

RX62N グループ

R01AN1088JJ0101

Rev.1.01

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

2012.06.15

要旨

本アプリケーションノートでは、RX62N 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。

本サンプルコードは、マイコン固有のシングルマスタ基本制御方法を実現したものです。本サンプルコードを使用して、スレーブデバイスを制御するためのソフトウェアを作成してください。

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

対象デバイス

対応 MCU RX62N 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
- Micron Technology 社製 M25P シリーズ Serial Flash memory 64Mbit
- Micron Technology 社製 M45PE シリーズ Serial Flash memory 1Mbit

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	2
2. 動作確認条件	3
3. 関連アプリケーションノート	4
4. ハードウェア説明	5
5. ソフトウェア説明	6
6. 応用例	32
7. 使用上の注意事項	41

1. 仕様

RX62N 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 RX62N グループ MCU 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 チャンネルの使用が可能です。複数チャンネルの使用は、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。
- データのソフトウェア変換により、MSB ファーストフォーマットで転送します。
- CPU 転送のみをサポートしています。DMAC/EXDMAC/DTC 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。

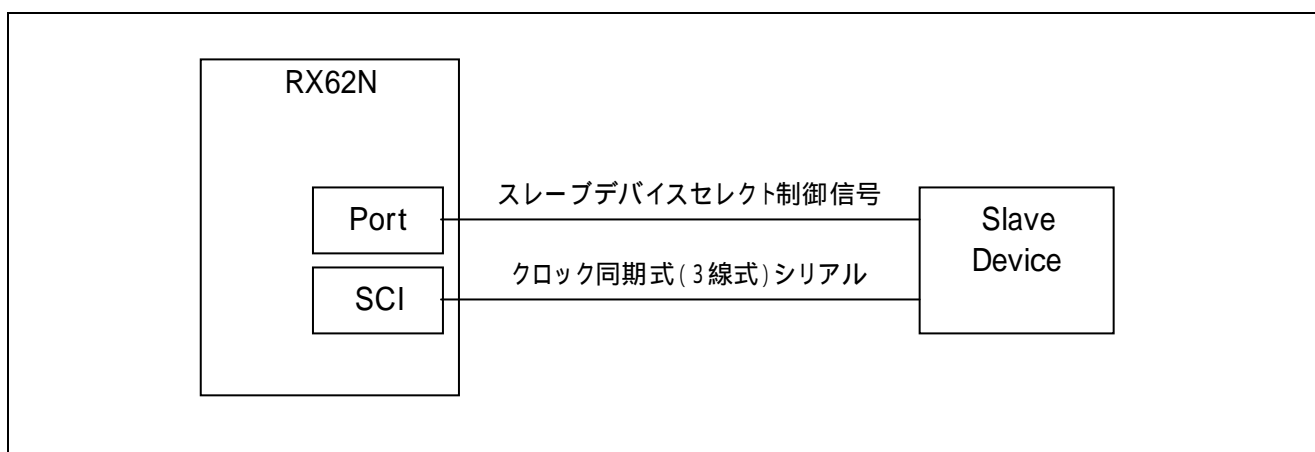


図 1-1 使用例

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
使用マイコン	RX62N グループ (プログラム ROM 512KB/RAM 96KB)
使用メモリ	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 : 96MHz、PCLK : 48MHz
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 4.09.00.007
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++コンパイラパッケージ (ツールチェーン 1.2.1.0) コンパイルオプション 総合開発環境のデフォルト設定 (1) を使用しています。 1 : 最適化レベル"2"、最適化方法"サイズ優先"
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.2.01
使用ソフトウェア	Renesas R1EX25xxx シリーズ Serial EEPROM 制御ソフトウェア (R01AN0565JJ) Ver.2.02
使用ボード	Renesas Starter Kit for RX62N

表 2-2 動作確認条件

項目	内容
使用マイコン	RX62N グループ (プログラム ROM 512KB/RAM 96KB)
使用メモリ	Micron Technology 社製 M25P シリーズ Serial Flash memory 64Mbit
動作周波数	ICLK : 96MHz、PCLK : 48MHz
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 4.09.00.007
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++コンパイラパッケージ (ツールチェーン 1.2.1.0) コンパイルオプション 総合開発環境のデフォルト設定 (1) を使用しています。 1 : 最適化レベル"2"、最適化方法"サイズ優先"
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.2.01
使用ソフトウェア	Micron Technology 社製 M25P シリーズ Serial Flash memory 制御ソフトウェア (R01AN0566JJ) Ver.2.01
使用ボード	Renesas Starter Kit for RX62N

表 2-3 動作確認条件

項目	内容
使用マイコン	RX62N グループ (プログラム ROM 512KB/RAM 96KB)
使用メモリ	Micron Technology 社製 M45PE シリーズ Serial Flash memory 1Mbit
動作周波数	ICLK : 96MHz、PCLK : 48MHz
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 4.09.00.007
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++コンパイラパッケージ (ツールチェーン 1.2.1.0) コンパイルオプション 総合開発環境のデフォルト設定 (1) を使用しています。 1 : 最適化レベル"2"、最適化方法"サイズ優先"
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.2.01
使用ソフトウェア	Micron Technology 社製 M45PE シリーズ Serial Flash memory 制御ソフトウェア(R01AN0567JJ) Ver.2.01
使用ボード	Renesas Starter Kit for RX62N

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(R01AN0565JJ)
- Micron Technology 社製 M25P シリーズ Serial Flash memory 制御ソフトウェア(R01AN0566JJ)
- Micron Technology 社製 M45PE シリーズ Serial Flash memory 制御ソフトウェア(R01AN0567JJ)

4. ハードウェア説明

4.1 ハードウェア構成例

図 4-1に接続図を示します。

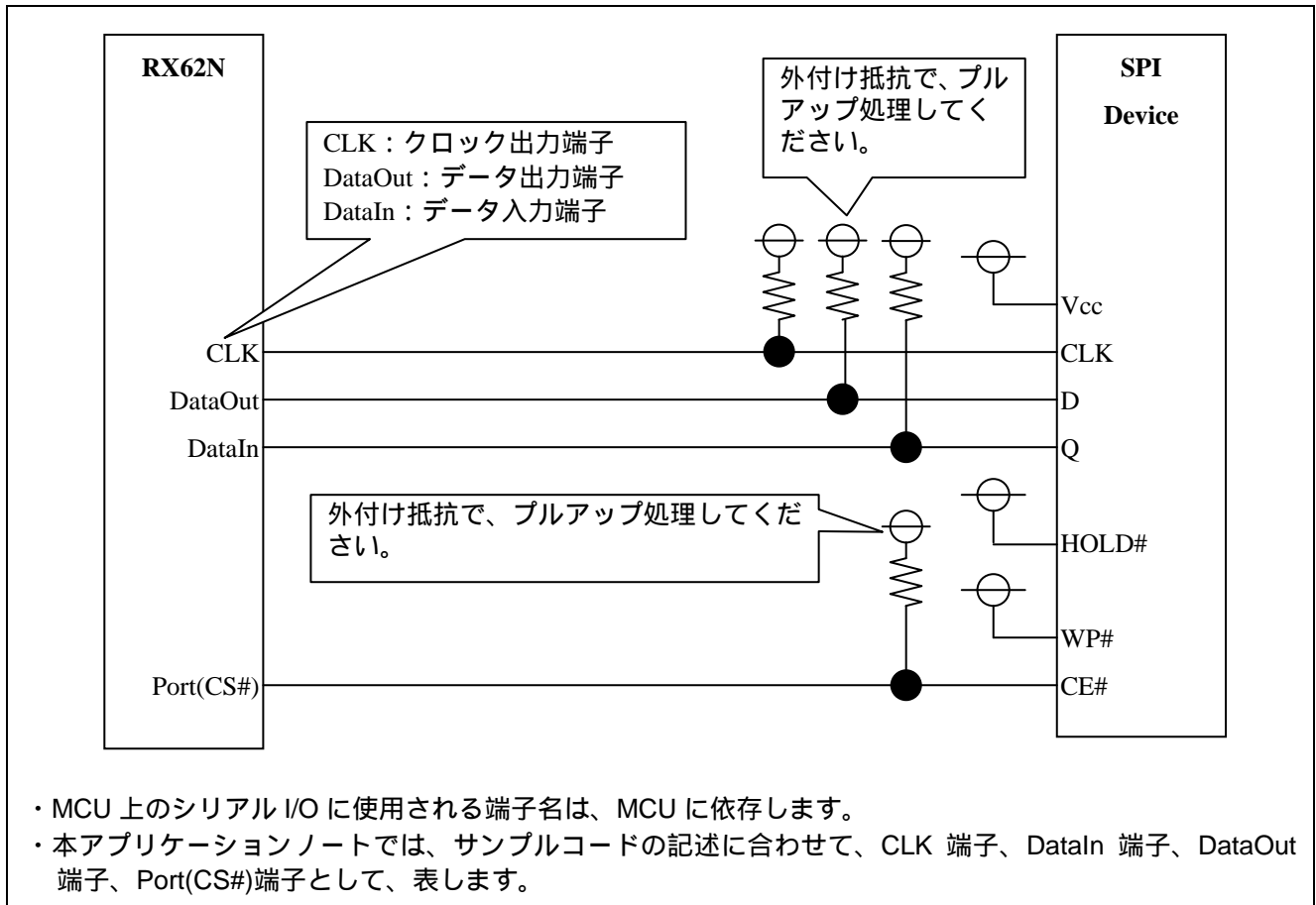


図 4-1 RX62N SCI と SPI スレーブデバイスの接続例

4.2 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。

5. ソフトウェア説明

5.1 動作概要

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。

- データの送信 / 受信を、クロック同期式モード (内部クロック使用) で、制御する。

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 5-1 に示す SPI モード 3 (CPOL=1、CPHA=1) のタイミングを発生します。

RX62N SCI は、MSB ファーストデータ転送機能をサポートしていません。本サンプルコードでは、データのソフトウェア変換により、MSB ファーストフォーマットで転送します。

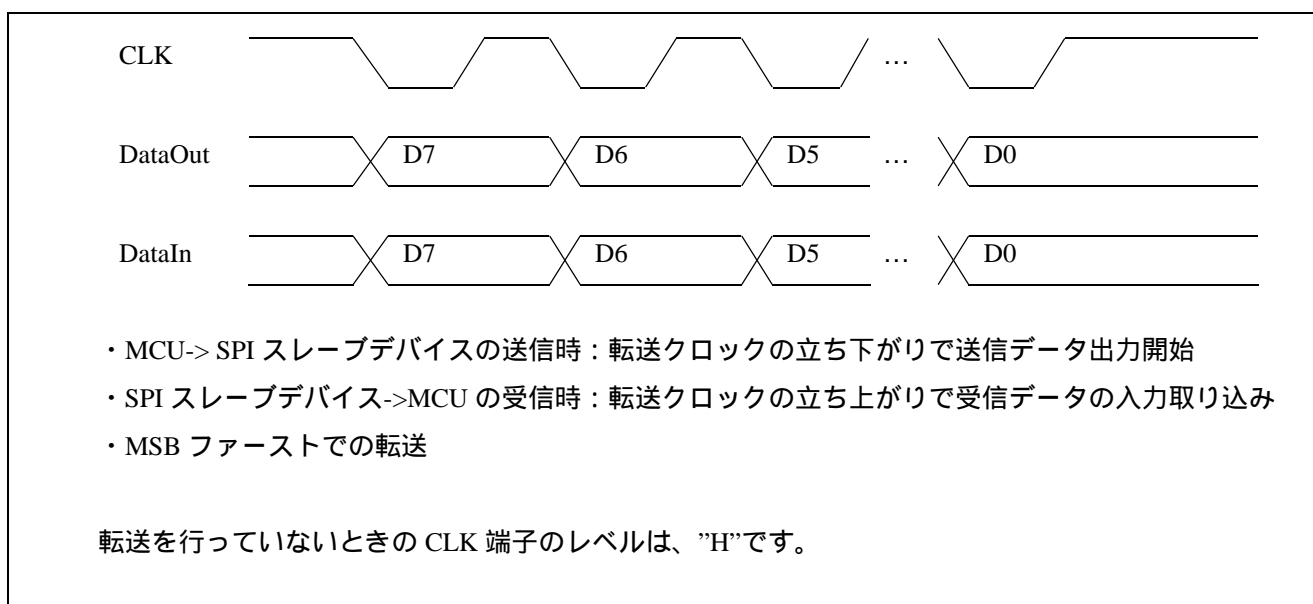


図 5-1 クロック同期式モード タイミング設定

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#端子を制御しません。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 (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、マイコン固有のシングルマスタ基本制御方法を実現したものです。

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 SPI モード 3 (CPOL=1、CPHA=1) を使った制御を実現しています。

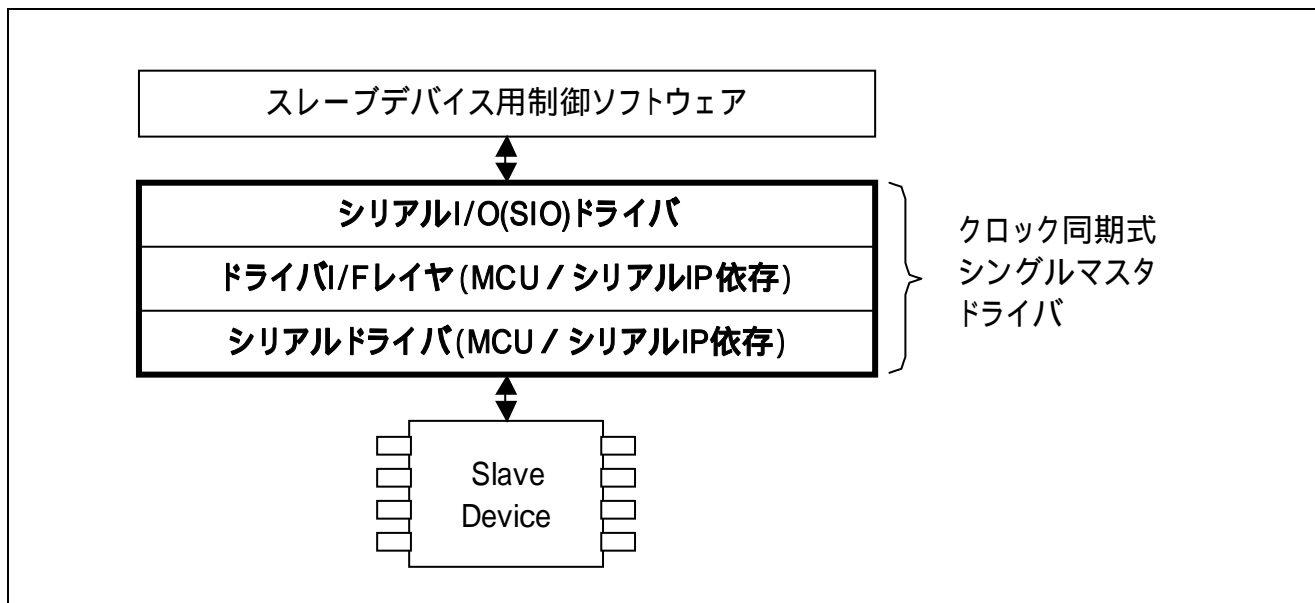


図 5-2 ソフトウェア構成

5.8 状態遷移図、および5.9 関数仕様に示す関数を参照して、スレーブデバイスへのアクセスを実現してください。

具体的な使用例として、3 関連アプリケーションノートに記載済のアプリケーションノートを参考にしてください。

5.2.2 データバッファと送信 / 受信データの関係

本サンプルコードは、ブロック型デバイスドライバであり、送信 / 受信データポインタを引数として設定します。RAM 上のデータバッファのデータ並びと送信 / 受信順番の関係は、以下のとおりで、エンディアンや使用するシリアル通信機能に関係なく、送信データバッファの並びの順に送信し、また、受信の順に受信データバッファに書き込みます。

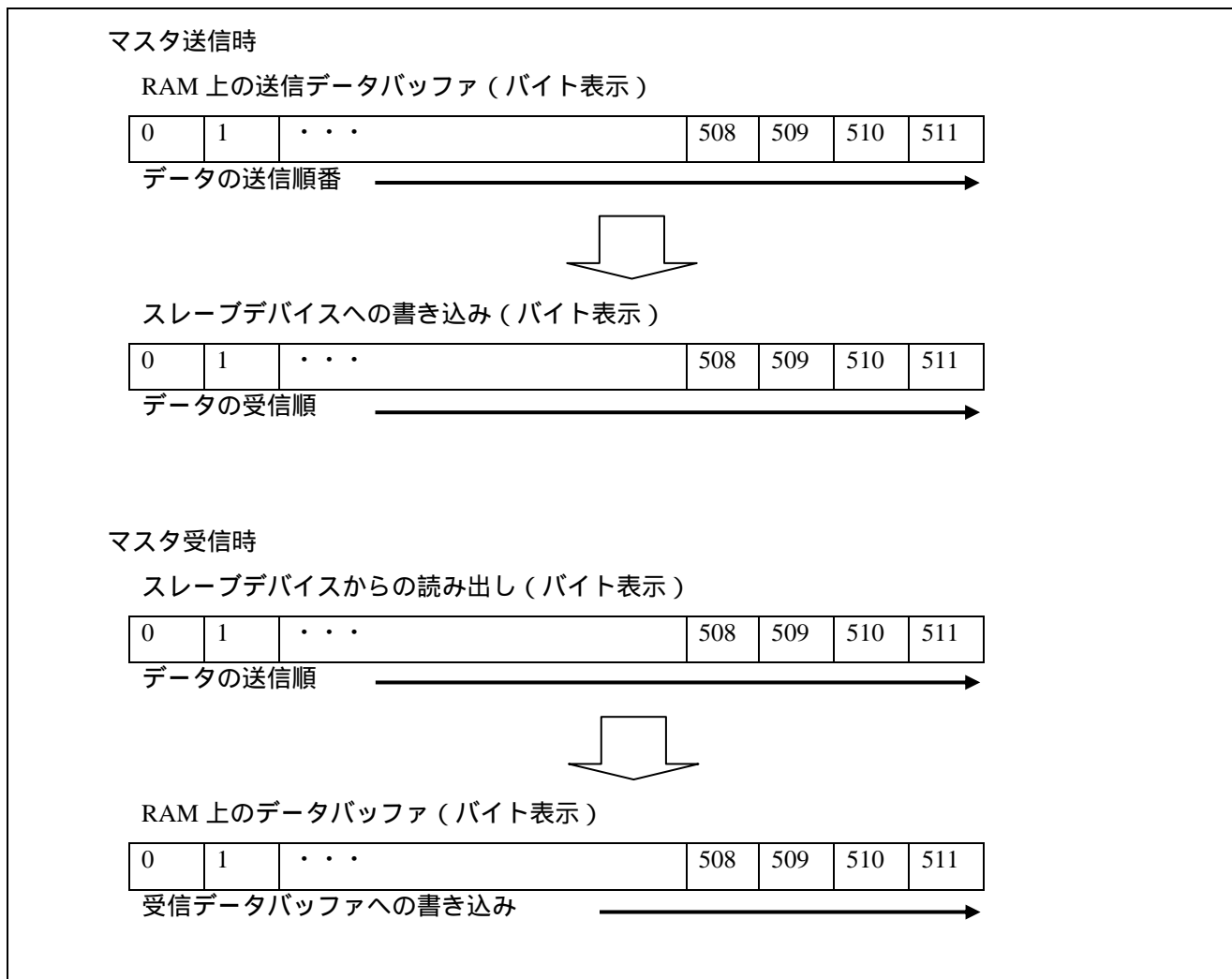


図 5-3 データバッファと送信 / 受信データの関係

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のメモリサイズは、6.2.2 R_SIO_sci.h (1) 使用する動作モードの定義 で SIO_OPTION_1 を選択した場合の値です。選択する定義により、メモリサイズは異なります。

表 5-1 必要メモリサイズ

使用メモリ	サイズ	備考
ROM	696 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	52 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_r01an1088jj_rx62n	<DIR>	サンプルコードのフォルダ
r01an1088jj0101_rx62n.pdf		アプリケーションノート
¥ source	<DIR>	プログラム格納用フォルダ
¥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.RX	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>	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.rx62n	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 リターン値

表 5-3に、サンプルコードで使用するリターン値を示します。

表 5-3 リターン値

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 reception 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 reception completion waiting time 50000* 1us = 50ms
SIO_T_SIO_WAIT	(uint16_t)MTL_T_1US	SIO transmission&reception completion waiting polling time
SIO_T_DMA_WAIT	(uint16_t)MTL_T_1US	DMA transmission&reception 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

関数名	説明
R_SIO_Init_Driver()	ドライバ初期化处理
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 状態遷移図

図 5-4に、状態遷移図を示します。

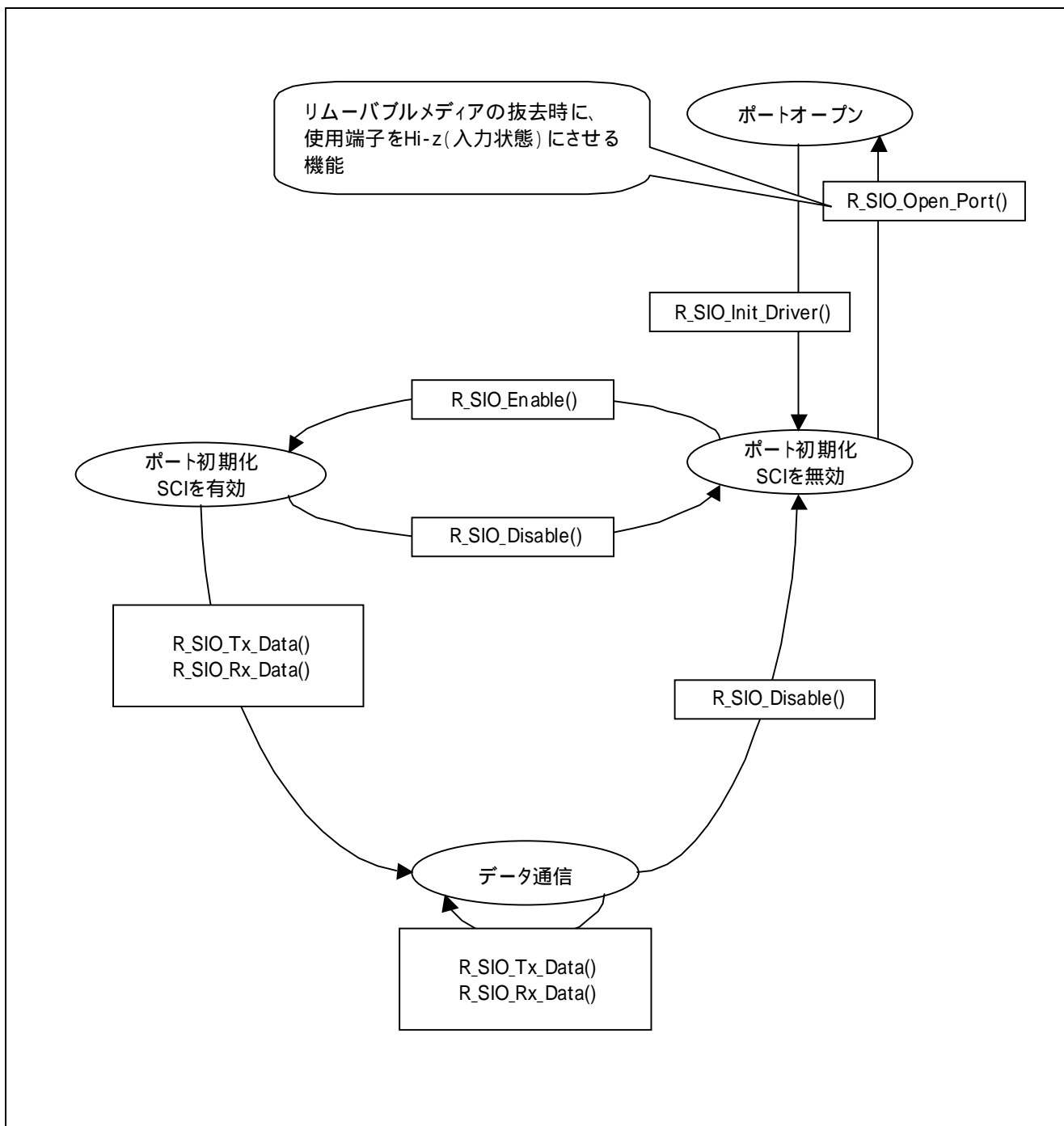


図 5-4 状態遷移図

5.9 関数仕様

5.9.1 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	<ul style="list-style-type: none"> ・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	<p>前の使用状態を考慮し、以下の処理を行います。</p> <ul style="list-style-type: none"> ・R_SIO_Disable()をコールします。

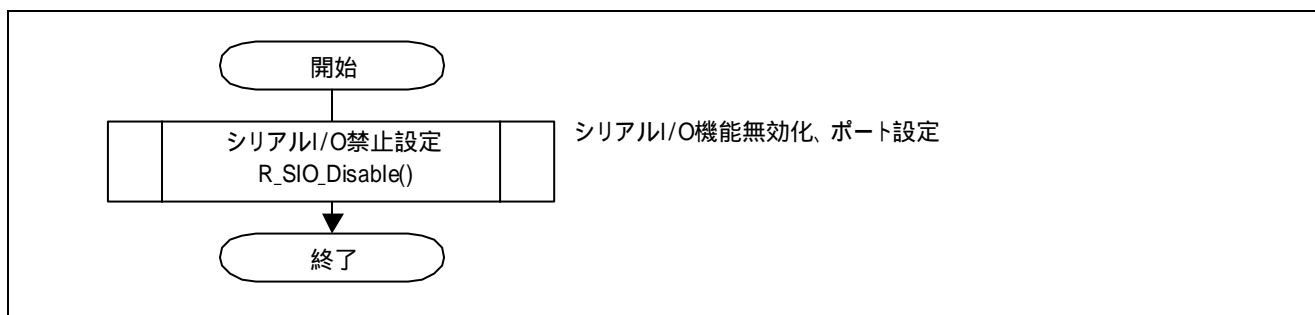


図 5-5 ドライバ初期化処理概要

5.9.2 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> SCI 関連のレジスタへ書き込みを行うため、使用する SCI のモジュールストップ状態を一時的に解除します。SCI 関連のレジスタを設定後、モジュールストップ状態へ遷移します。 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。

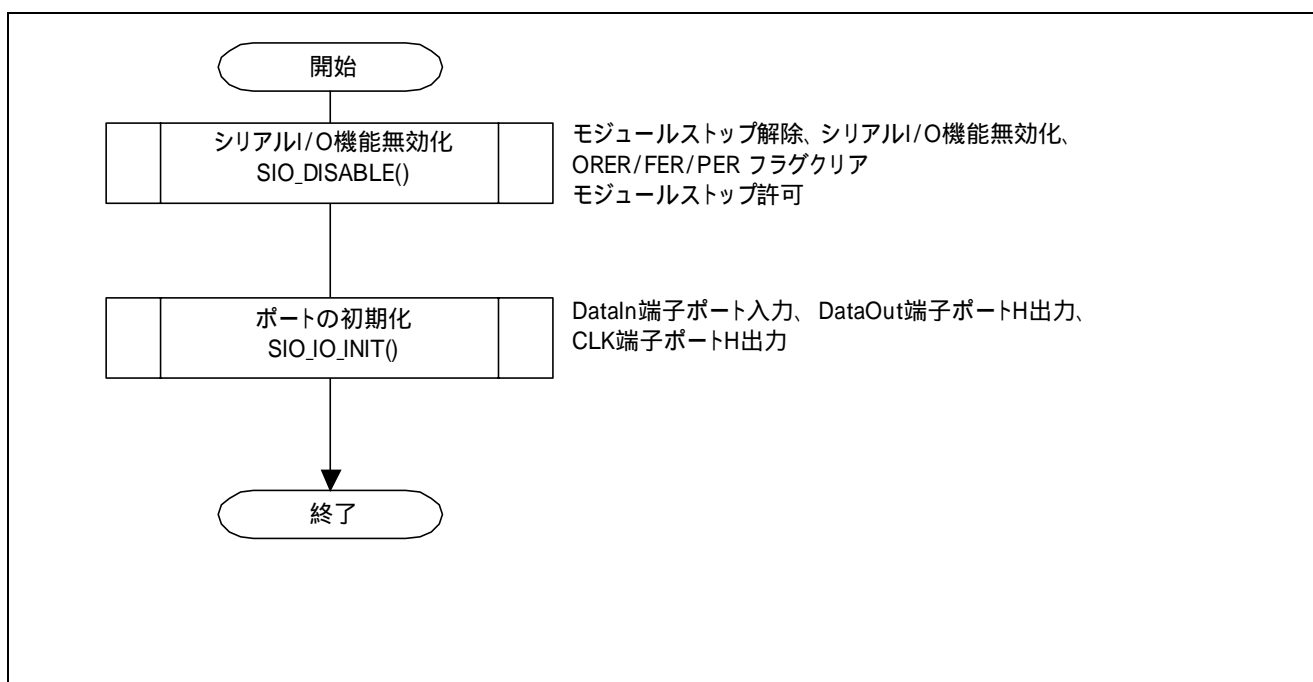


図 5-6 シリアル I/O 禁止設定処理概要

5.9.3 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	<ul style="list-style-type: none"> ・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	<ul style="list-style-type: none"> ・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。 ・ソフトウェア・ウェイト (10 μs) は、ビットレートの設定待ち時間です。

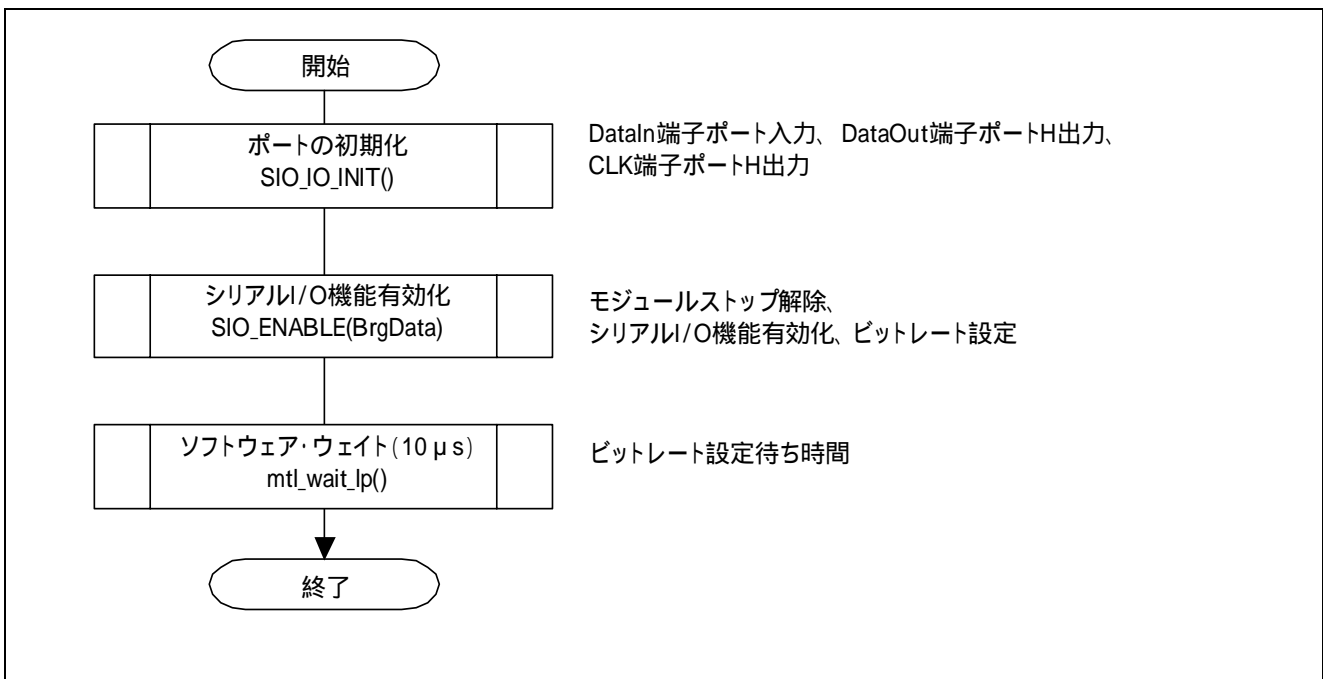


図 5-7 シリアル I/O 許可設定処理概要

5.9.4 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	<ul style="list-style-type: none"> ・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	<ul style="list-style-type: none"> ・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。

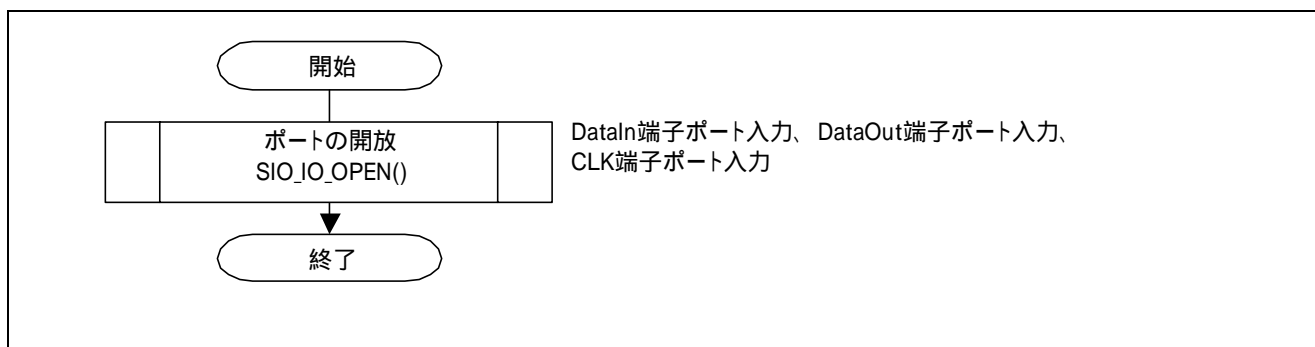


図 5-8 シリアル I/O 開放設定処理概要

5.9.5 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	<ul style="list-style-type: none"> ・ 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	<ul style="list-style-type: none"> ・ 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。(インライン関数 SIO_TX_ENABLE()) <ul style="list-style-type: none"> IR フラグクリア PFC 設定 (チャンネル 1/2/3/6 設定時のみ必要) SCR 設定 (送信許可設定) SCR をリード ・ 送信完了後、上記送信許可設定の逆の処理を行うことで、シリアル通信を停止します。(インライン関数 SIO_TX_DISABLE()) <ul style="list-style-type: none"> SCR 設定 (送受信停止設定) SCR をリード IR フラグクリア ・ 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。

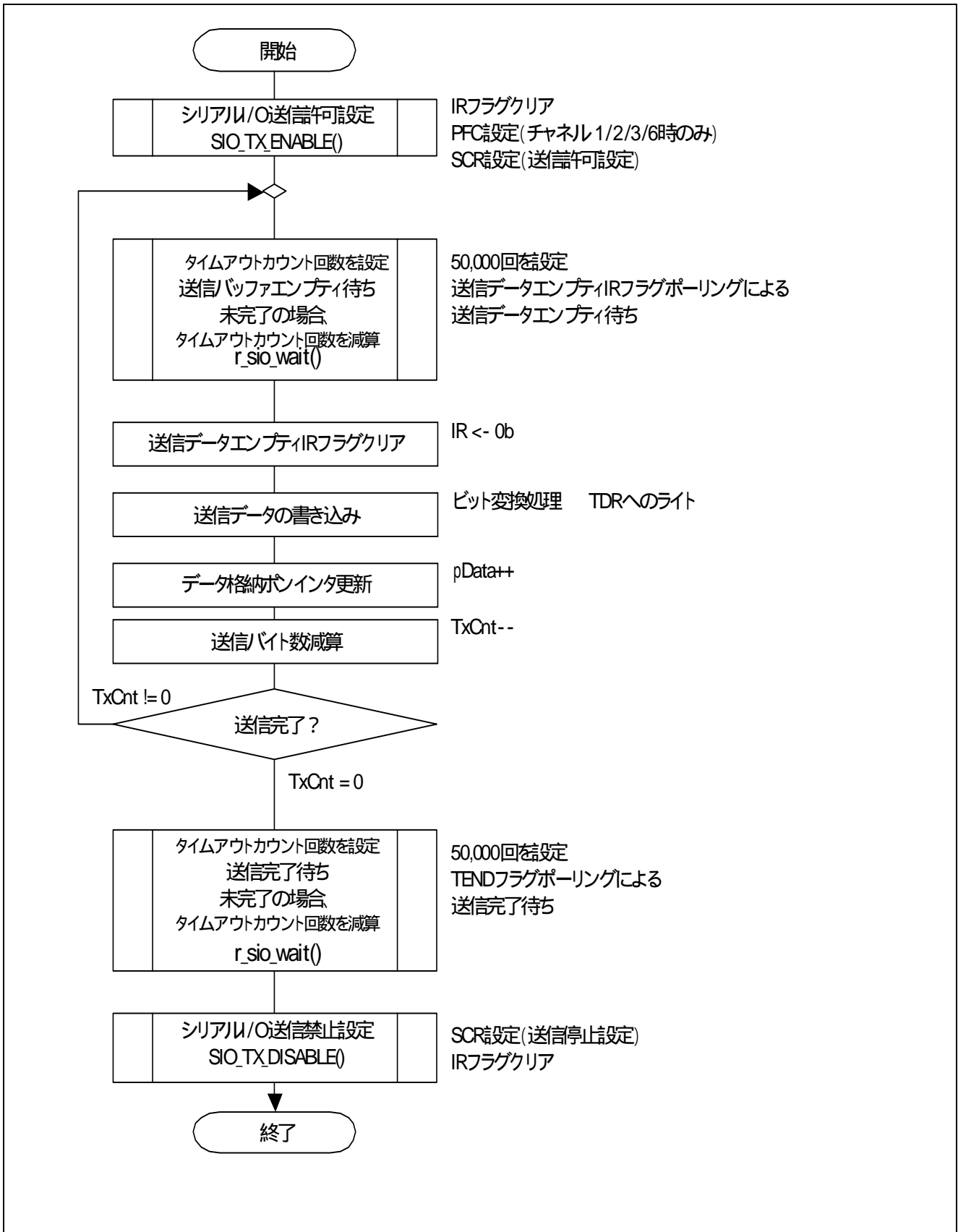


図 5-9 シリアル I/O データ送信処理概要

5.9.6 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	<ul style="list-style-type: none"> ・ 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	<ul style="list-style-type: none"> ・ 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。(インライン関数 SIO_TRX_ENABLE()) <ul style="list-style-type: none"> IR フラグクリア PFC 設定 (チャンネル 1/2/3/6 設定時のみ必要) SCR 設定 (送受信許可設定) SCR をリード ・ 受信完了後、上記送受信許可設定の逆の処理を行うことで、シリアル通信を停止します。(インライン関数 SIO_TRX_DISABLE()) <ul style="list-style-type: none"> SCR 設定 (送受信停止設定) SCR をリード IR フラグクリア ・ 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。

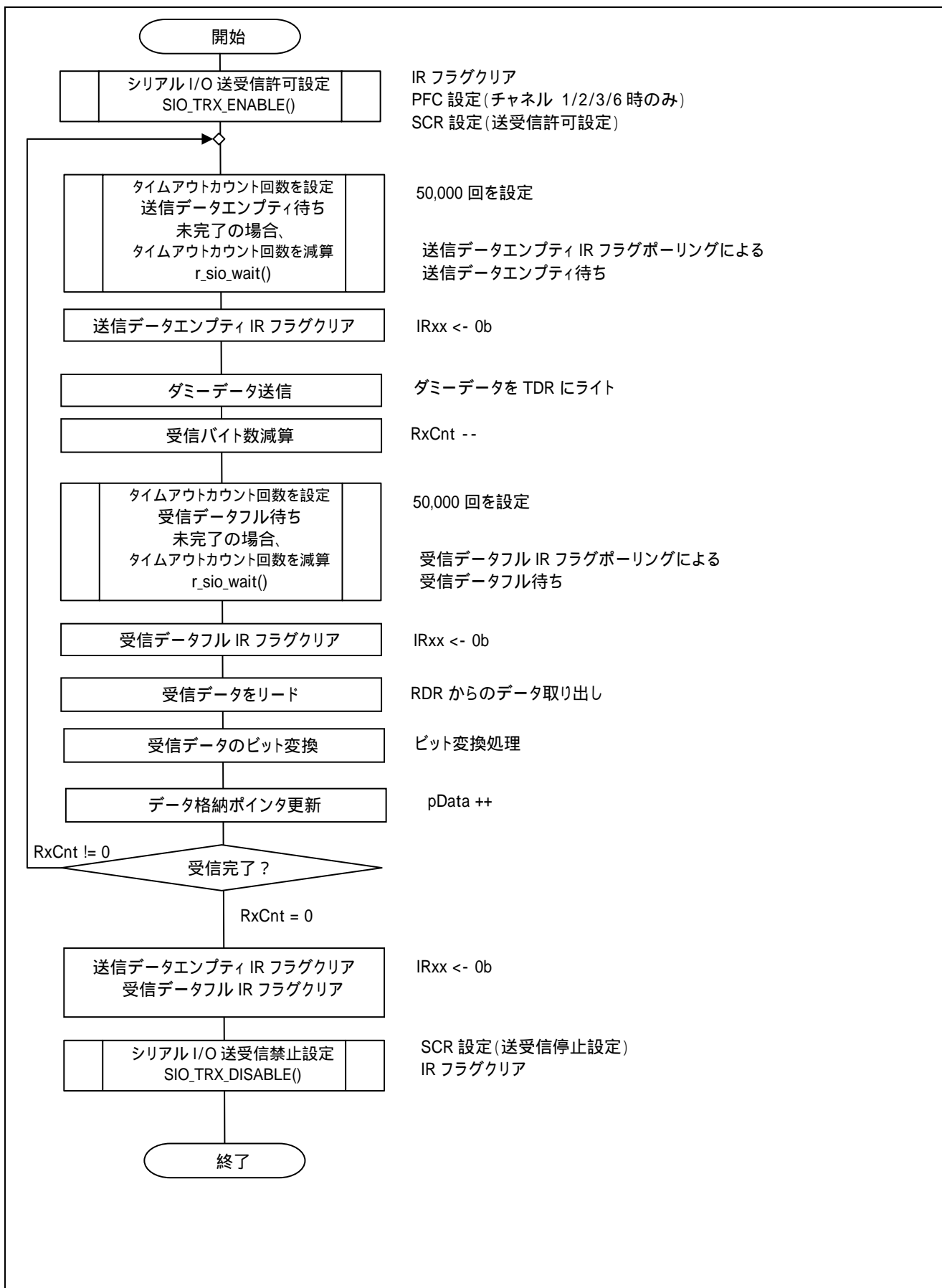


図 5-10 シリアル I/O データ受信処理概要

5.10 インライン関数仕様

以下に、本サンプルコードで使用するインライン関数を示します。

5.10.1 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。

DataIn 端子をポート入力に設定する。

SIO_DATAI_INIT()を参照してください。

DataOut 端子をポート”H”出力に設定する。

SIO_DATAO_INIT()を参照してください。

CLK 端子をポート”H”出力に設定する。

SIO_CLK_INIT()を参照してください。

(3) 備考

本インライン関数により、端子を周辺機能からポート機能に変更します。本関数を呼び出す前に、他の周辺機能を使用していないか確認してから実行するようにしてください。

5.10.2 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。

DataIn 端子をポート入力に設定する。

SIO_DATAI_INIT()を参照してください。

DataOut 端子をポート入力に設定する。

SIO_DATAO_OPEN()を参照してください。

CLK 端子をポート入力に設定する。

SIO_CLK_OPEN()を参照してください。

(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。

SIO_IO_INIT()を実行後に、本関数を実行してください。

5.10.3 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。

入力バッファコントロールレジスタ(ICR)を使って、DataIn 端子の入力バッファ機能を有効にする。

- ・ DataIn 端子 ICR <- 1b : 入力バッファ有効

プルアップ抵抗コントロールレジスタ(PCR)を使って、DataIn 端子の入力プルアップ抵抗を無効にする。

(1)

- ・ DataIn 端子 PCR <- 0b : 入力プルアップ無効 (2)

データディレクションレジスタ(DDR)を使って、DataIn 端子をポート入力に設定する。

- ・ DataIn 端子 DDR <- 0b : 入力ポート

(3) 備考

- 1 : プルアップ抵抗コントロールレジスタ(PCR)の設定が可能な端子は、Port9,PortA ~ PortE 及び PortG です。PortC を使用する場合は、SIO_PCR_DATAI を設定してください。

- 2 : 必要に応じて設定してください。

5.10.4 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。

オープンドレインコントロールレジスタ(ODR)を使って、DataOut 端子の出力形態を CMOS 出力に設定する。(1)

- DataOut 端子 ODR <- 0b : CMOS 出力 (3)

入力バッファコントロールレジスタ(ICR)を使って、DataOut 端子の入力バッファ機能を無効にする。(2)

- DataOut 端子 ICR <- 0b : 入力バッファ無効

データレジスタ(DR)を使って、DataOut 端子を H 出力にする。

- DataOut 端子 DR <- 1b : H 出力

データディレクションレジスタ(DDR)とデータレジスタ(DR)を使って、DataOut 端子をポート出力に設定する。

- DataOut 端子 DDR <- 1b : 出力ポート

- DataOut 端子 DR <- 1b : H 出力

(3) 備考

- 1 : オープンドレインコントロールレジスタ(ODR) の設定が可能な端子は、Port0 ~ Port3 及び PortC です。Port0 ~ Port3 及び PortC を使用する場合は、SIO_ODR_DATAO 及び SIO_ODR_CLK を設定してください。
- 2 : 端子を出力端子として使用する場合、入力バッファコントロールレジスタ(ICR)の設定で入力バッファ機能を有効にすると、出力データが端子状態として取り込まれます。したがって、出力として使用する端子は、ICR の設定で入力バッファ機能を無効にする必要があります。
- 3 : 必要に応じて設定してください。

5.10.5 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。

DataOut 端子をポート入力に設定する。

- DataOut 端子 DDR <- 0b : 入力ポート (入力バッファ無効)

(3) 備考

なし

5.10.6 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。

オープンドレインコントロールレジスタ(ODR)を使って、CLK 端子の出力形態を CMOS 出力に設定する。
(1)

・ CLK 端子 ODR <- 0b : CMOS 出力 (3)

入力バッファコントロールレジスタ(ICR)を使って、CLK 端子の入力バッファ機能を無効にする。(2)

・ CLK 端子 ICR <- 0b : 入力バッファ無効

データレジスタ(DR)を使って、CLK 端子を H 出力にする。

・ CLK 端子 DR <- 1b : H 出力

データディレクションレジスタ(DDR)とデータレジスタ(DR)を使って、CLK 端子をポート出力に設定する。

・ CLK 端子 DDR <- 1b : 出力ポート

・ CLK 端子 DR <- 1b : H 出力

(3) 備考

1 : オープンドレインコントロールレジスタ(ODR) の設定が可能な端子は、Port0 ~ Port3 及び PortC です。Port0 ~ Port3 及び PortC を使用する場合は、SIO_ODR_DATA0 及び SIO_ODR_CLK を設定してください。

2 : 端子を出力端子として使用する場合、入力バッファコントロールレジスタ(ICR)の設定で入力バッファ機能を有効にすると、出力データが端子状態として取り込まれます。したがって、出力として使用する端子は、ICR の設定で入力バッファ機能を無効にする必要があります。

3 : 必要に応じて設定してください。

5.10.7 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。

CLK 端子をポート入力に設定する。

・ CLK 端子 DDR <- 0b : 入力ポート (入力バッファ無効)

(3) 備考

なし

5.10.8 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 / 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ルにしたがって、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。

RX62N の場合、以下の処理を行います。

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

- ・ MSTPCRB.MSTPB_{xx} <- 0b : モジュールストップ解除、SCI レジスタへのリード/ライト可能化
- ・ MSTPCRB.MSTPB_{xx} をリード

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
- ・ DataIn 端子 ICR <- 1b : 入力バッファ有効
- ・ DataOut 端子, CLK 端子 ICR <- 0b : 入力バッファ無効
- ・ 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 クリア
SIO_SSR_CLEAR()を参照してください。
- ・ SEMR 設定
- ・ BRR に値を設定

(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、インライン関数終了後、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。

本関数実行前には、SIO_DISABLE()/SIO_TX_DISABLE()/SIO_TRX_DISABLE() (SCR 制御による通信動作停止) のいずれかを実行し、通信動作を停止させてください。

5.10.9 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。

RX62N の場合、以下の処理を行います。

SCI 関連のレジスタ設定を行うため、モジュールストップコントロールレジスタ(MSTPCRB)を使って、モジュールストップ解除状態にする。(1)

- ・ MSTPCRB.MSTPB_{xx} <- 0b : モジュールストップ解除、SCI レジスタへのリード/ライト可能化
- ・ MSTPCRB.MSTPB_{xx} をリード

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

SMR に初期値 00h を設定

SSR の ORER, FER, PER クリア

SIO_SSR_CLEAR()を参照してください。

モジュールストップコントロールレジスタ(MSTPCRB)を使って、モジュールストップ許可状態に設定

- ・ MSTPCRB.MSTPB_{xx} <- 1b : モジュールストップ許可、SCI レジスタへのリード/ライト不可(SCI レジスタの状態は保持)
- ・ MSTPCRB.MSTPB_{xx} をリード

(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。

SCI 関連のレジスタを設定するために、SCR 初期値 00h を設定し、送信 / 受信を停止させます。

- 1 : RX62N の場合、モジュールストップ状態に設定されたモジュールのレジスタは、読み出し、書き込みともできません。本インライン関数では SCR により SCI 機能の無効化を行うため、一度モジュールストップを解除し、SCR を設定後、モジュールストップを有効にする仕様としています。尚、モジュールストップ状態では、レジスタの値は保持されます。

5.10.10 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ルにしたがって、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()の後の初期化手順として、送信設定専用の初期化処理を行います。

RX62N の場合、以下の処理を行います。

IR フラグをクリア

SIO_IR_CLEAR()を参照してください。

SCI で使用する端子を選択する (チャンネル 1/2/3/6 設定時のみ)

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。

SCR をリード

(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。

5.10.11 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。

RX62N の場合、以下の処理を行います。

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。

SCR をリード

IR フラグをクリア

SIO_IR_CLEAR()を参照してください。

(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。

5.10.12 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ルにしたがって、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()の後の初期化手順として、送受信設定専用の初期化処理を設定します。

RX62N の場合、以下の処理を行います。

IR フラグをクリア

SIO_IR_CLEAR()を参照してください。

SCI で使用する端子を選択する (チャンネル 1/2/3/6 設定時のみ)

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。

SCR をリード

(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。

5.10.13 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切り替えます。必要に応じて、処理を見直してください。

RX62N の場合、以下の処理を行います。

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。

SCR をリード

IR フラグをクリア

SIO_IR_CLEAR()を参照してください。

(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。

5.10.14 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。

RX62N の場合、以下の処理を行います。

フラグ = 1 であれば、0 クリアする。

フラグをリードし、0 であることを確認する。

(3) 備考

なし

5.10.15 SIO_IR_CLEAR()

(1) 目的

IR フラグをクリアします。

(2) 機能

以下の手順でフラグのクリアを行います。必要に応じて、処理を見直してください。

RX62N の場合、以下の処理を行います。

IR フラグをクリア

(3) 備考

なし

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、RX62N 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。

```
/* タイマのカウント値を定義してください。 */
/* MCU 及びクロック、ウェイトに応じて設定してください。 */
#if 1
/* Setting for 12MHz no wait Ix8 = 96MHz (Compile Option "-optimize=2", com.V406R00) */
#define MTL_T_1US 10 /* loop Number of 1us */
#define MTL_T_2US 20 /* loop Number of 2us */
#define MTL_T_4US 40 /* loop Number of 4us */
#define MTL_T_5US 50 /* loop Number of 5us */
#define MTL_T_10US 100 /* loop Number of 10us */
#define MTL_T_20US 200 /* loop Number of 20us */
#define MTL_T_30US 300 /* loop Number of 30us */
#define MTL_T_50US 500 /* loop Number of 50us */
#define MTL_T_100US 1000 /* loop Number of 100us */
#define MTL_T_200US 2000 /* loop Number of 200us */
#define MTL_T_300US 3000 /* loop Number of 300us */
#define MTL_T_400US ( MTL_T_200US * 2 ) /* loop Number of 400us */
#define MTL_T_1MS 10000 /* 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、10 μ s を設定しています。

マルチメディアカードを使用する場合、100kHz 通信を想定し、10 μ s 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。必要な定義を 1 つ選んで設定してください。下記は、SIO_OPTION_1 を選択した場合の例です。表 6-1 動作モードに機能を示します。

```
/*-----*/
/* Define the combination of the MCU's resources. */
/*-----*/
#define SIO_OPTION_1      /* */ /* SI/O */
//#define SIO_OPTION_2    /* */ /* SI/O + CRC */
//#define SIO_OPTION_3    /* */ /* SI/O + S/W CRC */
```

表 6-1 動作モード

#define 定義	動作モード		
	SI/O (SCI)	CRC 演算 (MCU 内蔵演算器)	CRC 演算 (ソフトウェア処理)
SIO_OPTION_1		-	-
SIO_OPTION_2			-
SIO_OPTION_3		-	

SIO_OPTION_1 ~ SIO_OPTION_3 を選択した場合、データ受信 (full) を確認し、データ出力を行った後、次データの受信を行うことから、オーバランエラーの発生しない、安全な受信を行うことができます。連続受信中のデータエンディアンの変換処理を、次のダミーデータ書き込み後に行い、極力データ受信中にソフトウェア処理が行われる仕様としています。

MSB ファーストでの CRC-CCITT 演算処理のために MCU 内蔵 CRC 演算器を使う場合、SIO_OPTION_2 を指定してください。

MSBファーストでのCRC-CCITTソフトウェア演算処理を行う場合、SIO_OPTION_3を指定してください。

(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 演算処理を使用しないため、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SCIチャンネルの定義

使用する SCI チャンネル番号を指定してください。

```
/*-----*/
/* Define the SCI channel.                                   */
/*-----*/
#define SIO_SCI_CHANNEL      2          /* SCI Channel Select */
```

(4) 使用する端子の定義

使用するシリアル端子の定義を示します。

表 6-2 使用する端子の定義一覧を参考に使用する端子番号を指定してください。

```

/*-----*/
/* Define the control port.                                */
/*-----*/
/* Set to use port numbers and bit numbers */
#define SIO_DATAI_PORTNO    1          /* SIO DataIn Port No.          */
#define SIO_DATAI_BITNO     2          /* SIO DataIn Bit No.           */
#define SIO_CLK_PORTNO      1          /* SIO CLK Port No.             */
#define SIO_CLK_BITNO       1          /* SIO CLK Bit No.              */
#define SIO_DATAO_PORTNO    1          /* SIO DataOut Port No.         */
#define SIO_DATAO_BITNO     3          /* SIO DataOut Bit No.          */

```

表 6-2 使用する端子の定義一覧

#define 定義	設定値
SIO_DATAI_PORTNO	DataIn 端子のポート番号
SIO_DATAI_BITNO	DataIn 端子のビット番号
SIO_CLK_PORTNO	CLK 端子のポート番号
SIO_CLK_BITNO	CLK 端子のビット番号
SIO_DATAO_PORTNO	DataOut 端子のポート番号
SIO_DATAO_BITNO	DataOut 端子のビット番号

(5) 使用するポートファンクションレジスタ (PFxSCI) のSCIxSビットの定義

使用するシリアル端子に合わせて、ポートファンクションレジスタの SCIxS ビット (SCI 端子選択ビット) を設定してください。

```
/*-----*/
/* Define the control SCIxS Bit (SCI Pin Select) of Port Function Control Register*/
/*-----*/
#define SIO_SCI_SELECT      (uint8_t)(0)    /* SCI pin select set.(Set '0'or'1')*/
```

インライン関数 SIO_SIO_TX_ENABLE()と SIO_TRX_ENABLE()では、PFxSCI を定義することができます。使用する場合はコメントを外し、定義してください。ただし、PFxSCI の設定が可能な SCI チャネルは 1~3 及び 6 です。

```
#pragma inline(SIO_TX_ENABLE)
static void SIO_TX_ENABLE(void)          /* SIO Transmission Enable Setting */
{
    volatile uint8_t ucTmp;

    SIO_IR_CLEAR();                      /* SCI IR Clear */
    /* If SCI channel "1,2,3,6" is used, define the following PFC. */
    SIO_PFC_SCI    = SIO_SCI_SELECT;     /* Selects SCIn Pin. */

#pragma inline(SIO_TRX_ENABLE)
static void SIO_TRX_ENABLE(void)/*SIO Transmission&Reception Enable Setting*/
{
    volatile uint8_t ucTmp;

    SIO_IR_CLEAR();                      /* SCI IR Clear */
    /* If SCI channel "1,2,3,6" is used, define the following PFC. */
    SIO_PFC_SCI    = SIO_SCI_SELECT;     /* Selects SCIn Pin. */
```

(6) ソフトタイマの定義

本サンプルコードのみで使用するソフトタイマを設定してください。

設定値は、0.1us 以上となる値を設定してください。

```
/*-----*/
/* Define the wait time for timeout. */
/* Time out is occurred after 50000 times loop process of wait time. */
/*-----*/
#define SIO_T_SCI_WAIT      (uint16_t)(1) /* 0.1us wait When CPU clock = 96MHz*/
```

(7) プルアップ抵抗コントロールレジスタ (PCR) の定義

インライン関数 SIO_DATAI_INIT()では、PCR を定義することができます。使用する場合はコメントを外してください。ただし、PCR の設定が可能な端子は、Port9,PortA ~ PortE 及び PortG です。

```
/*----- DataIn control -----*/
#pragma inline(SIO_DATAI_INIT)
static void SIO_DATAI_INIT(void)          /* DataIn Initial Setting */
{
    SIO_ICR_DATAI    = 1;                  /* DataIn Input Buffer : Enable */
    /* SIO_PCR_DATAI = 0; */              /* DataIn Input Pull-up: off */
    SIO_DDR_DATAI    = SIO_IN;            /* DataIn Input */
}
```

(8) オープンドレインコントロールレジスタ (ODR) の定義

インライン関数 SIO_DATAO_INIT()と SIO_DATAO_CLK()では、ODR を定義することができます。使用する場合はコメントを外してください。ただし、ODR の設定が可能な端子は、Port0 ~ Port3 及び PortC です。

```
/*----- DataOut control -----*/
#pragma inline(SIO_DATAO_INIT)
static void SIO_DATAO_INIT(void)          /* DataOut Initial Setting */
{
    /* SIO_ODR_DATAO = 0; */              /* Open Drain Control: CMOS*/

/*----- CLK control -----*/
#pragma inline(SIO_CLK_INIT)
static void SIO_CLK_INIT(void)           /* CLK Initial Setting */
{
    /* SIO_ODR_CLK   = 0; */              /* Open Drain Control: CMOS*/
}
```


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。

7.4 CRC演算器のモジュールストップ設定について (オプション)

CRC 演算処理を使用した関数する場合、初期化処理にて、モジュールストップ解除を行います。モジュールストップ設定を行いません。モジュールストップ設定が必要な場合は、ユーザプログラムで制御してください。

7.5 入力バッファコントロールレジスタ (PORTn.ICR) に設定について

本サンプルコードでは、SCI 以外の周辺モジュールの設定を行っていないため、当該端子に割り当てられている周辺モジュールの設定にて、入力機能を無効にした状態で、本サンプルコードを使用してください。

入力機能を無効にしていない状態で、PORTn.ICR レジスタの設定を変更する場合、端子の状態によっては内部にエッジが発生し、意図しない動作をすることがあります。

7.6 コンパイルオプションについて

最適化レベル「2」設定、かつ、最適化方法「サイズ優先」設定での動作は、確認済です。

最適化レベル「2」指定、かつ、最適化方法「スピード優先」設定での動作は、未確認です。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

<http://japan.renesas.com/>

お問合せ先

<http://japan.renesas.com/inquiry>

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.01	2012.06.15	—	初版発行

すべての商標および登録商標は、それぞれの所有者に帰属します。

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）がありません。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っていると、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。型名が異なる製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したのですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、
家電、工作機械、パーソナル機器、産業用ロボット等
高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、
防災・防犯装置、各種安全装置等
当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関し、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍用用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社と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/contact/>