(D_SCIF_SCBRR_100K);
182
183       /* 1 ビット期間経過待ち */
184       scif_wait_1bit(D_SCIF_SCBRR_100K);
185
186       /* ==== FIFO コントロールレジスタ(SCFCR)の設定 ==== */
187       /* 送信 FIFO データレジスタリセット ：禁止 */
188       /* 受信 FIFO データレジスタリセット ：禁止 */
189       /* ループバックテスト ：禁止 */
190       SCIF0.SCFCR.WORD = 0x0000;
191
192       /* 送信 FIFO データ数トリガ ：データ数 8 */
193       /* 送信 FIFO データ数トリガは D_SCIF_DATA_NUM_SND_TRIGGER の設定値になります */
194       SCIF0.SCFCR.BIT.TTRG = scif_get_snd_trigger_num(D_SCIF_DATA_NUM_SND_TRIGGER);
```

```
195
196     /* 受信 FIFO データ数トリガ ：データ数 8 */
197     /* 受信 FIFO データ数トリガは D_SCIF_DATA_NUM_RCV_TRIGGER の設定値になります */
198     SCIF0.SCFCR.BIT.RTRG = scif_get_rcv_trigger_num(D_SCIF_DATA_NUM_RCV_TRIGGER);
199
200     /* ==== 割り込み優先レベル設定（優先レベル 1）==== */
201     INTC0.IPRG = INTC0.IPRG | 0x1000;
202
203     /* ==== ピンセレクトレジスタ B（PSELB）==== */
204     PFC.PSELB.BIT.PSB7 = 0;     /* SCIF0_RXD の選択 */
205     PFC.PSELB.BIT.PSB8 = 0;     /* SCIF0_TXD の選択 */
206
207     /* ==== ポート Q コントロールレジスタ（PQCR）の設定 ==== */
208     PFC.PQCR.BIT.PQ2MD = 0;     /* その他の機能選択 */
209     PFC.PQCR.BIT.PQ1MD = 0;     /* その他の機能選択 */
210     PFC.PQCR.BIT.PQ0MD = 0;     /* その他の機能選択 */
211
212     /* ==== シリアルコントロールレジスタ(SCSCR)の設定 ==== */
213     SCIF0.SCSCR.WORD |= 0x0030;     /* SCIF0 受信動作を許可する */
214                                     /* SCIF0 送信動作を許可する */
215
216   }
217
218   /*""FUNC COMMENT""************************************************
219   * ID                    :
220   * Outline               : クロックセレクト設定値取得
221   * Include               :
222   * Declaration           : unsigned char scif_get_clock_select
223   *                       :   (T_SCIF_bit_rate_type i_type_bps)
224   * Description           : ビットレートに対応するクロックセレクト
225   *                       : CKS[1:0]設定値を取得する。
226   *                       :
227   * Argument              : T_SCIF_bit_rate_type i_type_bps :
228   *                       :  ビットレート指定値
229   * Return Value          : unsigned char : クロックセレクト設定値
230   * Calling Functions     :
231   *""FUNC COMMENT END""*********************************************/
232   static unsigned char scif_get_clock_select(T_SCIF_bit_rate_type i_type_bps)
233   {
234     return       gc_scif_cks_scbrr_tbl[i_type_bps].mscsmr_cks;
235   }
236
237   /*""FUNC COMMENT""************************************************
238   * ID                    :
239   * Outline               : ビットレートレジスタ設定値(SCBRR)取得
240   * Include               :
241   * Declaration           : unsigned char scif_get_SCBRR
242   *                       :   (T_SCIF_bit_rate_type i_type_bps)
243   * Description           : ビットレートに対応するビットレート
244   *                       : レジスタ設定値を取得します。
245   *                       :
246   * Argument              : T_SCIF_bit_rate_type i_type_bps :
247   *                       :  ビットレート指定値
248   * Return Value          : unsigned char : ビットレートレジスタ設定値
249   * Calling Functions     :
250   *""FUNC COMMENT END""*********************************************/
251   static unsigned char scif_get_SCBRR(T_SCIF_bit_rate_type i_type_bps)
252   {
253     return       gc_scif_cks_scbrr_tbl[i_type_bps].mscbrr;
254   }
255
256   /*""FUNC COMMENT""************************************************
257   * ID                    :
258   * Outline               : SCIF0 データ送信処理
259   * Include               :
```

```
260    * Declaration            : void scif0_send
261    *                       :   (unsigned long i_count,
262    *                       :    unsigned char *i_ptr)
263    *                       :
264    * Description            : 送信データを送信データ格納領域に保存し
265    *                       : SCIF0 送信データ設定処理をコールします。
266    *                       :
267    * Argument               : unsigned long  i_count :
268    *                       :  送信データ数
269    *                       : unsigned char *i_ptr  :
270    *                       :  送信データ
271    * Return Value           : none
272    * Calling Functions       :
273    *""FUNC COMMENT END""***********************************************/
274    void scif0_send(unsigned long i_count, unsigned char *i_ptr)
275    {
276      /* SCIF 送信データ格納領域設定処理 */
277      scif_set_send_data(i_count, i_ptr);
278
279      /* SCIF0 送信データ設定処理 */
280      scif0_setdata();
281
282    }
283
284    /*""FUNC COMMENT""***************************************************
285    * ID                    :
286    * Outline                : 送信データ送信完了判定処理
287    * Include                :
288    * Declaration            : T_SCIF_send_chk_end
289    *                       :    scif0_setdata_chk_end(void)
290    *                       :
291    * Description            : 送信データ格納領域の送信データがあるか
292    *                       : どうか確認し、送信データがない場合には
293    *                       : 最終データ送信完了を確認するため、
294    *                       : TEND が 1 になるまで待ちます。
295    *                       :
296    * Argument               : none
297    * Return Value           : D_SCIF_SEND_NO_END(送信中)
298    *                       : D_SCIF_SEND_END(送信完了)
299    * Calling Functions       :
300    *""FUNC COMMENT END""***********************************************/
301    T_SCIF_send_chk_end scif0_setdata_chk_end(void)
302    {
303      unsigned long            send_num = 0;
304      T_SCIF_send_chk_end      ret;
305
306      /* SCIF 送信データ数取得処理 */
307      scif_get_send_data_num(&send_num);
308
309      /* 全てのデータを送信完了した場合 */
310      if(send_num == 0)
311      {
312        /* ==== 送信完了判定 ==== */
313        while(SCIF0.SCFSR.BIT.TEND == 0)
314        {
315        }
316
317        ret = D_SCIF_SEND_END;
318      }
319      /* 送信中の場合 */
320      else
321      {
322        ret = D_SCIF_SEND_NO_END;
323      }
324
```

```
325       return ret;
326
327     }
328
329     /*""FUNC COMMENT""************************************************
330     * ID                    :
331     * Outline               : SCIF0 送信データ設定処理
332     * Include               :
333     * Declaration           : void scif0_setdata(void)
334     *                       :
335     * Description           : 送信データ格納領域から送信データを取得し
336     *                       : SCFTDR0 データ設定処理をコールします。
337     *                       :
338     * Argument              : none
339     * Return Value          : none
340     * Calling Functions     :
341     *""FUNC COMMENT END""*******************************************/
342     void scif0_setdata(void)
343     {
344       unsigned long       send_num = 0;
345       unsigned char       send_data[D_SCIF_DATA_SIZE];
346
347       memset(send_data, 0x00, sizeof(send_data));
348
349       /* SCIF 送信データ数取得処理 */
350       scif_get_send_data_num(&send_num);
351
352       /* 送信データなし */
353       if(send_num <= 0)
354       {
355         return;
356       }
357
358       /* SCIF 送信データ取得処理 */
359       scif_get_send_data(send_num, send_data);
360
361       /* SCFTDR0 データ設定処理 */
362       scif_setdata_to_SCFTDR_ch0(send_num, send_data);
363
364     }
365
366     /*""FUNC COMMENT""************************************************
367     * ID                    :
368     * Outline               : SCFTDR0 データ設定処理
369     * Include               :
370     * Declaration           : void scif_setdata_to_SCFTDR_ch0
371     *                       :   (unsigned long i_num,
372     *                       :    unsigned char *i_pdata)
373     *                       :
374     * Description           : TDFE の状態を確認し送信 FIFO データエンプティ
375     *                       : であることを確認します。
376     *                       : SCFDR.TFDC を確認して設定可能な数の
377     *                       : 送信データを SCFTDR0 に設定します。
378     *                       : SCFTDR0 に送信データを設定後、送信データ
379     *                       : 格納領域を更新します。
380     *                       : データ設定後 TDFE,TEND をクリアします。
381     *                       : SCFTDR0 に送信データを設定後、
382     *                       : 残送信データがある場合は、
383     *                       : 送信データエンプティ割り込みを有効にします。
384     *                       : 残送信データがない場合は、
385     *                       : 送信データエンプティ割り込みを無効にします。
386     *                       :
387     * Argument              : unsigned long  i_num :
388     *                       :  送信データ数
389     *                       : unsigned char *i_pdata  :
```

```
390    *                       :  送信データ
391    *                       :
392    * Return Value           : none
393    * Calling Functions      :
394    *""FUNC COMMENT END""*************************************************/
395    static void scif_setdata_to_SCFTDR_ch0(unsigned long i_num, unsigned char *i_pdata)
396    {
397      int                  i = 0;
398      unsigned long        set_send_fifo_num = 0;
399
400      /* ==== シリアルステータスレジスタ(SCSFR0)の
401      送信 FIFO データエンプティを確認（TDFE フラグ）==== */
402      if(SCIF0.SCFSR.BIT.TDFE == 0)
403      {
404        return;
405      }
406
407      /* 未送信データ数取得 */
408      set_send_fifo_num = SCIF0.SCFDR.BIT.TFDC;
409
410      /* 送信 FIFO に設定可能なデータ数を計算 */
411      set_send_fifo_num = D_SCIF_FIFO_NUM - set_send_fifo_num;
412
413      /* 送信 FIFO 設定数 */
414      if(set_send_fifo_num > i_num)
415      {
416        set_send_fifo_num = i_num;
417      }
418
419      /* ==== 送信 FIFO データレジスタ(SCFTDR0)に送信データを書き込む ==== */
420      for(i = 0; i < set_send_fifo_num; i++)
421      {
422        i_num--;
423        SCIF0.SCFTDR = *i_pdata++;
424      }
425
426      /* SCIF 送信データ格納領域を更新する */
427      scif_set_send_data(i_num, i_pdata);
428
429      /* ==== シリアルステータスレジスタ(SCFSR0)の TDFE,TEND を
430      読出し後にクリアする ==== */
431      SCIF0.SCFSR.BIT.TDFE = 0;
432      SCIF0.SCFSR.BIT.TEND = 0;
433
434      /* ==== シリアルコントロールレジスタ(SCSCR)の設定 ==== */
435      /* 残送信データ数 > 0? */
436      if(i_num > 0)
437      {
438        SCIF0.SCSCR.BIT.TIE = 1;   /* 送信 FIFO データエンプティ割り込み有効 */
439      }
440      else
441      {
442        SCIF0.SCSCR.BIT.TIE = 0;   /* 送信 FIFO データエンプティ割り込み無効 */
443      }
444
445    }
446
447    /*""FUNC COMMENT""*************************************************
448    * ID                    :
449    * Outline               : SCIF 送信データ格納領域設定処理
450    * Include               :
451    * Declaration           : void scif_set_send_data
452    *                       :   (unsigned long i_count,
453    *                       :    unsigned char *i_ptr)
454    * Description           : 送信データを送信データ格納領域に設定します。
```

```
455    *                       :
456    * Limitation              : 送信データ格納領域より大きいデータの場合
457    *                       : 設定処理は行ないません。
458    * Argument               : unsigned long  i_count :
459    *                       :  送信データ数
460    *                       : unsigned char *i_ptr  :
461    *                       :  送信データ
462    * Return Value           : none
463    * Calling Functions       :
464    *""FUNC COMMENT END""*********************************************/
465    void scif_set_send_data(unsigned long i_count, unsigned char *i_ptr)
466    {
467      /* 設定データチェック */
468      if(i_count > sizeof(g_send_data))
469      {
470        return;
471      }
472
473      /* 送信データ領域のクリア */
474      scif_clear_senddata();
475
476      if(i_count > 0)
477      {
478        /* 送信データ数設定 */
479        g_send_count = i_count;
480
481        /* 送信データを設定 */
482        memcpy(g_send_data, i_ptr, g_send_count);
483      }
484
485    }
486
487    /*""FUNC COMMENT""*************************************************
488    * ID                    :
489    * Outline                : SCIF 送信データ数取得処理
490    * Include                :
491    * Declaration            : void scif_get_send_data_num
492    *                       :   (unsigned long *o_pnum)
493    * Description            : 送信データ格納領域より送信データ数を
494    *                       : 取得します。
495    *                       :
496    * Argument               : unsigned long  *o_pnum :
497    *                       :  送信データ数
498    * Return Value           : none
499    * Calling Functions       :
500    *""FUNC COMMENT END""*********************************************/
501    void scif_get_send_data_num(unsigned long *o_pnum)
502    {
503
504      /* パラメータチェック */
505      if(NULL == o_pnum)
506      {
507        return;
508      }
509
510      /* 送信データ数取得 */
511      *o_pnum = g_send_count;
512
513    }
514
515    /*""FUNC COMMENT""*************************************************
516    * ID                    :
517    * Outline                : SCIF 送信データ取得処理
518    * Include                :
519    * Declaration            : void scif_get_send_data
```

```
520    *                       :   (unsigned long i_num,
521    *                       :    unsigned char *o_ptr)
522    * Description            : 送信データを送信データ格納領域より
523    *                       : 取得します。
524    *                       :
525    * Limitation             : *o_ptr は scif_get_send_data_num()
526    *                       : で取得した領域サイズ以上分を指定してください。
527    *                       :
528    * Argument               : unsigned long  i_num :
529    *                       :  取得送信データ数
530    *                       : unsigned char *o_ptr :
531    *                       :  送信データ
532    * Return Value           : none
533    * Calling Functions       :
534    *""FUNC COMMENT END""*********************************************/
535    void scif_get_send_data(unsigned long i_num, unsigned char *o_ptr)
536    {
537      /* パラメータチェック */
538      if((NULL == o_ptr) || (i_num > sizeof(g_send_data)))
539      {
540        return;
541      }
542
543      /* 送信データを取得 */
544      memcpy(o_ptr, g_send_data, i_num);
545
546    }
547
548    /*""FUNC COMMENT""*************************************************
549    * ID                    :
550    * Outline               : SCIF0 データ受信処理
551    * Include               :
552    * Declaration            : void scif_rcv_ch0(void)
553    * Description            : 本関数は、送信 FIFO データエンプティ割り込み
554    *                       : 処理でコールされます。
555    *                       : SCIF0 の受信時のエラー状態を判定し、
556    *                       : 受信可能であるか決定します。
557    *                       ; 正常データである場合には、
558    *                       : 受信処理を行います。
559    *                       :
560    * Argument               : none
561    * Return Value           : none
562    * Calling Functions       :
563    *""FUNC COMMENT END""*********************************************/
564    void scif_rcv_ch0(void)
565    {
566      /* 受信時エラーフラグ対応処理 */
567      if(D_SCIF_RET_OK != scif0_rcv_error())
568      {
569        return;
570      }
571
572      /*  受信 FIFO 残データ取得処理 */
573      scif0_set_rest_data();
574
575    }
576
577    /*""FUNC COMMENT""*************************************************
578    * ID                    :
579    * Outline               : 受信時エラーフラグ対応処理
580    * Include               :
581    * Declaration            : int scif0_rcv_error(void)
582    * Description            : データ受信時に変化するエラーフラグ
583    *                       : (ORER)を参照し、エラー時の処理を行います。
584    *                       : ◆ORER(オーバラン)エラーの時
```

```
585    *                       :  受信関連レジスタをリセットし、
586    *                       :  受信データ格納領域をクリアします。
587    *                       :
588    * Argument               : none
589    * Return Value           : int : エラー発生有無
590    * Calling Functions      :
591    *""FUNC COMMENT END""*************************************************/
592    static int  scif0_rcv_error(void)
593    {
594      int       ret = D_SCIF_RET_OK;
595
596      /* ==== ORER(オーバラン)エラーの時 ==== */
597      /* ORER(オーバラン)エラーの時に行いたい処理を定義してください */
598      if(SCIF0.SCLSR.BIT.ORER == 1)
599      {
600        /* 送受信初期化処理 */
601        scif0_reset_status();
602
603        /* 受信データ格納用領域をクリア */
604        scif_clear_rcvdata();
605
606        ret = D_SCIF_RET_NG;
607
608      }
609
610      return   ret;
611    }
612
613    /*""FUNC COMMENT""*****************************************************
614    * ID                    :
615    * Outline               : 受信 FIFO 残データ取得処理
616    * Include               :
617    * Declaration           : void scif0_set_rest_data(void)
618    * Description           : 受信 FIFO に残っているデータを取得する。
619    *                       :
620    * Argument              : none
621    * Return Value          : none
622    * Calling Functions     :
623    *""FUNC COMMENT END""*************************************************/
624    void scif0_set_rest_data(void)
625    {
626      int       i = 0;
627      int       rest_rcvnum = 0;
628
629      /* 受信 FIFO に残っているデータ数を取得する */
630      rest_rcvnum = SCIF0.SCFDR.BIT.RFDC;
631
632      /* 受信 FIFO に残っているデータを取得する */
633      for(i = 0; i < rest_rcvnum; i++)
634      {
635        /* ==== 受信 FIFO データレジスタ(SCFRDR0)から受信データを読み出す ==== */
636        scif_set_rcvdata(&SCIF0.SCFRDR);
637
638      }
639
640      /* ==== シリアルステータスレジスタ(SCFSR0)の RDF をクリアする ====     */
641      /* RDF=1 を読み出した後、SCFRDR0 内の受信データ数が受信トリガ設定数より */
642      /* 少なくなるまで SCFRDR0 を読み出し、0 を書き込んだときクリアされる     */
643      SCIF0.SCFSR.BIT.RDF = 0; /* RDF ビットクリア */
644
645    }
646
647    /*""FUNC COMMENT""*****************************************************
648    * ID                    :
649    * Outline               : 送受信初期化処理
```

```
650     * Include               :
651     * Declaration            : void scif0_reset_status(void)
652     * Description            : 送受信関連のビットを初期化し、
653     *                       : FIFO のデータもクリアする。
654     *                       :
655     * Argument               : none
656     * Return Value           : none
657     * Calling Functions      :
658     *"""FUNC COMMENT END"""*************************************************/
659     static void scif0_reset_status(void)
660     {
661       /* 送受信初期化処理 */
662       SCIF0.SCSCR.WORD &= ~0x00f8;/* TIE,RIE,TE,RE クリア */
663       SCIF0.SCFCR.BIT.RFRST = 1;      /* 受信 FIFO リセット */
664       SCIF0.SCFCR.BIT.TFRST = 1;      /* 送信 FIFO リセット */
665       SCIF1.SCFSR.WORD &= ~0xff9fu;   /* 各フラグビットクリア */
666       SCIF0.SCLSR.BIT.ORER = 0;       /* ORER クリア */
667       SCIF0.SCFCR.BIT.RFRST = 0;      /* 受信 FIFO リセット解除 */
668       SCIF0.SCFCR.BIT.TFRST = 0;      /* 送信 FIFO リセット解除 */
669       SCIF0.SCSCR.WORD |= 0x0030;     /* SCIF0 受信動作を許可する */
670                                   /* SCIF0 送信動作を許可する */
671
672     }
673
674     /*"""FUNC COMMENT""************************************************
675     * ID                    :
676     * Outline               : SCIF0 送受信終了処理
677     * Include               :
678     * Declaration            : void scif0_stop(void)
679     * Description            : 全データ送受信完了後
680     *                       : SCSCR の TE,RE ビットをクリアします。
681     *                       :
682     * Argument               : none
683     * Return Value           : none
684     * Calling Functions      :
685     *"""FUNC COMMENT END""*************************************************/
686     void scif0_stop(void)
687     {
688       /* ==== シリアルコントロールレジスタ(SCSCR)の設定 ==== */
689       SCIF0.SCSCR.WORD &= ~0x00f8;    /* TIE,RIE,TE,RE クリア */
690       SCIF0.SCFCR.BIT.RFRST = 1;          /* 受信 FIFO リセット */
691       SCIF0.SCFCR.BIT.TFRST = 1;          /* 送信 FIFO リセット */
692       SCIF1.SCFSR.WORD &= ~0xff9fu;       /* 各ステータスフラグビットクリア */
693       SCIF0.SCLSR.BIT.ORER = 0;           /* ORER クリア */
694
695     }
696
697     /*"""FUNC COMMENT""************************************************
698     * ID                    :
699     * Outline               : 受信データ格納処理
700     * Include               :
701     * Declaration            : void scif_set_rcvdata(unsigned char *i_pRcv_data)
702     * Description            : 引数で指定した受信データを受信データ
703     *                       : 格納領域に設定します。
704     *                       :
705     * Argument               : unsigned char *i_pRcv_data：受信データ
706     * Return Value           : none
707     * Calling Functions      :
708     *"""FUNC COMMENT END""*************************************************/
709     static void scif_set_rcvdata(unsigned char *i_pRcv_data)
710     {
711       /* 受信データ格納領域に受信データを設定する */
712       if(g_rcv_count < sizeof(g_rcv_data))
713       {
714         g_rcv_data[g_rcv_count] = *i_pRcv_data;
```

```
715       g_rcv_count++;
716     }
717
718   }
719
720   /*""FUNC COMMENT""************************************************
721   * ID                   :
722   * Outline               : 送信データ格納領域クリア処理
723   * Include               :
724   * Declaration            : void scif_clear_senddata(void)
725   * Description            : 送信データ格納領域をクリアする。
726   *                       :
727   * Argument              : none
728   * Return Value           : none
729   * Calling Functions       :
730   *""FUNC COMMENT END""*******************************************/
731   void scif_clear_senddata(void)
732   {
733     g_send_count = 0;
734     memset(g_send_data, 0x00, sizeof(g_send_data));
735   }
736
737   /*""FUNC COMMENT""************************************************
738   * ID                   :
739   * Outline               : 受信データ格納領域クリア処理
740   * Include               :
741   * Declaration            : void scif_clear_rcvdata(void)
742   * Description            : 受信データ格納領域をクリアする。
743   *                       :
744   * Argument              : none
745   * Return Value           : none
746   * Calling Functions       :
747   *""FUNC COMMENT END""*******************************************/
748   void scif_clear_rcvdata(void)
749   {
750     g_rcv_count = 0;
751     memset(g_rcv_data, 0x00, sizeof(g_rcv_data));
752   }
753
754   /*""FUNC COMMENT""************************************************
755   * ID                   :
756   * Outline               : 格納受信データ取得処理
757   * Include               :
758   * Declaration            : void scif_get_rcvdata
759   *                       :   (unsigned long *o_pRcvdata_num,
760   *                       :    unsigned char *o_pRcvdata)
761   * Description            : 格納受信データを取得する。
762   *                       :
763   * Argument              : unsigned long  *o_pRcvdata_num :
764   *                       :  格納受信データ数
765   *                       : unsigned char  *o_pRcvdata  :
766   *                       :  格納受信データ
767   * Return Value           : none
768   * Calling Functions       :
769   *""FUNC COMMENT END""*******************************************/
770   void scif_get_rcvdata(unsigned long *o_pRcvdata_num, unsigned char *o_pRcvdata)
771   {
772     if((o_pRcvdata_num == NULL) || (o_pRcvdata == NULL))
773     {
774       return;
775     }
776
777     *o_pRcvdata_num = g_rcv_count;
778
779     memcpy(o_pRcvdata, g_rcv_data, g_rcv_count);
```

```
780
781     return;
782   }
783
784   /*""FUNC COMMENT""************************************************
785   * ID                    :
786   * Outline               : SCFCR.TTRG 設定値取得処理
787   * Include               :
788   * Declaration           : unsigned char scif_get_snd_trigger_num
789   *                       :   (T_SCIF_send_fifo_trigger i_trigger_num)
790   * Description           : 引数で指定したデータ数トリガに対応する
791   *                       : SCFCR.TTRG 設定値を取得する。
792   *                       :
793   * Argument              : T_SCIF_send_fifo_trigger  i_trigger_num
794   *                       :  データ数トリガ
795   * Return Value          : unsigned char :
796   *                       : レジスタ設定値(SCFCR.TTRG に設定)
797   * Calling Functions     :
798   *""FUNC COMMENT END""*******************************************/
799   static unsigned char scif_get_snd_trigger_num(T_SCIF_send_fifo_trigger i_trigger_num)
800   {
801     unsigned char    ret = 0;
802     int              i = 0;
803     int              count = 0;
804
805     count = sizeof(gc_scif_ttrg_tbl) / sizeof(T_SCIF_SCFCR_TTRG_SET);
806
807     for(i = 0; i < count; i++)
808     {
809       if(gc_scif_ttrg_tbl[i].mtrigger_num == i_trigger_num)
810       {
811           /* ==== SCFCR.TTRG 設定 ==== */
812           ret = gc_scif_ttrg_tbl[i].mttrg;
813           break;
814       }
815     }
816
817     return ret;
818   }
819
820   /*""FUNC COMMENT""************************************************
821   * ID                    :
822   * Outline               : SCFCR.RTRG 設定値取得処理
823   * Include               :
824   * Declaration           : unsigned char scif_get_rcv_trigger_num
825   *                       :   (T_SCIF_rcv_fifo_trigger i_trigger_num)
826   * Description           : 引数で指定したデータ数トリガに対応する
827   *                       : SCFCR.RTRG 設定値を取得する。
828   *                       :
829   * Argument              : T_SCIF_rcv_fifo_trigger  i_trigger_num :
830   *                       :  データ数トリガ
831   * Return Value          : unsigned char :
832   *                       : レジスタ設定値(SCFCR.RTRG に設定)
833   * Calling Functions     :
834   *""FUNC COMMENT END""*******************************************/
835   static unsigned char scif_get_rcv_trigger_num(T_SCIF_rcv_fifo_trigger i_trigger_num)
836   {
837     unsigned char    ret = 0;
838     int              i = 0;
839     int              count = 0;
840
841     count = sizeof(gc_scif_rtrg_tbl) / sizeof(T_SCIF_SCFCR_RTRG_SET);
842
843     for(i = 0; i < count; i++)
844     {
```

```
845         if(gc_scif_rtrg_tbl[i].mtrigger_num == i_trigger_num)
846         {
847             /* ==== SCFCR.RTRG 設定 ==== */
848             ret = gc_scif_rtrg_tbl[i].mrtrg;
849             break;
850         }
851       }
852
853       return ret;
854
855     }
856
857     /*""FUNC COMMENT""************************************************
858     * ID                    :
859     * Outline               : 1 ビット期間経過待ち処理
860     * Include               :
861     * Declaration           : void scif_wait_1bit(
862     *                       :   unsigned long i_bit_rate)
863     *                       :
864     * Description           : 指定したビットレートに対応した
865     *                       : 1 ビット期間以上の十分な時間経過後に
866     *                       : 本関数を抜けます。
867     *                       :
868     *                       : gc_scif_cks_scbrr_tbl テーブルに設定され
869     *                       : ている cpu 実行回数分 nop 処理を行い、
870     *                       : 1 ビット期間以上の経過を待ちます。
871     *                       :
872     * Limitation            : Iφを変更した場合は、
873     *                       : gc_scif_cks_scbrr_tbl テーブルの設定値
874     *                       : も修正ください。
875     *                       :
876     * Argument              : long i_bit_rate :  動作ビットレート
877     *                       :
878     * Return Value          : none
879     * Calling Functions     :
880     *""FUNC COMMENT END""*********************************************/
881     static void scif_wait_1bit(unsigned long i_bit_rate)
882     {
883       unsigned long    cpu_count;
884       unsigned long    i;
885
886       /* 1 ビット期間以上の十分な待ち時間分の cpu 実行回数を取得 */
887       cpu_count = gc_scif_cks_scbrr_tbl[i_bit_rate].cpu_count_1bit;
888
889       for(i = 0; i < cpu_count; i++)
890       {
891         nop();
892       }
893
894       return;
895     }
896
897     /* End of File */
```

(3) サンプルプログラムリスト "scif.h"

```
/***********************************************************************
* DISCLAIMER

* This software is supplied by Renesas Technology Corp. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Technology Corp. and is protected under
* all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* TECHNOLOGY CORP. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
***********************************************************************/
/* Copyright (C) 2009. Renesas Technology Corp., All Rights Reserved.       */
/*""FILE COMMENT""*********** Technical reference data ***************
* System Name  : SH7730 Sample Program
* File Name    : scif.h
* Abstract     : SH7730 SCIF クロック同期式 全二重送受信設定例
* Version      : Ver 1.00
* Device       : SH7730
* Tool-Chain   : High-performance Embedded Workshop (Version 4.04.01.001)
*              : C/C++ Compiler Package for SuperH Family (V.9.02release00)
* OS           : None
* H/W Platform : アルファプロジェクト製 SH-4A ボード 型番 AP-SH4A-1A
* Description  : SH7730 FIFO 内蔵シリアルコミュニケーション
*              : インタフェース(SCIF)
*              : クロック同期式全二重通信サンプルプログラム
* Operation    :
* Limitation   :
*              :
*********************************************************************
* History      : 27.Feb.2009 Ver. 1.00 First Release
*""FILE COMMENT END""**************************************************/

#ifndef  __SCIF_DEF_H__
#define __SCIF_DEF_H__

/* ==== マクロ定義 ==== */
/* 以下より、書き込むプログラムを選択してください。
   シリアルクロック出力側 SH7730 用 SERIAL_CLOCK_OUT_MCU
   シリアルクロック入力側 SH7730 用 SERIAL_CLOCK_IN_MCU
   ※どちらか片方のみ定義してください。
*/
#define SERIAL_CLOCK_OUT_MCU

/* 戻り値 */
#define D_SCIF_RET_NG         -1
#define D_SCIF_RET_OK         0

/* FIFO 設定可能数 */
```

```
  65    #define D_SCIF_FIFO_NUM           16
  66
  67    /* 送信・受信データ格納領域サイズ */
  68    #define D_SCIF_DATA_SIZE      50
  69
  70    /* ==== FIFO データ数トリガ ==== */
  71    /* 動作させたい FIFO のトリガ数を以下の定義している */
  72    /* デファインから設定してください                */
  73    #define D_SCIF_DATA_NUM_SND_TRIGGER       D_SCIF_SEND_FIFO_TRIGER8
  74                                                  /* 設定可能値 0,2,4,8 */
  75
  76    #define D_SCIF_DATA_NUM_RCV_TRIGGER       D_SCIF_RCV_FIFO_TRIGER8
  77                                                  /* 設定可能値 1,2,8,14 */
  78
  79    /* 送信 FIFO データ数トリガ */
  80    typedef enum
  81    {
  82      D_SCIF_SEND_FIFO_TRIGER0 = 0,               /* 送信 FIFO データトリガ数 0 */
  83      D_SCIF_SEND_FIFO_TRIGER2 = 2,               /* 送信 FIFO データトリガ数 2 */
  84      D_SCIF_SEND_FIFO_TRIGER4 = 4,               /* 送信 FIFO データトリガ数 4 */
  85      D_SCIF_SEND_FIFO_TRIGER8 = 8              /* 送信 FIFO データトリガ数 8 */
  86    } T_SCIF_send_fifo_trigger;
  87
  88    /* 受信 FIFO データ数トリガ */
  89    typedef enum
  90    {
  91      D_SCIF_RCV_FIFO_TRIGER1 = 1,              /* 受信 FIFO データトリガ数 1 */
  92      D_SCIF_RCV_FIFO_TRIGER2 = 2,              /* 受信 FIFO データトリガ数 2 */
  93      D_SCIF_RCV_FIFO_TRIGER8 = 8,              /* 受信 FIFO データトリガ数 8 */
  94      D_SCIF_RCV_FIFO_TRIGER14 = 14               /* 受信 FIFO データトリガ数 14 */
  95    } T_SCIF_rcv_fifo_trigger;
  96
  97    /* 送信 FIFO データ数トリガ */
  98    typedef enum
  99    {
 100      D_SCIF_SEND_NO_END = 0,                     /* 送信中   */
 101      D_SCIF_SEND_END                             /* 送信完了 */
 102    } T_SCIF_send_chk_end;
 103
 104    /* ==== ビットレート設定タイプ ==== */
 105    typedef enum
 106    {
 107       D_SCIF_SCBRR_1K,
 108       D_SCIF_SCBRR_2_5K,
 109       D_SCIF_SCBRR_5K,
 110       D_SCIF_SCBRR_10K,
 111       D_SCIF_SCBRR_25K,
 112       D_SCIF_SCBRR_50K,
 113       D_SCIF_SCBRR_100K,
 114       D_SCIF_SCBRR_250K,
 115       D_SCIF_SCBRR_500K
 116    } T_SCIF_bit_rate_type;
 117
 118    /* ==== 構造体定義 ==== */
 119    /* SCBRR 設定情報 */
 120    typedef struct {
 121      unsigned char    mscbrr;
 122      unsigned char    mscsmr_cks;
 123      unsigned long    cpu_count_1bit;
 124    } T_SCIF_CKS_SCBRR_SET_INFO;
 125
 126    /* SCFCR.TTRG 設定用 */
 127    typedef struct {
 128      unsigned char    mtrigger_num;
 129      unsigned char    mttrg;
```

```
130    } T_SCIF_SCFCR_TTRG_SET;
131
132    /* SCFCR.RTRG 設定用 */
133    typedef struct {
134      unsigned char    mtrigger_num;
135      unsigned char    mrtrg;
136    } T_SCIF_SCFCR_RTRG_SET;
137
138    /* ==== 関数宣言 ==== */
139    void scif_init_SCIF0(void);
140    void scif0_send(unsigned long i_count, unsigned char *i_ptr);
141    void scif0_setdata(void);
142    void scif0_set_rest_data(void);
143    void scif0_stop(void);
144    void scif_rcv_ch0(void);
145    T_SCIF_send_chk_end scif0_setdata_chk_end(void);
146
147    void scif_set_send_data(unsigned long i_count, unsigned char *i_ptr);
148    void scif_get_send_data_num(unsigned long *o_pnum);
149    void scif_get_send_data(unsigned long i_num, unsigned char *o_ptr);
150    void scif_clear_senddata(void);
151    void scif_clear_rcvdata(void);
152    void scif_get_rcvdata(unsigned long *o_pRcvdata_num, unsigned char *o_pRcvdata);
153
154    #endif /* __SCIF_DEF_H__ */
```

(4) サンプルプログラムリスト "intprg.c"
SCIF0 に関連する割り込み処理を実装しています。

```
1       …途中省略…
2
3       /* H'C00 SCIF SCIFI0 */
4       void INT_SCIF_SCIFI0(void)
5       {
6         unsigned short   dummy;
7
8         /* 送信 FIFO データエンプティ割り込み要求あり? */
9         if(SCIF0.SCFSR.WORD & 0x0020)           /* TDFE フラグセット? */
10        {
11
12          /* SCIF0 送信データ設定処理 */
13          scif0_setdata();
14
15          /* SCIF0 受信処理 */
16          scif_rcv_ch0();
17
18          /* 更新したはずの割り込み要因による割り込みの受け付けを避ける対応 */
19          dummy = SCIF0.SCFSR.WORD;
20          int_responstime_wait(INTC_RESPONSEWAIT); /* 優先順位判定時間分待ち */
21
22        }
23
24      }
25
26      …途中省略…
```

(5) サンプルプログラムリスト"vecttbl.src "

SCIF0 に関連する割り込み実行時の割り込み優先度を設定しています。
SCIF0 に関連する割り込みの優先度を 1 に設定しているため、SCIF0 に関連する割り込み中に新たな SCIF0 に関連する割り込みが発生しないように優先度に 1 を設定しています。

```
1       …途中省略…
2
3       ;SCIF
4                   ;H'C00      SCIF SCIFI0
5                   .data.b     H'10
6
7       …途中省略…
8
```

5. 実行結果

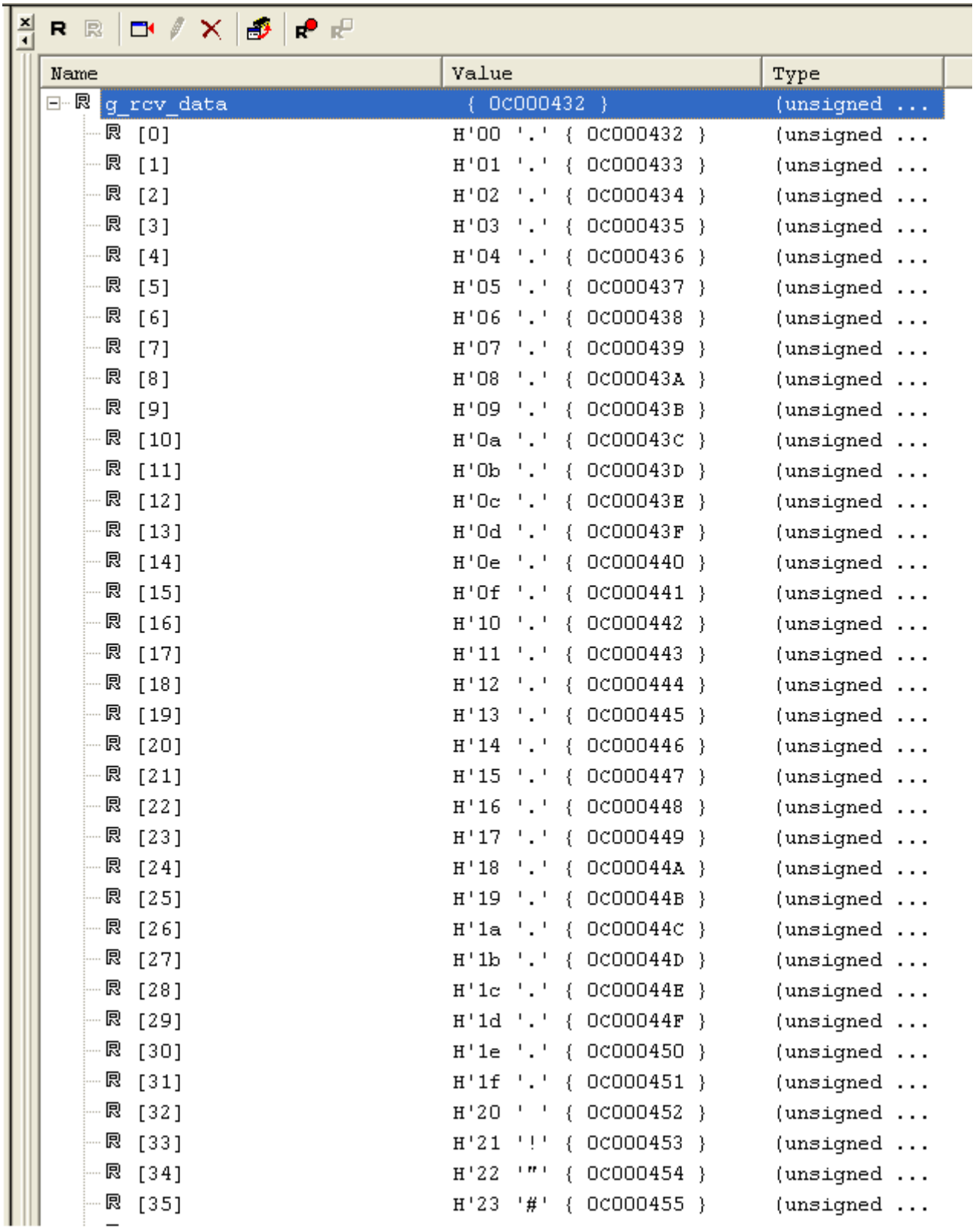
上記サンプルプログラムの送受信処理の実行結果については、シリアルクロック出力側 SH7730 とシリアルクロック入力側 SH7730 の両方で、送信データと受信期待データの一致を比較処理するループを抜けているため、SCIF0 にて全二重通信できていることが確認できます。

また、図 17のように、シリアルクロック出力側 SH7730 の SCIF0 受信データ格納領域 (g_rcv_data[D_SCIF_DATA_SIZE]) を High-performance Embedded Workshop で表示すると、シリアルクロック入力側 SH7730 から送信されたデータ（0x23～0x00) が正しく受信できていることが確認できます。

図 18のように、シリアルクロック入力側 SH7730 の SCIF0 受信データ格納領域 (g_rcv_data[D_SCIF_DATA_SIZE]) を High-performance Embedded Workshop で表示すると、シリアルクロック出力側 SH7730 から送信されたデータ (0x00～0x23) が正しく受信できていることが確認できます。

Name	Value	Type
g_rcv_data	"#"!	(unsigned ...
[0]	H'23 '#' { 0C000432 }	(unsigned ...
[1]	H'22 '"' { 0C000433 }	(unsigned ...
[2]	H'21 '!' { 0C000434 }	(unsigned ...
[3]	H'20 ' ' { 0C000435 }	(unsigned ...
[4]	H'1f '.' { 0C000436 }	(unsigned ...
[5]	H'1e '.' { 0C000437 }	(unsigned ...
[6]	H'1d '.' { 0C000438 }	(unsigned ...
[7]	H'1c '.' { 0C000439 }	(unsigned ...
[8]	H'1b '.' { 0C00043A }	(unsigned ...
[9]	H'1a '.' { 0C00043B }	(unsigned ...
[10]	H'19 '.' { 0C00043C }	(unsigned ...
[11]	H'18 '.' { 0C00043D }	(unsigned ...
[12]	H'17 '.' { 0C00043E }	(unsigned ...
[13]	H'16 '.' { 0C00043F }	(unsigned ...
[14]	H'15 '.' { 0C000440 }	(unsigned ...
[15]	H'14 '.' { 0C000441 }	(unsigned ...
[16]	H'13 '.' { 0C000442 }	(unsigned ...
[17]	H'12 '.' { 0C000443 }	(unsigned ...
[18]	H'11 '.' { 0C000444 }	(unsigned ...
[19]	H'10 '.' { 0C000445 }	(unsigned ...
[20]	H'0f '.' { 0C000446 }	(unsigned ...
[21]	H'0e '.' { 0C000447 }	(unsigned ...
[22]	H'0d '.' { 0C000448 }	(unsigned ...
[23]	H'0c '.' { 0C000449 }	(unsigned ...
[24]	H'0b '.' { 0C00044A }	(unsigned ...
[25]	H'0a '.' { 0C00044B }	(unsigned ...
[26]	H'09 '.' { 0C00044C }	(unsigned ...
[27]	H'08 '.' { 0C00044D }	(unsigned ...
[28]	H'07 '.' { 0C00044E }	(unsigned ...
[29]	H'06 '.' { 0C00044F }	(unsigned ...
[30]	H'05 '.' { 0C000450 }	(unsigned ...
[31]	H'04 '.' { 0C000451 }	(unsigned ...
[32]	H'03 '.' { 0C000452 }	(unsigned ...
[33]	H'02 '.' { 0C000453 }	(unsigned ...
[34]	H'01 '.' { 0C000454 }	(unsigned ...
[35]	H'00 '.' { 0C000455 }	(unsigned ...

図17　シリアルクロック出力側 SCIF0 受信データ格納領域



Name	Value	Type
g_rcv_data	{ 0C000432 }	(unsigned ...
[0]	H'00 '.' { 0C000432 }	(unsigned ...
[1]	H'01 '.' { 0C000433 }	(unsigned ...
[2]	H'02 '.' { 0C000434 }	(unsigned ...
[3]	H'03 '.' { 0C000435 }	(unsigned ...
[4]	H'04 '.' { 0C000436 }	(unsigned ...
[5]	H'05 '.' { 0C000437 }	(unsigned ...
[6]	H'06 '.' { 0C000438 }	(unsigned ...
[7]	H'07 '.' { 0C000439 }	(unsigned ...
[8]	H'08 '.' { 0C00043A }	(unsigned ...
[9]	H'09 '.' { 0C00043B }	(unsigned ...
[10]	H'0a '.' { 0C00043C }	(unsigned ...
[11]	H'0b '.' { 0C00043D }	(unsigned ...
[12]	H'0c '.' { 0C00043E }	(unsigned ...
[13]	H'0d '.' { 0C00043F }	(unsigned ...
[14]	H'0e '.' { 0C000440 }	(unsigned ...
[15]	H'0f '.' { 0C000441 }	(unsigned ...
[16]	H'10 '.' { 0C000442 }	(unsigned ...
[17]	H'11 '.' { 0C000443 }	(unsigned ...
[18]	H'12 '.' { 0C000444 }	(unsigned ...
[19]	H'13 '.' { 0C000445 }	(unsigned ...
[20]	H'14 '.' { 0C000446 }	(unsigned ...
[21]	H'15 '.' { 0C000447 }	(unsigned ...
[22]	H'16 '.' { 0C000448 }	(unsigned ...
[23]	H'17 '.' { 0C000449 }	(unsigned ...
[24]	H'18 '.' { 0C00044A }	(unsigned ...
[25]	H'19 '.' { 0C00044B }	(unsigned ...
[26]	H'1a '.' { 0C00044C }	(unsigned ...
[27]	H'1b '.' { 0C00044D }	(unsigned ...
[28]	H'1c '.' { 0C00044E }	(unsigned ...
[29]	H'1d '.' { 0C00044F }	(unsigned ...
[30]	H'1e '.' { 0C000450 }	(unsigned ...
[31]	H'1f '.' { 0C000451 }	(unsigned ...
[32]	H'20 ' ' { 0C000452 }	(unsigned ...
[33]	H'21 '!' { 0C000453 }	(unsigned ...
[34]	H'22 '"' { 0C000454 }	(unsigned ...
[35]	H'23 '#' { 0C000455 }	(unsigned ...

図18　シリアルクロック入力側 SCIF0 受信データ格納領域

6. 参考ドキュメント

- ソフトウェアマニュアル
  SH-4A ソフトウェアマニュアル (RJJ09B0090)
  (最新版をルネサス テクノロジホームページから入手してください。)
- ハードウェアマニュアル
  SH7730 グループ ハードウェアマニュアル (RJJ09B0339)
  (最新版をルネサス テクノロジ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 テクノロジホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/inquiry
csc@renesas.com

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2009.09.24	—	初版発行

本資料ご利用に際しての留意事項

１．　本資料は、お客様に用途に応じた適切な弊社製品をご購入いただくための参考資料であり、本資料中に記載の技術情報について弊社または第三者の知的財産権その他の権利の実施、使用を許諾または保証するものではありません。
２．　本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例など全ての情報の使用に起因する損害、第三者の知的財産権その他の権利に対する侵害に関し、弊社は責任を負いません。
３．　本資料に記載の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、あるいはその他軍事用途の目的で使用しないでください。また、輸出に際しては、「外国為替および外国貿易法」その他輸出関連法令を遵守し、それらの定めるところにより必要な手続を行ってください。
４．　本資料に記載の製品データ、図、表、プログラム、アルゴリズムその他応用回路例などの全ての情報は本資料発行時点のものであり、弊社は本資料に記載した製品または仕様等を予告なしに変更することがあります。弊社の半導体製品のご購入およびご使用に当たりましては、事前に弊社営業窓口で最新の情報をご確認いただきますとともに、弊社ホームページ(http://www.renesas.com)などを通じて公開される情報に常にご注意ください。
５．　本資料に記載した情報は、正確を期すため慎重に制作したものですが、万一本資料の記述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、弊社はその責任を負いません。
６．　本資料に記載の製品データ、図、表などに示す技術的な内容、プログラム、アルゴリズムその他応用回路例などの情報を流用する場合は、流用する情報を単独で評価するだけでなく、システム全体で十分に評価し、お客様の責任において適用可否を判断してください。弊社は、適用可否に対する責任は負いません。
７．　本資料に記載された製品は、各種安全装置や運輸・交通用、医療用、燃焼制御用、航空宇宙用、原子力、海底中継用の機器・システムなど、その故障や誤動作が直接人命を脅かしあるいは人体に危害を及ぼすおそれのあるような機器・システムや特に高度な品質・信頼性が要求される機器・システムでの使用を意図して設計、製造されたものではありません（弊社が自動車用と指定する製品を自動車に使用する場合を除きます）。これらの用途に利用されることをご検討の際には、必ず事前に弊社営業窓口へご照会ください。なお、上記用途に使用されたことにより発生した損害等について弊社はその責任を負いかねますのでご了承願います。
８．　第７項にかかわらず、本資料に記載された製品は、下記の用途には使用しないでください。これらの用途に使用されたことにより発生した損害等につきましては、弊社は一切の責任を負いません。
　　１）生命維持装置。
　　２）人体に埋め込み使用するもの。
　　３）治療行為（患部切り出し、薬剤投与等）を行うもの。
　　４）その他、直接人命に影響を与えるもの。
９．　本資料に記載された製品のご使用につき、特に最大定格、動作電源電圧範囲、放熱特性、実装条件およびその他諸条件につきましては、弊社保証範囲内でご使用ください。弊社保証値を越えて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、弊社はその責任を負いません。
１０．弊社は製品の品質および信頼性の向上に努めておりますが、特に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。弊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、社会的損害など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、誤動作防止設計などの安全設計（含むハードウェアおよびソフトウェア）およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
１１．本資料に記載の製品は、これを搭載した製品から剥がれた場合、幼児が口に入れて誤飲する等の事故の危険性があります。お客様の製品への実装後に容易に本製品が剥がれることがなきよう、お客様の責任において十分な安全設計をお願いします。お客様の製品から剥がれた場合の事故につきましては、弊社はその責任を負いません。
１２．本資料の全部または一部を弊社の文書による事前の承諾なしに転載または複製することを固くお断りいたします。
１３．本資料に関する詳細についてのお問い合わせ、その他お気付きの点等がございましたら弊社営業窓口までご照会ください。

D039444