RX610 グループ
SCI を使ったクロック同期式シングルマスタ制御ソフトウェア

R01AN0534JJ0100
Rev.1.00
2011.06.30

要旨

本アプリケーションノートでは、RX610 グループ シリアルコミュニケーションインタフェース（以下、SCI）のクロック同期式（3線式）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方法とサンプルコードの使用方法を説明します。

ポート制御による SPI 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。

なお、本サンプルコードは、スレーブデバイスとしての SPI デバイスを制御するための下位層に位置するソフトウェアです。

別途、スレーブデバイス制御のための上位層に位置するソフトウェアを用意していますので、入手してください。

対象デバイス

対応 MCU　　RX610 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 SPI Serial EEPROM

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。

目次

1. 仕様

RX610 グループの SCI のクロック同期式（３線式）シリアル通信を使用し、クロック同期式制御を行います。ポート制御による SPI 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 RX610 とし、SCI 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です。
- MCU 内蔵のクロック同期式（３線式）シリアル通信機能を使用します。また、ユーザ設定した 1 チャネルの使用が可能です。複数チャネルの使用は、できません。
- 本サンプルコードは、チップセレクト制御をサポートしていません。SPI デバイスを制御する場合、別途、デバイスセレクト制御を組み込む必要があります。
- 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です。
- MSB ファースト転送をサポートしています。
- CPU 転送のみをサポートしています。DMAC 転送をサポートしていません。
- 割り込みによる転送起動をサポートしていません。

表 1-1 使用する周辺機器と用途

周辺機器	用途
SCI	**クロック同期式（３線式）シリアル** **1ch（必須）**
Port	SPI **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、本サンプルコードでは、扱いません。**

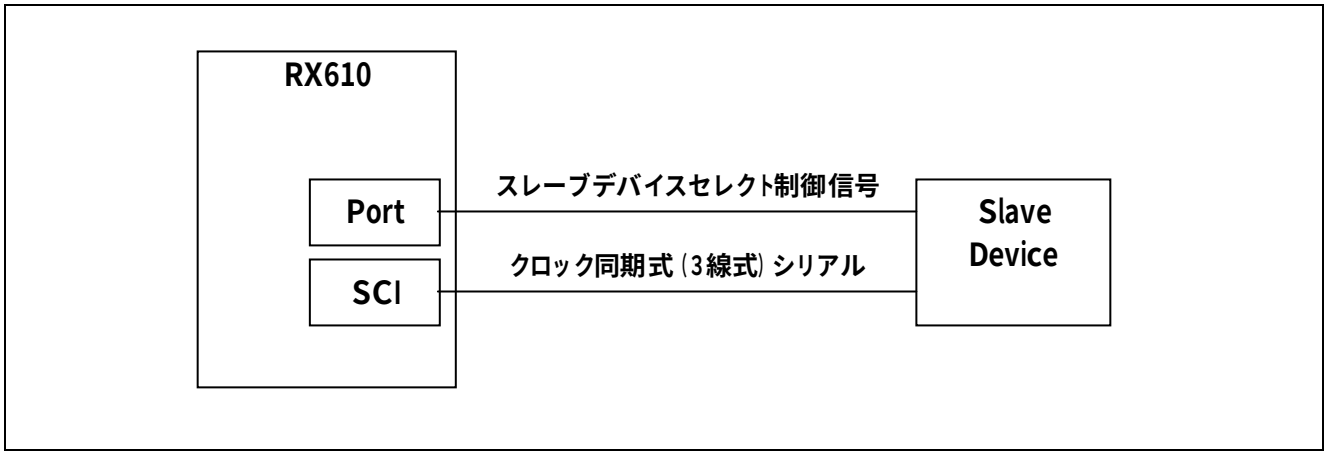


図 1-1 使用例

2. 動作確認条件

本アプリケーションノートのサンプルコードは、以下の動作条件で動作を確認しています。

表 2-1 動作確認条件

項目	内容
評価に使用したマイコン	RX610 グループ（プログラム ROM 2MB/RAM 128KB）
評価に使用したメモリ	ルネサス エレクトロニクス製 R1EX25XXX シリーズ SPI Serial EEPROM
動作周波数	ICLK：100MHz、PCLK：50MHz
動作電圧	3.3V
総合開発環境	ルネサス エレクトロニクス製 High-performance embedded Workshop Version 4.07.00.007
C コンパイラ	ルネサス エレクトロニクス製 RX ファミリ用 C/C++コンパイラケッケージ （ツールチェーン 1.0.0.0）
	コンパイルオプション 総合開発環境のデフォルト設定を使用しています。
エンディアン	ビッグエンディアン／リトルエンディアン
サンプルコードのバージョン	Ver.2.00
評価に使用したソフトウェア	ルネサス エレクトロニクス製 R1EX25xxx シリーズの SPI Serial EEPROM 制御ソフトウェア Ver.2.00
評価に使用したボード	Renesas Starter Kit for RX610

3. 関連アプリケーションノート

本アプリケーションノートに関連するアプリケーションノートを以下に示します。合わせて参照してください。

- Renesas R1EX25xxx シリーズ Serial EEPROM 制御ソフトウェア(R10AN0565JJ)

4. ハードウェア説明

4.1 使用端子一覧

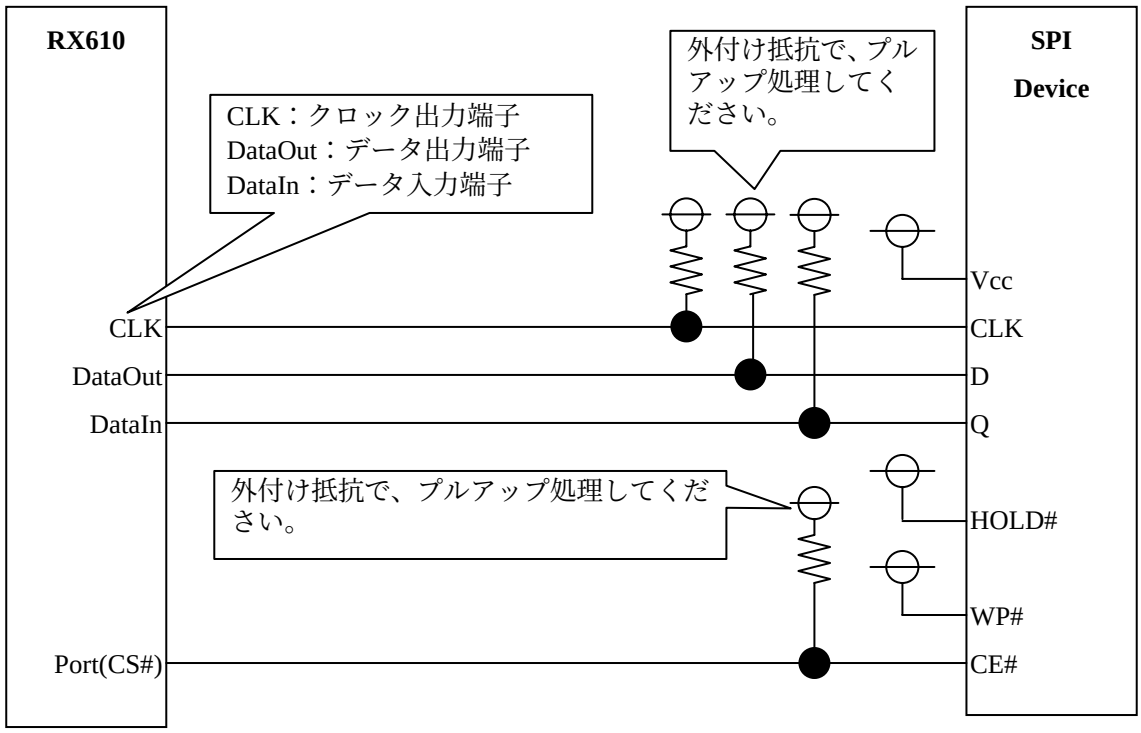
表 4-1に、使用端子と機能を示します。

表 4-1 使用端子と機能

端子名	入出力	内容
SCK （図 4-1の CLK）	出力	クロック出力
TxD （図 4-1の DataOut）	出力	マスタデータ出力
RxD （図 4-1の DataIn）	入力	マスタデータ入力
Port （図 4-1の Port(CS#)）	出力	スレーブデバイスセレクト出力 ただし、本サンプルコードでは、扱いません。

4.2 参考回路

図 4-1に接続図を示します。



・MCU 上のシリアル I/O に使用される端子名は、MCU に依存します。
・本アプリケーションノートでは、サンプルコードの記述に合わせて、CLK 端子、DataIn 端子、DataOut 端子、Port(CS#)端子として、表します。

図 4-1 RX610 SCI と SPI スレーブデバイスの接続例

5. ソフトウェア説明

5.1 動作説明

SCI のクロック同期式（３線式）シリアル通信機能を使って、クロック同期式シングルマスタ制御を実現します。

本サンプルコードでは、以下の制御を行っています。

- データの入出力を、クロック同期式モード（内部クロック使用）で、制御する。

本サンプルコードは、以下のように、デバイス上のデータのバイトオフセット値と、転送元／先のメモリのバイトオフセット値が合致するようにしたものです。

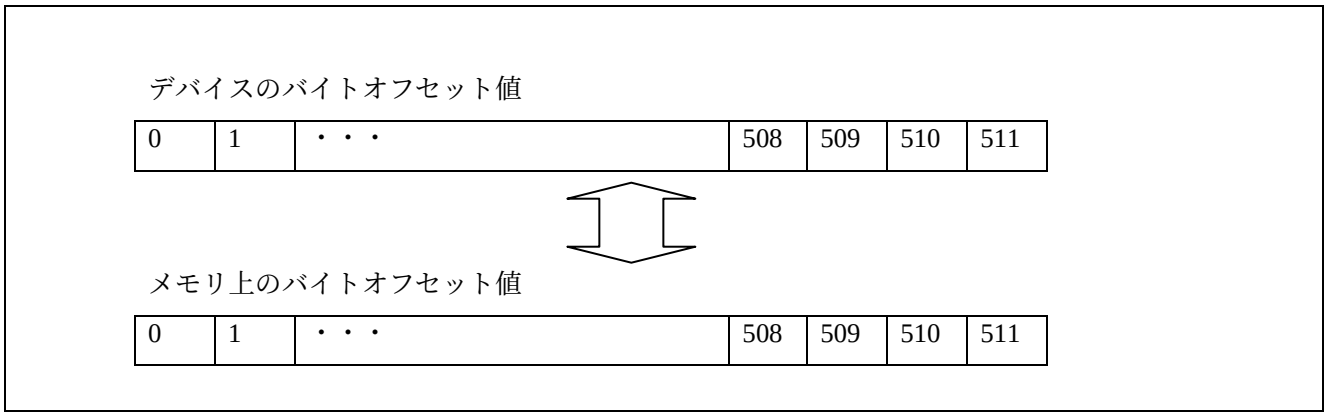
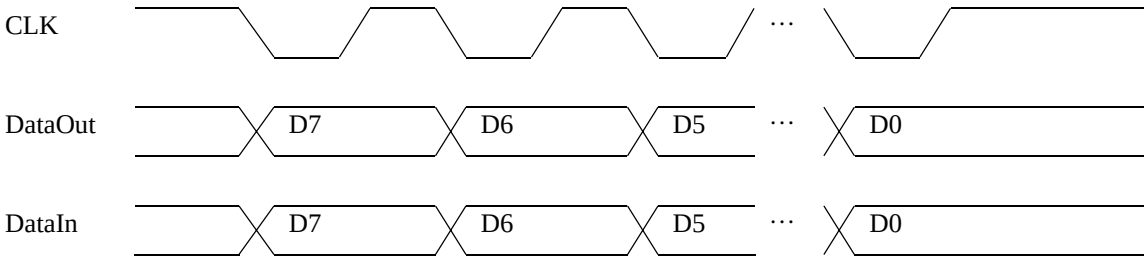


図 5-1 転送データの格納

5.1.1 クロック同期式モードで発生させるタイミング

SPI スレーブデバイス制御のため、図 5-2に示す SPI モード 3（CPOL=1、CPHA=1）のタイミングを発生します。



・MCU-> SPI 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

・SPI 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 CLK 端子のレベルは、”H”です。

図 5-2 クロック同期式モード タイミング設定

使用可能なシリアルクロック周波数は、MCU および SPI スレーブデバイスのデータシートで、確認してください。

5.1.2 SPI スレーブデバイスの CE#端子制御

本サンプルコードでは、SPI スレーブデバイスの CE#端子の制御をしておりません。SPI デバイスを制御する場合、別途、SPI 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、制御させることを推奨します。

また、SPI デバイスの CE#（MCU の Port(CS#)）信号の立ち下がりから、SPI デバイスの CLK（MCU の CLK）信号の立ち下がりまでの時間（SPI 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、SPI デバイスの CLK（MCU の CLK）信号の立ち上がりから、SPI デバイスの CE#（MCU の Port(CS#)）信号の立ち上がりまでの時間時間（SPI デバイスの CE#ホールド時間）を設けてください。

SPI 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。

5.2 ソフトウェア制御概要

5.2.1 ソフトウェア構成

本サンプルコードは、スレーブデバイスとしての SPI デバイスを制御するための下位層に位置するソフトウェアです。

本サンプルコードでは、SPI スレーブデバイスの CE#端子制御無しの SPI モード 3（CPOL=1、CPHA=1）を使った制御を実現しています。

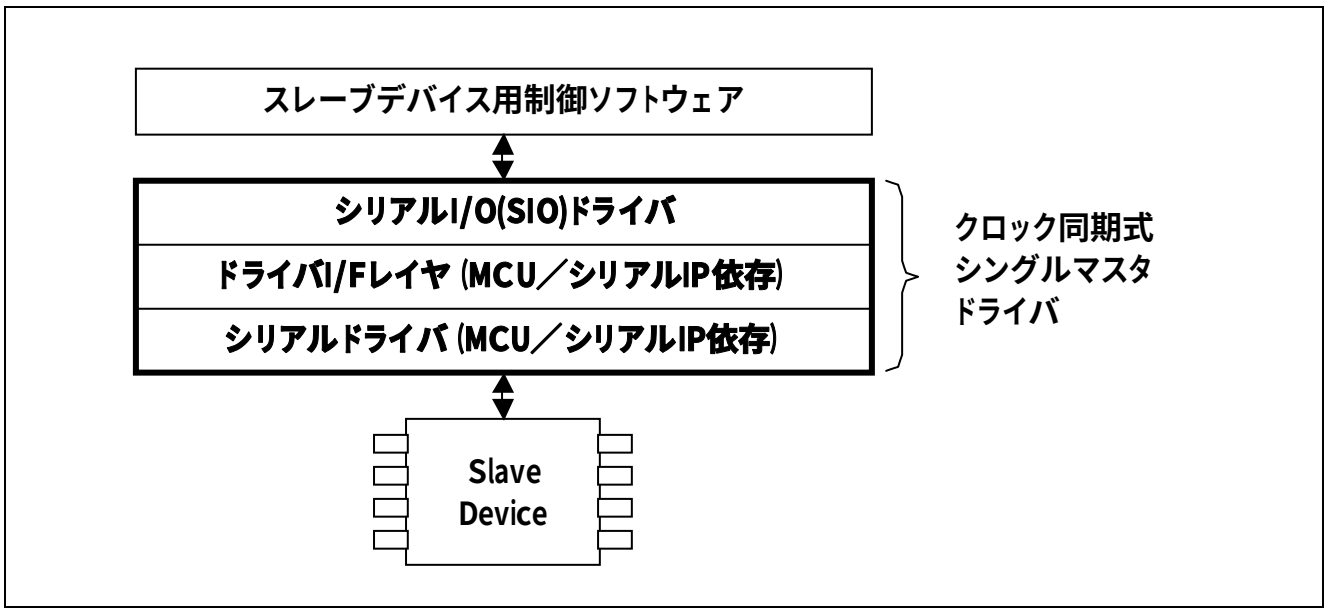


図 5-3 ソフトウェア構成

以下の送信／受信の動作を実現しています。

①クロック同期式シングルマスタソフトウェアを使ったデータ送信

②クロック同期式シングルマスタソフトウェアを使ったデータ受信

本サンプルコードは、以下の 5 つの基本処理で構成しています。

- シリアル許可 －DataIn 端子のポート入力化、DataOut 端子と CLK 端子のポート”H”出力
  シリアル I/O 有効化、ビットレート設定
- シリアル禁止 －シリアル I/O 無効化、DataIn 端子のポート入力化、
  DataOut 端子と CLK 端子のポート”H”出力化
- シリアル開放 －シリアル I/O 無効化、DataIn 端子のポート入力化、
  DataOut 端子と CLK 端子のポート入力化
- データ送信 －SPI デバイスへのデータ送信処理
- データ受信 －SPI デバイスからのデータ受信処理

5.2.2 シリアル許可（R_SIO_Enable()）

シリアル I/O で使用する DataIn 端子をポート入力、DataOut 端子と CLK 端子をポート”H”出力にします。

その後、シリアル I/O 機能を有効にし、DataIn 端子をデータ入力、DataOut 端子をデータ出力、CLK 端子をクロック出力に切り替えます。

シリアル I/O で使用する通信速度（ビットレート）を設定します。

5.2.3 シリアル禁止（R_SIO_Disable()）

シリアル I/O で使用する端子をポートに切り替えて、DataIn 端子をポート入力、DataOut 端子と CLK 端子をポート”H”出力にします。

5.2.4 シリアル開放（R_SIO_Open_Port()）

シリアル I/O で使用する端子をポートに切り替えて、DataIn 端子と DataOut 端子と CLK 端子をポート入力にします。

5.2.5 データ送信（R_SIO_Tx_Data ()）

シリアル I/O を使って、データを送信します。

送信設定にて、送信します。

5.2.6 データ受信（R_SIO_Rx_Data ()）

シリアル I/O を使って、データを受信します。

送受信設定にて、受信します。

5.3 必要メモリサイズ

表 5-1に必要とするメモリサイズを示します。

表 5-1 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	838 バイト（リトルエンディアン）	R_SIO_sci_rx.c
RAM	0 バイト（リトルエンディアン）	R_SIO_sci_rx.c
最大使用ユーザスタック	168 バイト	
最大使用割り込みスタック	－	

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。

5.4 ファイル構成

表 5-2に、サンプルコードで使用するファイルを示します。なお、統合開発環境で自動生成するファイルを除きます。

表 5-2 ファイル構成

¥an_r01an0534jj_rx610 <DIR>			**サンプルコードのフォルダ**
r01an0534jj0100_rx610.pdf			**アプリケーションノート**
¥ r01an0534jj_rx610_src <DIR>			**プログラム格納用フォルダ**
	¥com <DIR>		**共通関数格納用フォルダ**
	（注１）	mtl_com.c	**共通関数の各種定義**
		mtl_com.h.common	**共通ヘッダファイル**
		mtl_com.h.RX600	**共通関数のヘッダファイル**
		mtl_endi.c	**共通ファイル（エンディアン設定関連）**
		mtl_mem.c	**共通ファイル（標準ライブラリ関数）**
		mtl_os.c mtl_os.h	**共通ファイル（標準ライブラリ関数）**
		mtl_str.c	**共通ファイル（標準ライブラリ関数）**
		mtl_tim.c mtl_tim.h	**共通ファイル（ループタイマ関連）**
		mtl_tim.h.sample	**ループタイマの設定値サンプル**
	¥r_sio_sci_rx <DIR>		RX610 SCI **用クロック同期式シングルマスタ制御ソフトウェアのフォルダ**
		R_SIO.h	**ヘッダファイル**
		R_SIO_sci.h.rx610	I/F **モジュール共通定義**
		R_SIO_sci_rx.c	I/F **モジュール**

注１． com **フォルダに含まれるファイルは、スレーブデバイス用制御ソフトウェアでも使用するものです。最新のものを使用してください。**

5.5 定数一覧

5.5.1 戻り値

表 5-3に、サンプルコードで使用する戻り値を示します。

表 5-3 戻り値

定数名	設定値	内容
SIO_OK	(error_t)(0)	Successful Operation
SIO_ERR_PARAM	(error_t)(-1)	Parameter Error
SIO_ERR_HARD	(error_t)(-2)	Hardware Error
SIO_ERR_OTHER	(error_t)(-7)	Other Error

5.5.2 各種定義

表 5-4に、サンプルコードで使用する各種定義した値を示します。

表 5-4 各種定義値

定数名	設定値	内容
SIO_LOG_ERR	1	Log type : Error
SIO_TRUE	(uint8_t)0x01	Flag "ON"
SIO_FALSE	(uint8_t)0x00	Flag "OFF"
SIO_HI	(uint8_t)0x01	Port "H"
SIO_LOW	(uint8_t)0x00	Port "L"
SIO_OUT	(uint8_t)0x01	Port output setting
SIO_IN	(uint8_t)0x00	Port input setting
SIO_TX_WAIT	(uint16_t)50000	SIO transmission completion waiting time 50000* 1us = 50ms
SIO_RX_WAIT	(uint16_t)50000	SIO receive completion waiting time 50000* 1us = 50ms
SIO_DMA_TX_WAIT	(uint16_t)50000	DMA transmission completion waiting time 50000* 1us = 50ms
SIO_DMA_RX_WAIT	(uint16_t)50000	DMA receive completion waiting time 50000* 1us = 50ms
SIO_T_SIO_WAIT	(uint16_t)MTL_T_1US	SIO transmit&receive completion waiting polling time
SIO_T_DMA_WAIT	(uint16_t)MTL_T_1US	DMA transmit&receive completion waiting polling time
SIO_T_BRR_WAIT	(uint16_t)MTL_T_10US	BRR setting wait time

5.6 構造体／共用体一覧

以下に、サンプルコードで使用する構造体を示します。

```
/* uint32_t <-> uint8_t conversion */
typedef union {
  uint32_t   ul;
  uint8_t    uc[4];
} SIO_EXCHG_LONG;                          /* total 4byte                */


/* uint16_t <-> uint8_t conversion */
typedef union {
  uint16_t   us;
  uint8_t    uc[2];
} SIO_EXCHG_SHORT;                         /* total 2byte                */
```

5.7 関数一覧

表 5-5に関数一覧を示します。

表 5-5 関数一覧

関数名	説明
R_SIO_Init_Driver()	ドライバ初期化処理
R_SIO_Disable()	シリアル I/O 禁止設定処理
R_SIO_Enable()	シリアル I/O 許可設定処理
R_SIO_Open_Port()	シリアル I/O 開放設定処理
R_SIO_Tx_Data()	シリアル I/O データ送信処理
R_SIO_Rx_Data()	シリアル I/O データ受信処理

5.8 関数仕様

5.8.1 ドライバ初期化処理

R_SIO_Init_Driver	
概要	**ドライバ初期化処理**
ヘッダ	R_SIO.h, R_SIO_sci.h, mtl_com.h
宣言	error_t R_SIO_Init_Driver(void)
説明	**・ドライバの初期化を行います。シリアル** I/O **機能を無効化し、端子をポートに設定します。** **・システム起動時に一度だけ呼び出してください。** **・本関数コール前に、スレーブデバイスセレクト制御信号を**"H"**出力にしてください。**
引数	**なし**
リターン値	SIO_OK ; Successful operation
備考	**前の使用状態を考慮し、以下の処理を行います。** **・送信／受信を停止します。** **・**SSR **の** PER **フラグ／**FER **フラグ／**OERE **フラグをクリアします。**

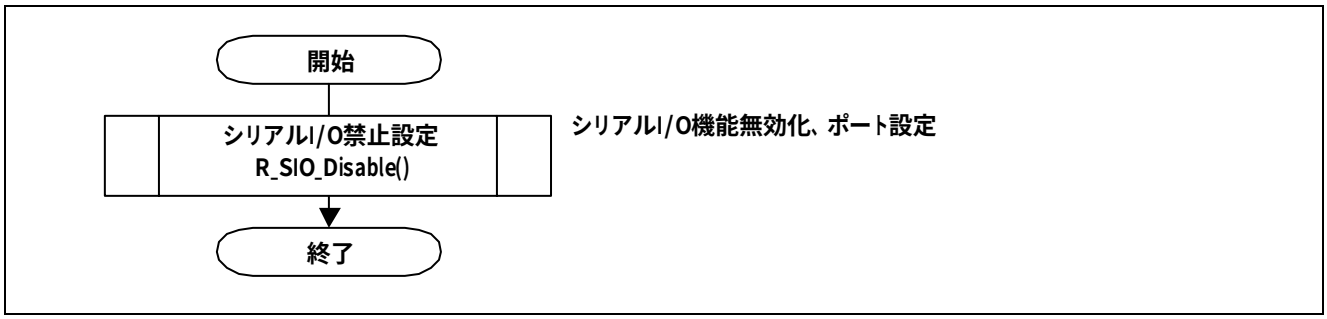


図 5-4 **ドライバ初期化処理概要**

5.8.2 シリアル I/O 禁止設定処理

R_SIO_Disable	
概要	シリアル I/O 禁止設定処理
ヘッダ	R_SIO.h, R_SIO_sci.h, mtl_com.h
宣言	error_t R_SIO_Disable(void)
説明	・シリアル I/O 機能を無効化し、端子をポートに設定します。 シリアル I/O を無効化します。 シリアル I/O で使用する端子をポート設定にします。 ・本関数コール前に、スレーブデバイスセレクト制御信号を"H"出力にしてください。
引数	なし
リターン値	SIO_OK ; Successful operation
備考	・SMR と SCR に 00h を書き込み、初期化します。（ハードウェアマニュアル記載の初期化の SCR への 00h ライトの実行） ・送受信に備え、SSR の PER フラグ／FER フラグ／ORE の各フラグをリード後、0 クリアします。 ・使用するシリアル I/O をモジュールストップ状態に設定します。 ・使用しない場合、本関数をコールし、シリアル I/O 機能を無効化することができます。

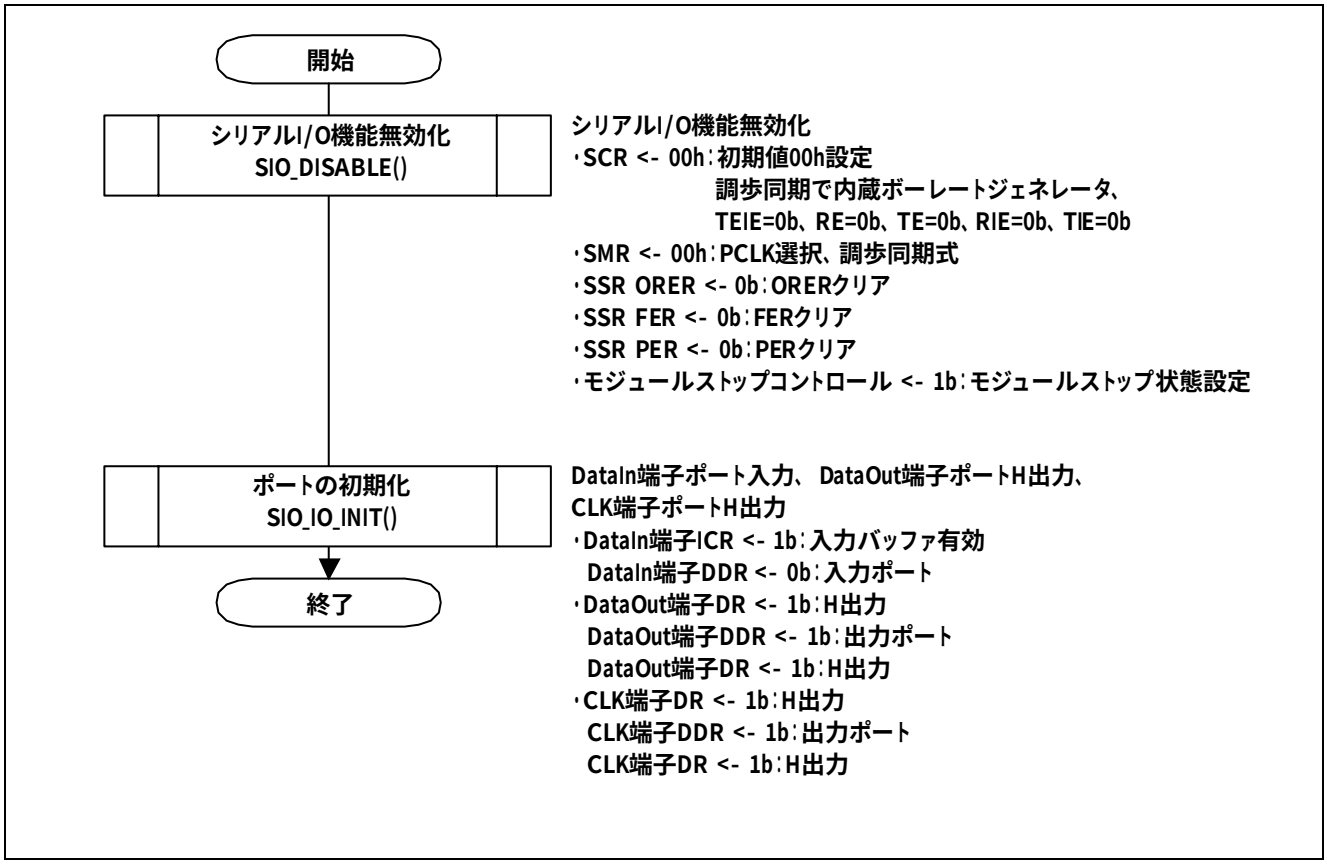


図 5-5 シリアル I/O 禁止設定処理概要

5.8.3 シリアル I/O 許可設定処理

R_SIO_Enable	
概要	シリアル I/O 許可設定処理
ヘッダ	R_SIO.h, R_SIO_sci.h, mtl_com.h
宣言	error_t R_SIO_Enable(uint8_t BrgData)
説明	・シリアル I/O 機能を有効化、ビットレートを設定します。 シリアル I/O で使用する端子をポート設定にします。 シリアル I/O を有効化し、ビットレートを設定します。 ・R_SIO_Disable()コール後に、本関数をコールしてください。 ・シリアル I/O データ送信処理とシリアル I/O データ受信処理実行前に、本関数をコールしてください。 ・ビットレートを変更したい場合、本関数を使用してください。事前に、シリアル I/O 禁止設定処理を実行してください。
引数	uint8_t BrgData ; ビットレート設定値
リターン値	SIO_OK ; Successful operation
備考	・使用するシリアル I/O をモジュールストップ解除状態に設定します。 ・ハードウェアマニュアル記載の初期化フローチャートにしたがって、以下を実行します。（R_SIO_Disable()コールにより、SCR に初期値 00h が書き込まれた状態を想定します。） ①SCR TIE=RIE=TE=RE=TEIE=0b（SCR に初期値 00h 設定済み） ②SCR CKE を設定（本関数にて） ③SMR を設定（本関数にて） ④SCMR を設定（本関数にて） ⑤BRR を設定（本関数にて） ⑥ソフトウェア･ウェイト 10μs：ビットレート≧0.1Mbps を想定（本関数にて） ウェイト時間が不足の場合、値を見直してください。

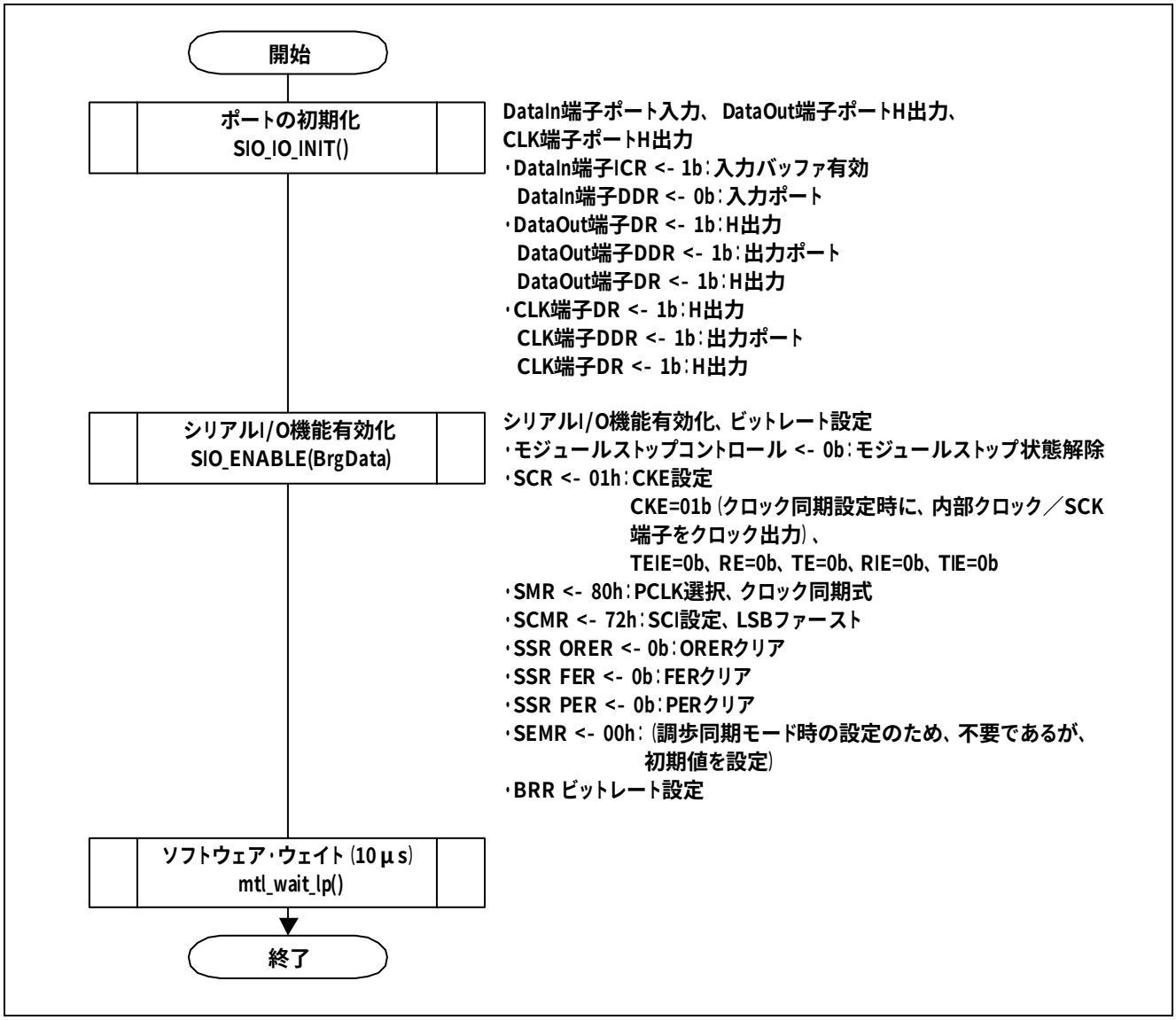


図 5-6 シリアル I/O 許可設定処理概要

5.8.4 シリアル I/O 開放設定処理

R_SIO_Open_Port	
概要	SIO port(DataOut, DataIn, CLK 開放設定処理
ヘッダ	R_SIO.h, R_SIO_sci.h, mtl_com.h
宣言	error_t R_SIO_Open_Port(void)
説明	・シリアル I/O に使用する端子をオープン（入力状態）にします。 ・本関数コール前に、スレーブデバイスセレクト制御信号を”H”出力にしてください。
引数	なし
リターン値	SIO_OK ; Successful operation
備考	・リムーバブルメディアの挿抜目的で、用意した関数です。リムーバブルメディアの挿入前、およびリムーバブルメディアの抜去前に、本関数を使用してください。リムーバブルメディアの抜去前には、シリアル I/O 禁止設定処理を実行してください。

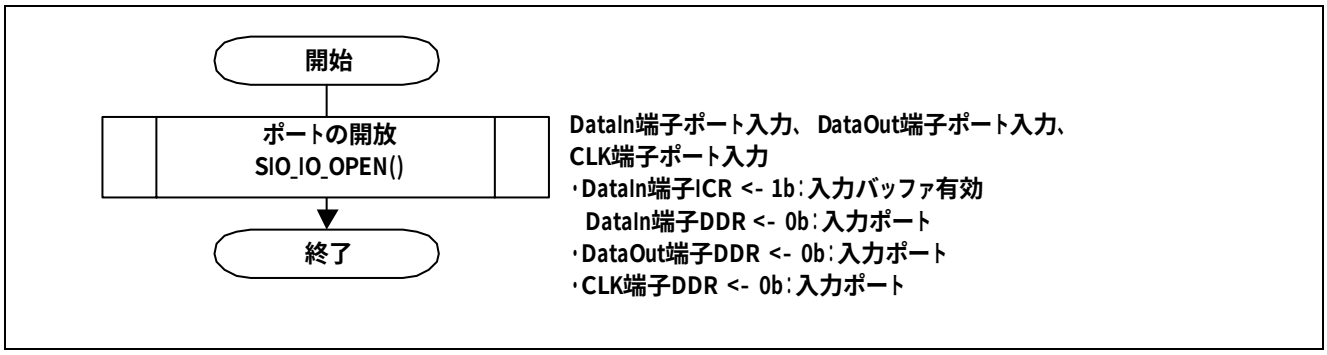


図 5-7 シリアル I/O 開放設定処理概要

5.8.5 シリアル I/O データ送信処理

R_SIO_Tx_Data

概要	**シリアル I/O データ送信処理**			
ヘッダ	R_SIO.h, R_SIO_sci.h, mtl_com.h			
宣言	error_t R_SIO_Tx_Data(uint16_t TxCnt, uint8_t FAR* pData)			
説明	**・pData のデータを指定バイト数分送信します。** **・本関数コール前に、シリアル I/O 許可設定処理を実行してください。** **・本関数の実行の結果、異常終了であれば、シリアル I/O 禁止設定処理を実行してください。**			
引数	uint16_t	TxCnt	;	**送信バイト数**
	uint8_t FAR*	pData	;	**送信データ格納バッファポインタ**
リターン値	SIO_OK		;	Successful operation
	SIO_ERR_HARD		;	Hardware error
備考	**・ハードウェアマニュアル記載の初期化フローチャートにしたがって、以下を実行します。** **①SCR TE,RE,TIE,RIE,TEIE の設定** **・送信完了後、ハードウェアマニュアル記載のシリアル送信のフローチャートにしたがって、**SCR TE=RE=TIE=RIE=TEIE=0b **に設定します。** **・継続使用しない場合、シリアル I/O 禁止設定処理を実行することを推奨します。**			

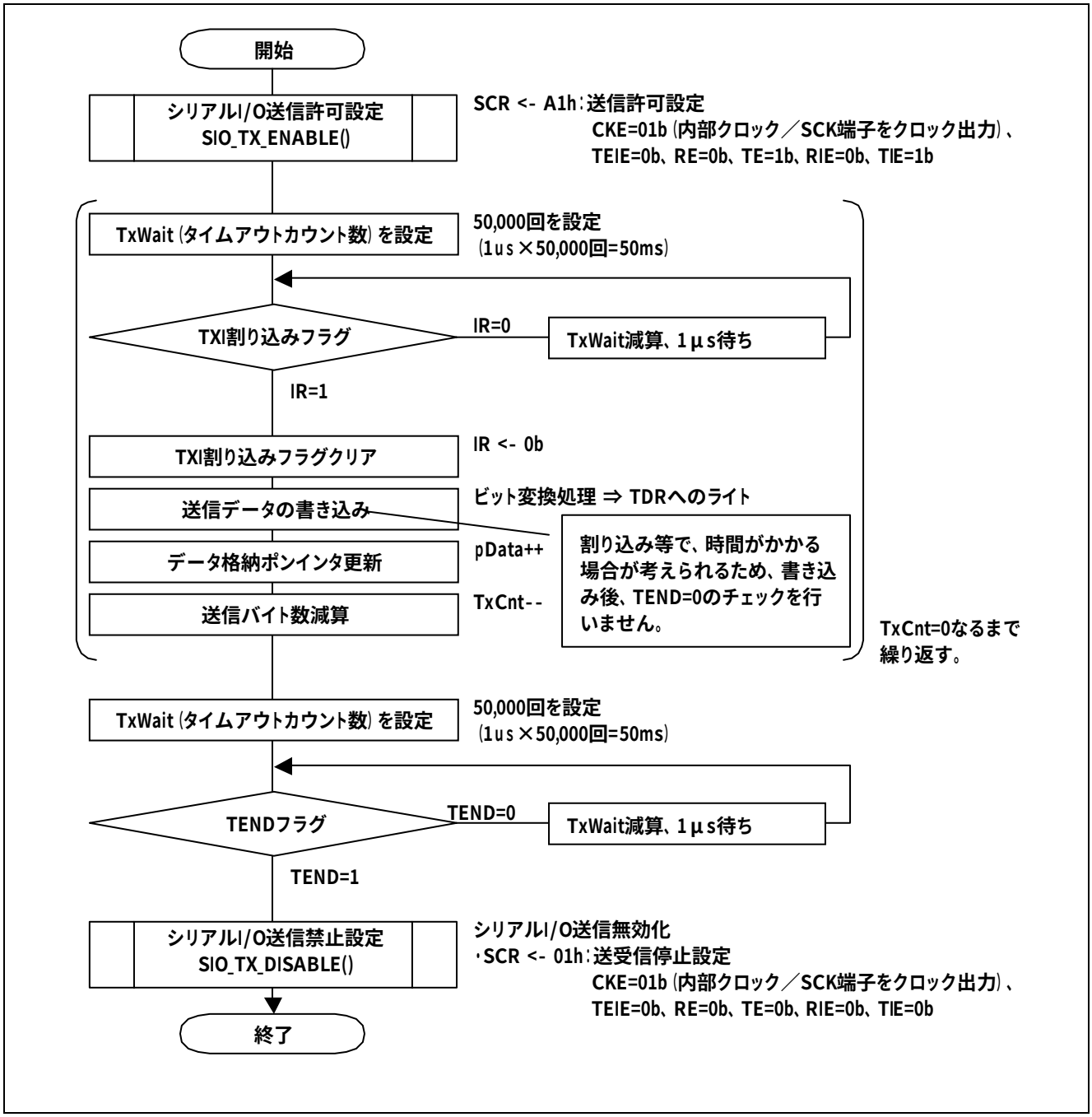


図 5-8 シリアル I/O データ送信処理概要

5.8.6 シリアル I/O データ受信処理

R_SIO_Rx_Data

概要	**シリアル I/O データ受信処理**			
ヘッダ	R_SIO.h, R_SIO_sci.h, mtl_com.h			
宣言	error_t R_SIO_Rx_Data(uint16_t RxCnt, uint8_t FAR* pData)			
説明	**・指定バイト数分でデータを受信し、pData に格納します。** **・本関数コール前に、シリアル I/O 許可設定処理を実行してください。** **・本関数の実行の結果、異常終了であれば、シリアル I/O 禁止設定処理を実行してください。**			
引数	uint16_t	RxCnt	;	**受信バイト数**
	uint8_t FAR*	pData	;	**受信データ格納バッファポインタ**
リターン値	SIO_OK		;	Successful operation
	SIO_ERR_HARD		;	Hardware error
備考	**・ハードウェアマニュアル記載の初期化フローチャートにしたがって、以下を実行します。** **①SCR TE,RE,TIE,RIE,TEIE の設定** **・受信完了後、ハードウェアマニュアル記載のシリアル送信のフローチャートにしたがって、SCR TE=RE=TIE=RIE=TEIE=0b に設定します。** **・1 バイトダミーデータ送信によるクロック発生させ、1 バイトマスタ受信動作させるため、オーバーランエラーは発生しません。したがって、フローチャート上、オーバーフロー確認処理を削除しています** **・継続使用しない場合、シリアル I/O 禁止設定処理を実行することを推奨します。**			

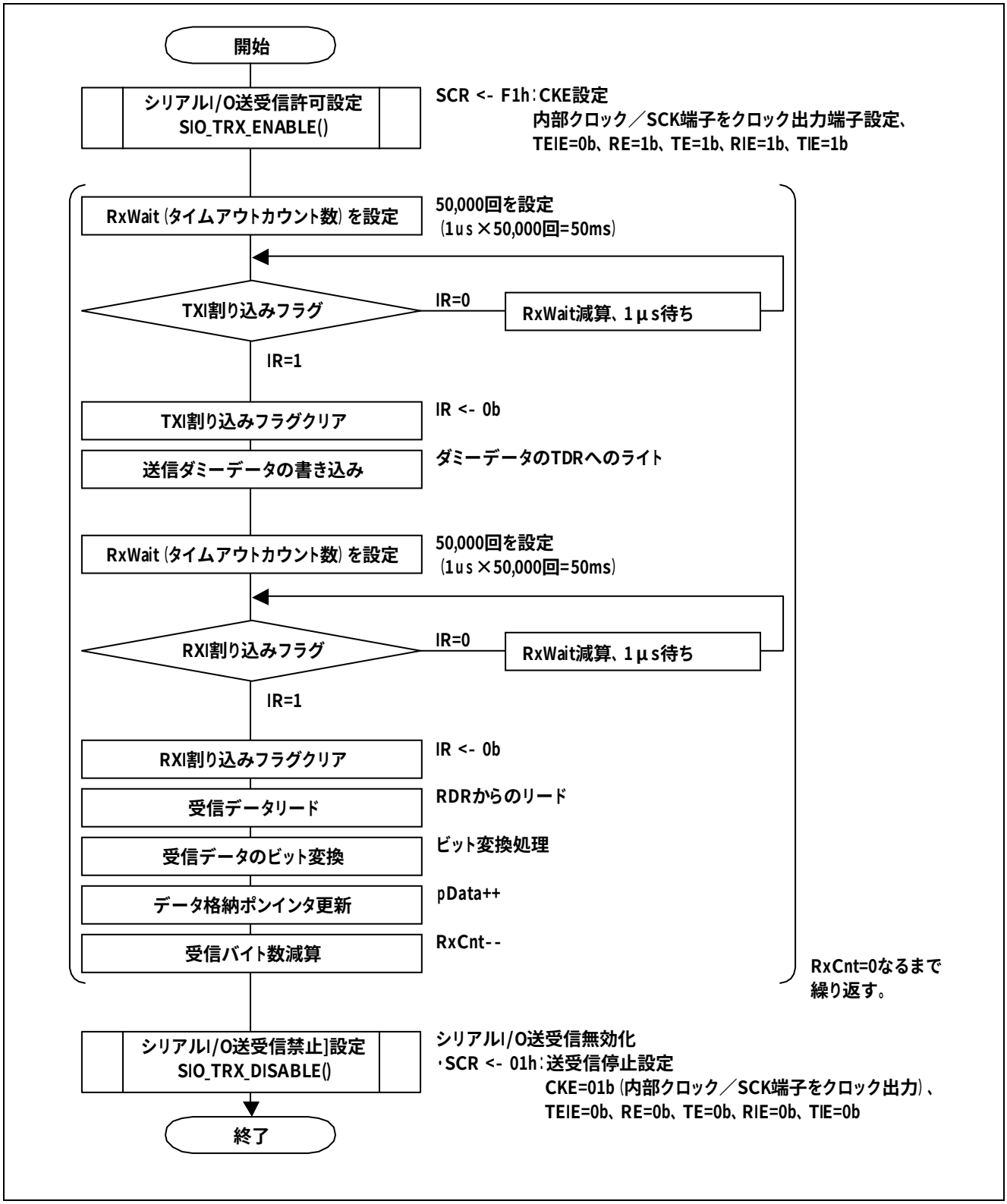


図 5-9 シリアル I/O データ受信処理概要

5.9 マクロ関数仕様

以下に、本サンプルコードで使用するマクロ関数を示します。

5.9.1 マクロ関数 SIO_IO_INIT()

(1) **目的**

入力端子をポート入力状態、出力端子をポート出力状態にします。

(2) **機能**

DataIn 端子をポート入力状態、DataOut 端子と CLK 端子をポート出力状態にします。

以下の処理を実現します。必要に応じて、処理を見直してください。

①DataIn 端子をポート入力に設定する。

②DataOut 端子をポート”H”出力に設定する。

③CLK 端子をポート”H”出力に設定する。

5.9.2 マクロ関数 SIO_IO_OPEN()

(1) **目的**

入力端子と出力端子をポート入力状態にします。

(2) **機能**

DataIn 端子と DataOut 端子と CLK 端子入力端子をポート入力状態にします。

以下の処理を実現します。必要に応じて、処理を見直してください。

①DataIn 端子をポート入力に設定する。

②DataOut 端子をポート入力に設定する。

③CLK 端子をポート入力に設定する。

(3) **備考**

リムーバブルメディアの挿入前および抜去前での、全端子の Hi-z 化をするために使用してください。

5.9.3 マクロ関数 SIO_DATAI_INIT()

(1) **目的**

DataIn 端子をポート入力状態にします。

(2) **機能**

以下の処理を実現します。必要に応じて、処理を見直してください。

①DataIn 端子をポート入力に設定する。

5.9.4 マクロ関数 SIO_DATAO_INIT()

(1) **目的**

DataOut 端子をポート”H”出力にします。

(2) **機能**

以下の処理を実現します。必要に応じて、処理を見直してください。

①DataOut 端子をポート”H”出力に設定する。

5.9.5 マクロ関数 SIO_DATAO_OPEN()

(1) **目的**

DataOut 端子をポート入力状態にします。

(2) **機能**

以下の処理を実現します。必要に応じて、処理を見直してください。

①DataOut 端子をポート入力に設定する。

5.9.6 マクロ関数 SIO_CLK_INIT()

(1) **目的**

CLK 端子をポート”H”出力にします。

(2) **機能**

以下の処理を実現します。必要に応じて、処理を見直してください。

①CLK 端子をポート”H”出力に設定する。

5.9.7 マクロ関数 SIO_CLK_OPEN()

(1) **目的**

CLK 端子をポート入力状態にします。

(2) **機能**

以下の処理を実現します。必要に応じて、処理を見直してください。

①CLK 端子をポート入力に設定する。

5.9.8 マクロ関数 SIO_ENABLE()

(1) **目的**

シリアル I/O を初期化し、機能を有効化します。ただし、送信許可／受信許可／送受信許可にするまでの共通処理を実行します。また、ビットレートを設定します。

(2) **機能**

ハードウェアマニュアにしたがって、シリアル I/O を初期化します。必要に応じて、処理を見直してください。

RX610 の場合、以下の処理を行います。

①モジュールストップコントロールレジスタを使って、モジュールストップ解除状態に設定します。

②送信設定と送受信設定の有効化手順において、共通の処理を行います。

送信設定と送受信設定の共通部分の以下を設定します。

・SCR.TIE, RIE, TE, RE, TEIE ビットを”0”にする

・SCR.CKE[1:0]ビットを設定

・SMR, SCMR に送信／受信フォーマットを設定

・SSR の ORER, FER, PER の各フラグを読み出した後、0 を書き込んでクリア、各フラグのクリア確認

・SEMR 設定

・BRR に値を書く

(3) **備考**

ビットレート設定後、待ちを必要とするシリアル I/O の場合、マクロ関数終了後、待ち処理を入れてください。

SIO_DISABLE()と対となるものです。本関数を実行した場合、SIO_DISABLE()を実行して、処理を終了してください。

5.9.9 マクロ関数 SIO_DISABLE()

(1) **目的**

シリアル I/O 機能を無効化します。

(2) **機能**

シリアル I/O を無効化します。送信設定／送受信設定の無効化手順において、共通の処理を行います。必要に応じて、処理を見直してください。

RX610 の場合、以下の処理を行います。

①SCR に初期値 00h を設定し、送信／受信を停止

②SMR に初期値 00h を設定

③SSR の ORER, FER, PER の各フラグを読み出した後、0 を書き込んでクリア、各フラグのクリア確認

④モジュールストップコントロールレジスタを使って、モジュールストップ状態に設定

(3) **備考**

SIO_ENABLE()と対となるものです。SIO_ENABLE()を実行した場合、本関数を実行して、処理を終了してください。

5.9.10 マクロ関数 SIO_TX_ENABLE()

(1) **目的**

シリアル I/O を送信許可にします。

(2) **機能**

ハードウェアマニュアにしたがって、シリアル I/O を送信許可設定します。端子をポート機能からシリアル I/O 機能への切り替え後、送信許可設定します。必要に応じて、処理を見直してください。

本処理では、SIO_ENABLE()の後の初期化手順から、送信設定専用の初期化処理を行います。

RX610 の場合、以下の処理を行います。

①送信許可を設定します。

SCR.TE=1b, TIE=1b を設定し、送信を許可します。

(3) **備考**

SIO_TX_DISABLE()と対となるものです。本関数を実行した後は、SIO_TX_DISABLE()を実行して、処理を終了してください。

5.9.11 マクロ関数 SIO_TX_DISABLE()

(1) **目的**

シリアル I/O の送信機能を停止します。

(2) **機能**

SIO_TX_ENABLE()の処理の逆手順により、送信機能を停止します。送信停止設定処理後、端子をシリアル I/O 機能からポート機能へ切り替えます。必要に応じて、処理を見直してください。

RX610 の場合、以下の処理を行います。

①送信設定の停止処理を行います。

SCR.TE=0b, RE=0b, TIE=0b, RIE=0b, TEIE=0b を設定し、送信を停止させます。

(3) **備考**

SIO_TX_ENABLE()と対となるものです。SIO_TX_ENABLE()を実行した後は、本関数を実行して、処理を終了してください。

5.9.12 マクロ関数 SIO_TRX_ENABLE()

(1) **目的**

シリアル I/O を送受信許可にします。

(2) **機能**

ハードウェアマニュアにしたがって、シリアル I/O を送受信許可設定します。端子をポート機能からシリアル I/O 機能への切り替え後、送受信許可設定します。必要に応じて、処理を見直してください。

本処理では、SIO_ENABLE()の後の初期化手順から、送受信設定専用の初期化処理を設定します。

RX610 の場合、以下の処理を行います。

①送受信許可を設定します。

SCR.TE=1b, RE=1b, TIE=1b, RIE=1b を設定し、送受信を許可します。

(3) **備考**

SIO_TRX_DISABLE()と対となるものです。本関数を実行した後は、SIO_TRX_DISABLE()を実行して、処理を終了してください。

5.9.13 マクロ関数 SIO_TRX_DISABLE()

(1) **目的**

シリアル I/O の送受信機能を停止します。

(2) **機能**

SIO_TRX_ENABLE()の処理の逆手順により、送受信機能を停止します。送受信停止設定処理後、端子をシリアル I/O 機能からポート機能への切り替えます。必要に応じて、処理を見直してください。

RX610 の場合、以下の処理を行います。

①送受信設定の停止処理を行います。

SCR.TE=0b, RE=0b, TIE=0b, RIE=0b, TEIE=0b を設定し、送受信を停止させます。

(3) **備考**

SIO_TRX_ENABLE()と対となるものです。SIO_TRX_ENABLE()を実行した後は、本関数を実行して、処理を終了してください。

5.9.14 マクロ関数 SIO_SSR_CLEAR()

(1) **目的**

SSR のエラーフラグをクリアします。

(2) **機能**

ORER フラグ、FER フラグ、PER フラグをクリアします。

RX610 の場合、フラグ毎に、以下の処理を行います。

①フラグ＝1 であれば、0 クリアする。

②フラグをリードし、0 であることを確認する。

5.10 状態遷移図

図 5-10に、状態遷移図を示します。

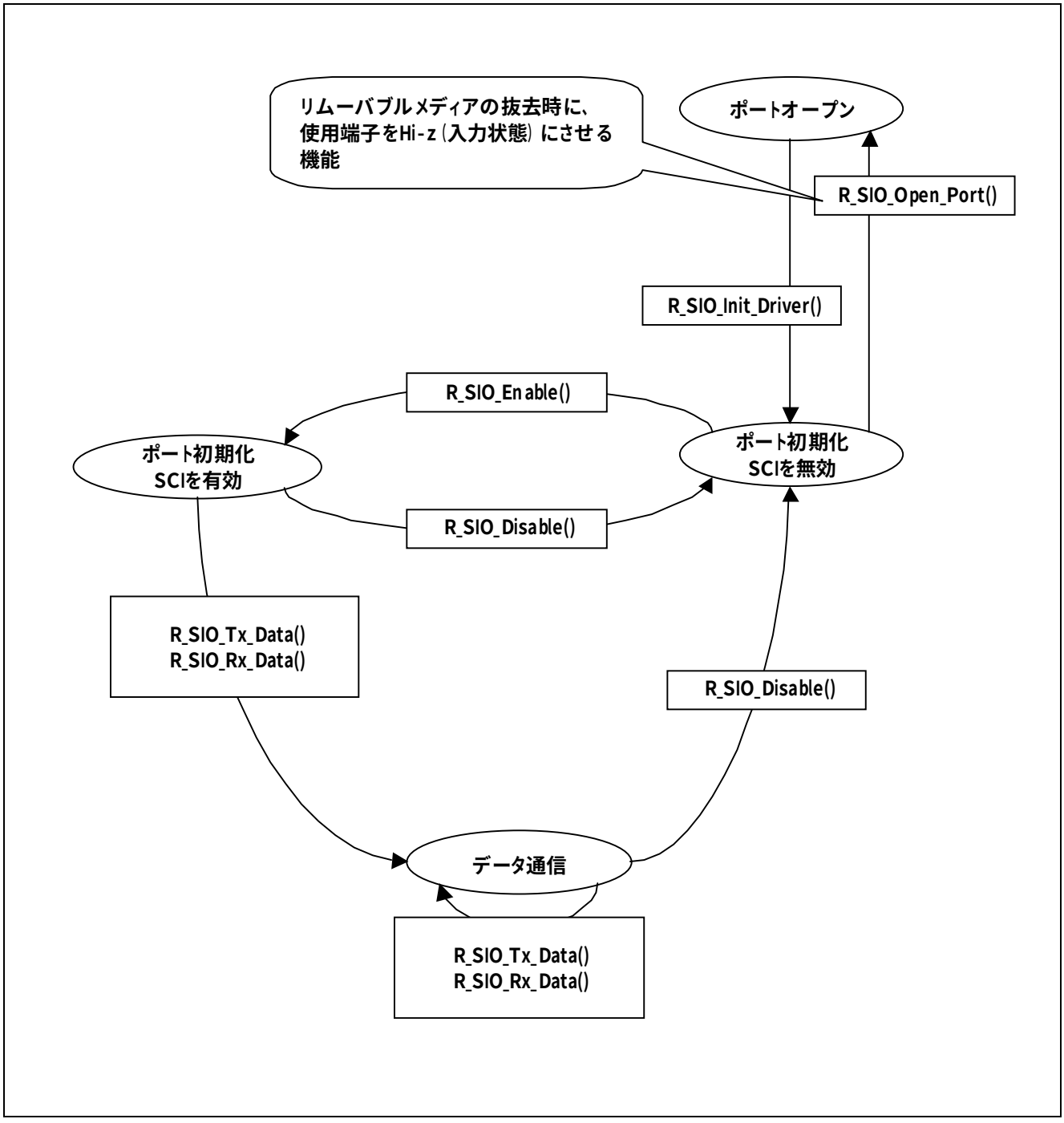


図 5-10 状態遷移図

6. 応用例

シリアル I/O 制御部分の設定例を示します。

使用する上での設定例を以下に示します。

設定箇所は、各ファイル中の「/** SET **/」というコメントの部分です。

6.1 mtl_com.h（共通ヘッダファイル）

共通で使用される共通関数のヘッダです。

mtl_com.h.XXX（mtl_com.h.common を除く）は、MCU 毎に評価目的で作成したものです。どれか一つを mtl_com.h にリネームして使用してください。対象 MCU のものが無い場合には、参照して、mtl_com.h を作成してください。

(1) OS のヘッダファイル定義

本サンプルコードは、OS 非依存です。

下記の例は、OS を使用しない場合の例です。

本サンプルコードは、OS 非依存のため、使用しない設定にしてください。他のソフトウェアに依存します。

```
/* システムコールを使用するため、                                          */
/* プロトタイプ宣言のある OS のヘッダファイルをインクルードしてください。  */
/* OS を使用しない場合は、下記デファインとインクルードをコメントにしてください。 */
//#define MTL_OS_USE                    /* Use OS                        */
//#include <RTOS.h>                     /* OS header file                */
//#include "mtl_os.h"
```

(2) 共通アクセス領域を定義したヘッダファイル定義

MCU の機能レジスタの定義がされているヘッダファイルをインクルードします。

主にデバイスドライバがポート制御等に使用するため、インクルードする必要があります。

MCU に合わせて、ヘッダファイルをインクルードしてください。

下記の例は、RX610 のヘッダファイルをインクルードする場合の例です。

本サンプルコード使用時は、インクルードしてください。

```
/* MCU の SFR 領域のデファイン値を使用しているため、                      */
/* I/O 周りのデファイン定義のあるヘッダファイルをインクルードしてください。 */
#include "iodefine.h"                       /* definition of MCU SFR     */
```

(3) ループタイマの定義

ソフトウェア・ループタイマを使用する場合、以下のヘッダをインクルードします。

主にデバイスドライバが待ち時間を確保するために、使用します。

ソフトウェア・ループタイマを使用しない場合は、下記インクルードをコメントにしてください

下記の例は、ソフトウェア・ループタイマを使用する場合の例です。

本サンプルコード使用時は、インクルードしてください。

```
/* ループタイマを使用しない場合は、下記インクルードをコメントにしてください。 */
#include "mtl_tim.h"
```

(4) エンディアンタイプ定義

リトルエンディアン／ビッグエンディアンのどちらかの指定が可能です。

下記は、ビッグエンディアンに設定した場合の例です。

```
/* MCU が、(1)SuperH でリトルエンディアン設定、(2)M16C  の場合、定義を有効にしてください。*/
/* その他の MCU を指定する時はリトルエンディアンの定義をコメントにしてください。          */
//#define MTL_MCU_LITTLE                    /* Little Endian                 */
```

(5) エンディアン処理の高速化の定義

mtl_end.c の処理の高速化指定が可能です。M16C を使用する場合、高速になります。

RX ファミリの場合は、コメントアウトし、定義しないでください。

```
/* M16C を使用する場合、定義を有効にしてください。                                  */
/* mtl_endi.c の処理の高速化が可能です。                                          */
//#define MTL_ENDI_HISPEED             /* Uses the high-speed function.    */
```

(6) 使用する標準ライブラリのタイプの定義

使用する標準ライブラリのタイプを定義してください。

下記に示す処理をコンパイラ添付のライブラリで使用する場合は、下記デファイン定義をコメントにしてください。

下記の例は、コンパイラ添付のライブラリを使用する場合例です。

```
/* 使用する標準ライブラリのタイプを指定してください。                               */
/* 下記に示す処理をコンパイラ添付のライブラリで使用する場合は、                     */
/* 下記デファイン定義をコメントにしてください。                                     */
/* memcmp() / memmove() / memcpy() / memset() / strcat() / strcmp() / strcpy() /
strlen()              */
//#define MTL_USER_LIB                    /* use optimized library          */
```

(7) アクセスする RAM 領域の定義

使用する RAM 領域を定義してください。

標準関数や一部の処理に効率の良い処理を適用します。

RX ファミリの場合は、MTL_MEM_NEAR を定義してください。

```
/* 使用する処理群がアクセスする RAM 領域を定義してください。                       */
/* 標準関数や一部の処理に効率の良い処理を適用します。                               */
//#define MTL_MEM_FAR         /* Supports Far RAM area of M16C/60          */
#define MTL_MEM_NEAR          /* Supports Near RAM area.       (Others)    */
```

6.1.2 mtl_tim.h

mtl_com.h にて、ループタイマを定義した場合に、インクルードされます。

使用する MCU、クロック、コンパイルオプション等に依存します。

また、命令キャッシュを有効にしているシステムでは、キャッシュ内に、ループタイマ処理が格納されている場合を想定して、設定してください。

使用環境に応じて、測定し直してください。

```
/* タイマのカウンタ値を定義してください。                                    */
/* MCU 及びクロック、ウエイトに応じて設定してください。                      */
#if 1
/* Setting for 12.5MHz no wait Ix8 = 100MHz(Compile Option "-optimize=1" or
"-optimize=1 -speed")*/
#define MTL_T_1US             30     /* loop Number of   1us           */
#define MTL_T_2US             60     /* loop Number of   2us           */
#define MTL_T_4US            120     /* loop Number of   4us           */
#define MTL_T_5US            150     /* loop Number of   5us           */
#define MTL_T_10US           300     /* loop Number of  10us           */
#define MTL_T_20US           600     /* loop Number of  20us           */
#define MTL_T_30US           900     /* loop Number of  30us           */
#define MTL_T_50US          1500     /* loop Number of  50us           */
#define MTL_T_100US         3000     /* loop Number of 100us           */
#define MTL_T_200US         6000     /* loop Number of 200us           */
#define MTL_T_300US         9000     /* loop Number of 300us           */
#define MTL_T_400US  ( MTL_T_200US  *  2 )   /* loop Number of 400us */
#define MTL_T_1MS          30000     /* loop Number of   1ms           */
#endif
```

上記は、未測定のため、妥当な値が設定されていませんので、評価してください。

6.2 クロック同期式シングルマスタ制御ソフトウェアの設定

設定箇所は、各ファイル中の「/** SET **/」というコメントの部分です。

6.2.1 R_SIO.h

(1) BRR 設定後のウェイトの定義

SCI の BRR 設定後、転送 1bit 期間をソフトウェア・ウェイトにより待ちます。ウェイト時間を設定してください。

初期値として、10μs を設定しています。

マルチメモリカードを使用する場合、100kHz 通信を想定し、10μs を設定してください。

```
#define SIO_T_BRR_WAIT          (uint16_t)MTL_T_10US /* BRR setting wait time */
```

6.2.2 R_SIO_sci.h

SCI 用の定義ファイルです。

R_SIO_sci.h.XXX は、各 MCU に評価目的で作成したものです。どれか一つを R_SIO_sci.h にリネームして使用してください。対象 MCU のものが無い場合には、参照して、R_SIO_sci.h を作成してください。

(1) 使用する動作モードの定義

使用する MCU のリソースの設定が可能です。

下記は、MSB ファーストでの CRC-CCITT 演算処理を行う場合に、SIO_OPTION_2 を指定してください。

シリアル EEPROM、シリアルフラッシュメモリを制御する場合は、CRC-CCITT 演算処理は不要です。コメントアウトしてください。

また、マルチメディアカード制御にて CRC-CCITT 演算処理を行う場合、別途 R_SIO_sci_rx_mmc.c が必要です。

```
/*------------------------------------------------------------------------*/
/*  Define the combination of the MCU's resources.                        */
/*------------------------------------------------------------------------*/
//#define SIO_OPTION_1     /* Low speed*/ /* SI/O                          */
#define SIO_OPTION_2       /*          */ /* SI/O      + CRC calculation   */
```

(2) 使用する CRC 演算処理の定義

使用する CRC 演算処理を定義してください。

シリアル EEPROM、シリアルフラッシュメモリを制御する場合は、CRC-CCITT 演算処理を使用しません。コメントアウトしてください。

マルチメディアカードを制御する場合、同時に両方を定義してください。

```
/*------------------------------------------------------------------------*/
/*  Define the CRC calculation.                                           */
/*------------------------------------------------------------------------*/
#define SIO_CRCCCITT_USED     /* CRC-CCITT used                    */
#define SIO_CRC7_USED         /* CRC7 used                         */
```

(3) 使用する端子の定義

使用する端子を指定してください。

```
/*------------------------------------------------------------------------*/
/*  Define the control port.                                              */
/*  Delete comment of a related macrodefinition, and please validate setting.*/
/*------------------------------------------------------------------------*/
#define SIO_DR_DATAO    PORT2.DR.BIT.B6         /* SIO DataOut          */
#define SIO_PORT_DATAI  PORT2.PORT.BIT.B5       /* SIO DataIn           */
#define SIO_DR_CLK      PORT2.DR.BIT.B7         /* SIO CLK              */
#define SIO_DDR_DATAO   PORT2.DDR.BIT.B6        /* SIO DataOut          */
#define SIO_DDR_DATAI   PORT2.DDR.BIT.B5        /* SIO DataIn           */
#define SIO_DDR_CLK     PORT2.DDR.BIT.B7        /* SIO CLK              */
#define SIO_ICR_DATAI   PORT2.ICR.BIT.B5        /* SIO DataIn           */
```

(4) モジュールストップレジスタの定義

使用する SCIF に関するモジュールストップレジスタを指定してください。

```
#define SIO_MSTPCR_SCI  SYSTEM.MSTPCRB.BIT.MSTPB30 /* SCI Module stop setting*/
```

(5) 使用する SCI チャネルの定義

使用する SCI チャネルを指定してください。

以下は、チャネル 1 を使用する場合の例です。

```
/*----------------- SIO definitions -----------------*/

#define SIO_SMR         SCI1.SMR.BYTE        /* Serial mode register       */
#define SIO_SCR         SCI1.SCR.BYTE        /* Serial control register    */
#define SIO_SSR         SCI1.SSR.BYTE        /* Serial status register     */
#define SIO_SCMR        SCI1.SCMR.BYTE       /* Smart card mode register   */
#define SIO_BRR         SCI1.BRR             /* Bit rate register          */
#define SIO_SEMR        SCI1.SEMR.BYTE       /* Serial extend mode register*/
#define SIO_TXBUF       SCI1.TDR       /* SCI Transmit FIFO data register  */
#define SIO_RXBUF       SCI1.RDR       /* SCI Receive FIFO data register   */

#define SIO_ORER        SCI1.SSR.BIT.ORER    /* SCI Overrun error flag     */
#define SIO_FER         SCI1.SSR.BIT.FER     /* SCI Framing error flag     */
#define SIO_PER         SCI1.SSR.BIT.PER     /* SCI Parity error flag      */
#define SIO_TXEND       SCI1.SSR.BIT.TEND    /* SCI Transmit end flag      */
#define SIO_TXNEXT      ICU.IR[220].BIT.IR   /* SCI Transmit data empty    */
#define SIO_RXNEXT      ICU.IR[219].BIT.IR   /* SCI Receive data full      */
```

7. 使用上の注意事項

7.1 組み込み時の注意事項

本サンプルコードを組み込む場合は、R_SIO.h、R_SIO_sci.h（R_SIO_sci.h.XXX をリネーム）をインクルードしてください。

7.2 不必要な関数について

使用されない関数は、ROM を不必要に消費しますので、コメントアウト等の処理にて、組み込まれないことを推奨します。

7.3 他 MCU を使用する場合

他 MCU を使用する場合、容易に対応が可能です。

準備するファイルは、

- R_SIO_sci.h.XXX に相当する I/O モジュール共通定義
- mtl_com.h.XXX に相当するヘッダ定義

です。添付のものを参考に、作成してください。

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/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.06.30	—	初版発行

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。

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・電話番号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 エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■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/inquiry

Colophon 1.0