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

要旨

本アプリケーションノートではRXファミリ クワッドシリアルペリフェラルインタフェース（以下、QSPIと略す）をクロック同期方式で使用したクロック同期式シングルマスタ制御モジュールとその使用方法を説明します。本モジュールは、Firmware Integration Technology （以下、FIT と略す）を使ったクロック同期式シングルマスタ制御モジュールです。以降、本モジュールを QSPI FIT モジュールと称します。また、同様に FIT を使った他の機能制御モジュールを FIT モジュールもしくは“機能名”FIT モジュールと表します。

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

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

対象デバイス

対応 MCU

RX64M グループ、RX65N グループ、RX651 グループ、RX66N グループ

RX71M グループ、RX72M グループ、RX72N グループ

動作確認に使用したデバイス

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

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

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

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

対象コンパイラ

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

各コンパイラの動作確認内容については「6.1 動作確認環境」を参照してください。

関連ドキュメント

QSPI FIT モジュールに関連するアプリケーションノートを以下に示します。併せて参照してください。

- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- RX Family LONGQ Module Using Firmware Integration Technology（R01AN1880）
- RX ファミリ DMA コントローラ DMACA 制御モジュール Firmware Integration Technology（R01AN2063）
- RX Family DTC Module Using Firmware Integration Technology（R01AN1819）
- RX ファミリ コンペアマッチタイマ (CMT)モジュール Firmware Integration Technology（R01AN1856）
- RX ファミリ EEPROM アクセス クロック同期式制御モジュール Firmware Integration Technology（R01AN2325）
- RX Family General Purpose Input/Output Driver Module Using Firmware Integration Technology（R01AN1721）
- RX Family Multi-Function Pin Controller Module Using Firmware Integration Technology（R01AN1724）

目次

1. 概要

1.1 QSPI FIT モジュールとは

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

また、QSPI FIT モジュールは API として、プロジェクトに組み込んで使用します。QSPI FIT モジュールの組み込み方については、「2.12 FITモジュールの追加方法」を参照してください。

1.2 QSPI FIT モジュールの概要

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

QSPI の QSSL 端子をポート制御によりスレーブデバイスセレクト端子に割り当てることが可能です。

表 1-1使用する周辺機器と用途を、図 1.1　使用例に使用例を示します。

以下に、機能概略を示します。

- マスタデバイスを RX ファミリ MCU とする QSPI 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
- クロック同期式の Single-SPI／Dual-SPI／Quad-SPI モードをサポート
- QSPI をクロック同期式シリアル通信方法で使用。ユーザ設定した 1 チャネルもしくは複数チャネルの制御が可能
- 異なるチャネルからリエントラントが可能
- スレーブデバイスセレクト制御を未サポート

  別途ポート制御によるスレーブデバイスセレクト制御の組み込みが必要です。
- ビッグエンディアン／リトルエンディアンでの動作が可能
- MSB ファーストフォーマットで転送
- ソフトウェア転送のみをサポート。

  DMACもしくはDTC転送を行う場合、別途DMACもしくはDTC転送プログラムが必要です。

表 1-1　使用する周辺機器と用途

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

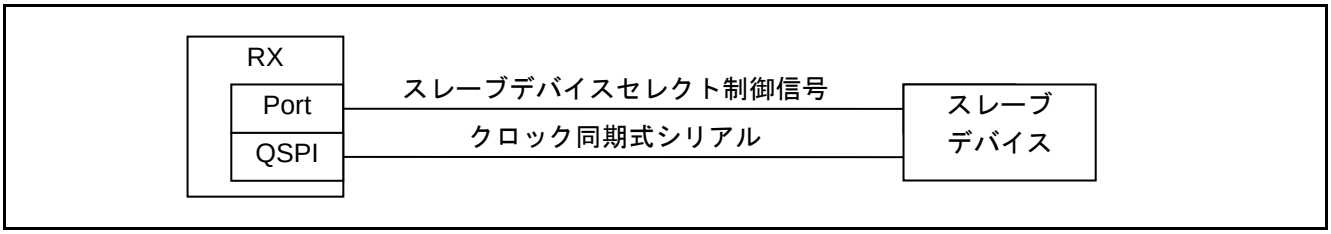


図 1.1　使用例

1.3 API の概要

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

表 1-2 API 関数

関数名	説明
R_QSPI_SMstr_Open()	ドライバの初期化処理
R_QSPI_SMstr_Close()	ドライバの終了処理
R_QSPI_SMstr_Control()	ドライバのコントロール（SPI モードとビットレート）設定処理
R_QSPI_SMstr_Write()	シングルマスタ送信（Single／Dual／Quad-SPI 動作）処理
R_QSPI_SMstr_Read()	シングルマスタ受信（Single／Dual／Quad-SPI 動作）処理
R_QSPI_SMstr_Get_BuffRegAddress()	SPDR レジスタアドレス取得処理
R_QSPI_SMstr_Int_Spti_Ier_Clear()	SPTI 送信割り込み要求禁止処理
R_QSPI_SMstr_Int_Spri_Ier_Clear()	SPRI 受信割り込み要求禁止処理
R_QSPI_SMstr_Int_Spti_Dmacdtc_Flag_Set()	DMAC/DTC 送信完了フラグ設定処理
R_QSPI_SMstr_Int_Spri_Dmacdtc_Flag_Set()	DMAC/DTC 受信完了フラグ設定処理
R_QSPI_SMstr_GetVersion()	ドライバのバージョン情報取得処理
R_QSPI_SMstr_Set_LogHdlAddress()	LONGQ FIT モジュールのハンドラアドレス設定処理
R_QSPI_SMstr_Log()	LONGQ FIT モジュールを使ったエラーログ取得処理
R_QSPI_SMstr_1ms_Interval()　注２	インターバルタイマカウント処理

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

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

1.4　処理例

1.4.1　ハードウェア設定

(1)　ハードウェア構成例

図 **1.2**に接続図を示します。なお、高速で動作させた場合を想定し、各信号ラインの回路的マッチングを取るためのダンピング抵抗やコンデンサの付加を検討してください。

必ずプルアップ処理を行ってください。プルアップ処理が無い場合、データ線が Hi-Z 時に、スレーブデバイスがライトプロテクト状態やホールド状態になる可能性があります。

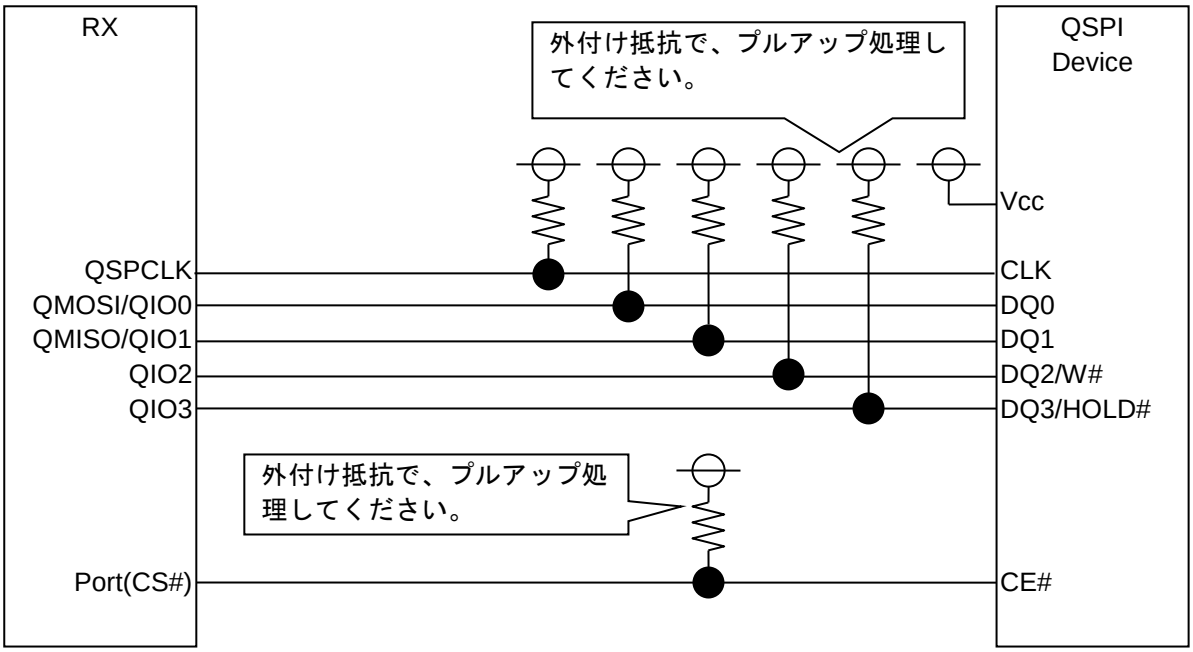


図 1.2 接続図

(2)　使用端子一覧

表 1-3に、使用端子と機能を示します。

表 1-3　使用端子と機能

端子名	入出力	内容
QSPCLK	出力	クロック出力
QMOSI/QIO0（注１）	入出力	マスタデータ出力／データ 0
QMISO/QIO1（注２）	入出力	マスタデータ入力／データ 1
QIO2（注３）	入出力	データ 2
QIO3（注３）	入出力	データ 3
図 1.2 記載の Port(CS#)	出力	スレーブデバイスセレクト出力 ただし、QSPI FIT モジュールでは、扱いません。

注１：以下、QIO0 と略す。
注２：以下、QIO1 と略す。
注３：使用しない場合でも、端子を割り当てる必要があります。本端子は、他の周辺機能で使用することはできません。

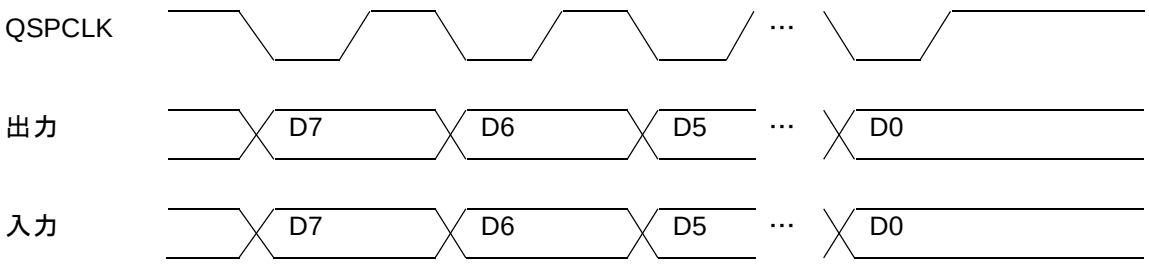
1.4.2　ソフトウェア説明

(1)　動作概要

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

(2)　制御可能なスレーブデバイス

制御可能なスレーブデバイスは、図 1.3に示す SPI モード 3（CPOL=1、CPHA=1）をサポートしたスレーブデバイスです。図は、入出力信号の極性と位相のタイミングを示すものであり、ビット幅を示すものではありません。



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

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

・MSB ファーストでの転送

転送を行っていないときの QSPCLK 端子のレベルは、”H”です。

図 1.3　制御可能なスレーブデバイスのタイミング

使用可能なシリアルクロック周波数は、MCU のユーザーズマニュアル ハードウェア編およびスレーブデバイスのデータシートで、確認してください。

(3)　スレーブデバイスの CE#端子制御

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

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

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

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

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

(4)　ソフトウェア構成

図 1.4にソフトウェア構成を示します。

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

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

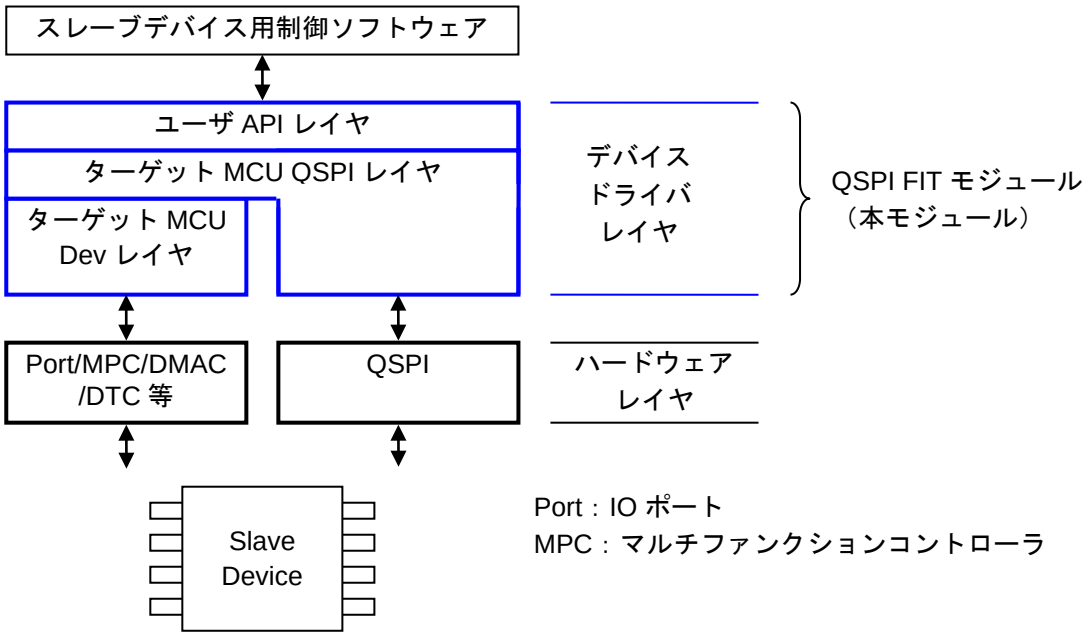


図 1.4　ソフトウェア構成

(a)　ユーザ API レイヤ（r_qspi_smstr.c）

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

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

(b)　ターゲット MCU QSPI レイヤ（r_qspi_smstr_target.c）

QSPI のリソース制御部分です。

チャネル数や QSPI の仕様の違い等を MCU 毎に提供します。

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

QSPI 以外のリソース制御部分です。

IO ポート／マルチファンクションピンコントローラ等の制御部分です。他の FIT モジュール等を使用してシステムに合わせて組み込みが必要です。

(d)　スレーブデバイス用制御ソフトウェア

例として「RX ファミリ EEPROM アクセス クロック同期式モジュール Firmware Integration Technology（R01AN2325）」を参照してください。この Serial EEPROM 制御ソフトウェアには、FIT モジュールとの合わせ込みのためのドライバ I/F 関数（r_eeprom_spi_drvif_devX.c：X=0or1）があります。

(5)　データバッファと送信／受信データの関係

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

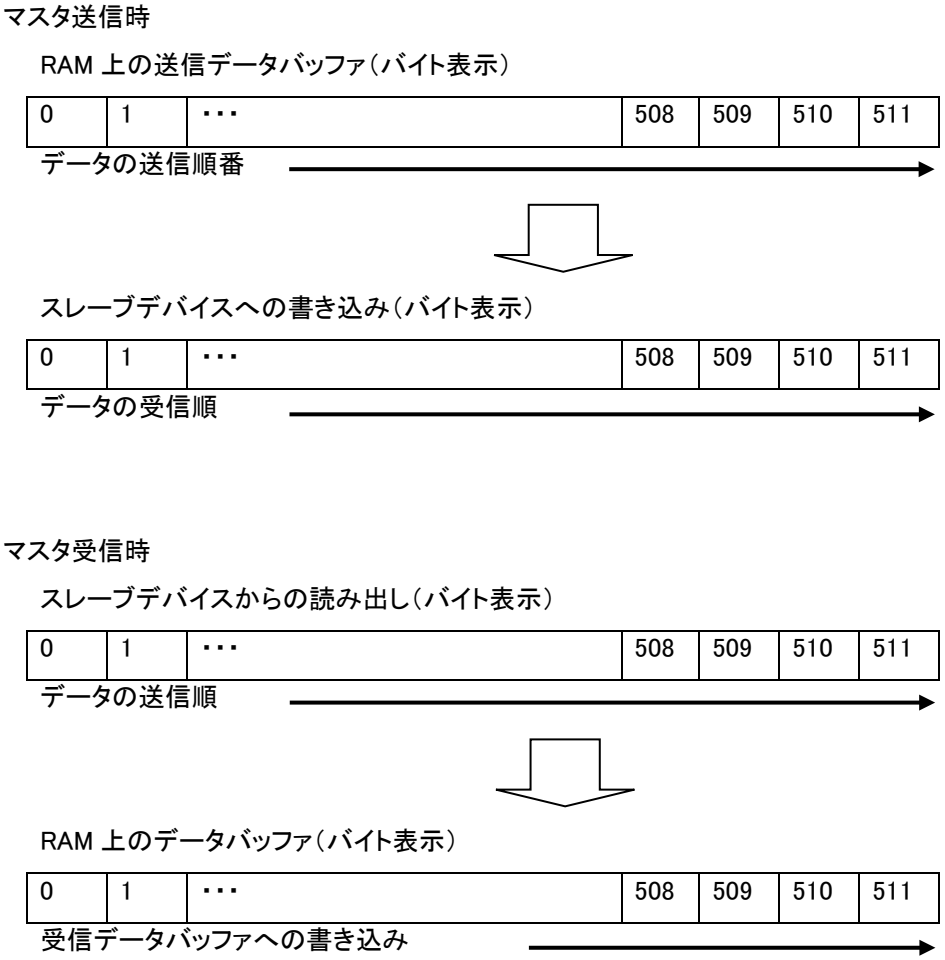


図 1.5　データバッファと送信／受信データの関係

(6)　シングルマスタ送信処理

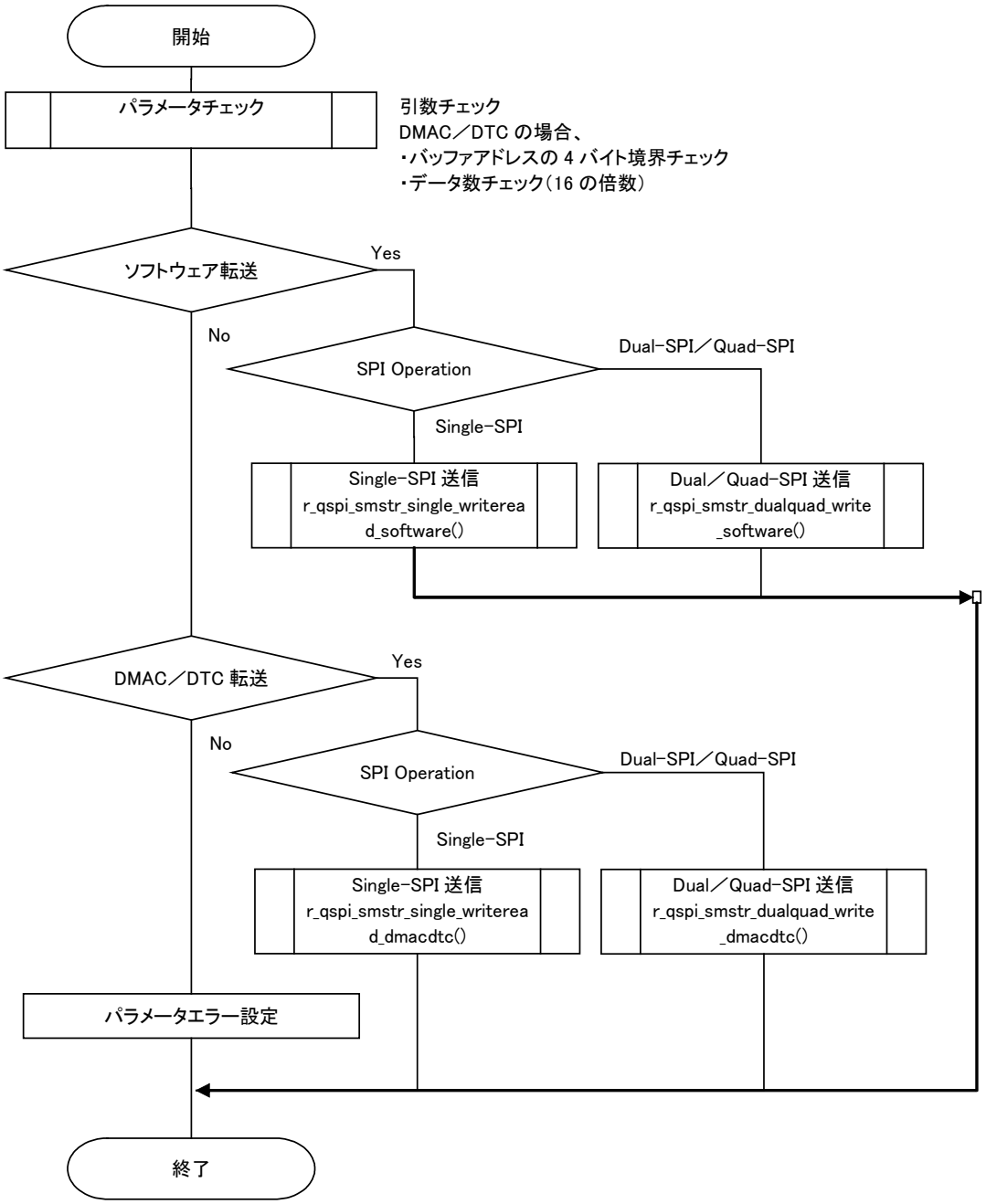


図 1.6　データ送信処理

(a)　Single-SPI 送信処理（ソフトウェア）

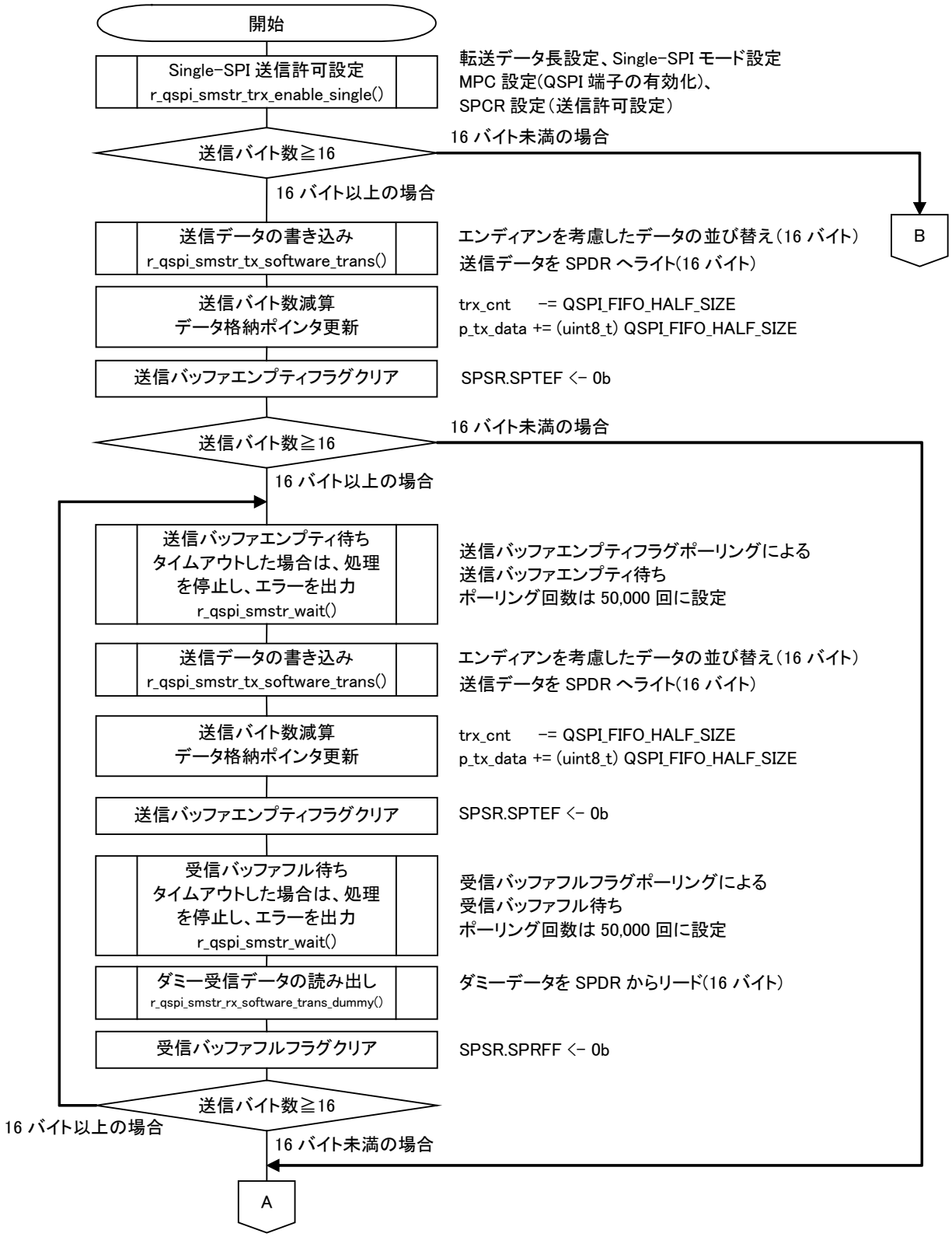


図 1.7　Single-SPI 送信処理 1（ソフトウェア）

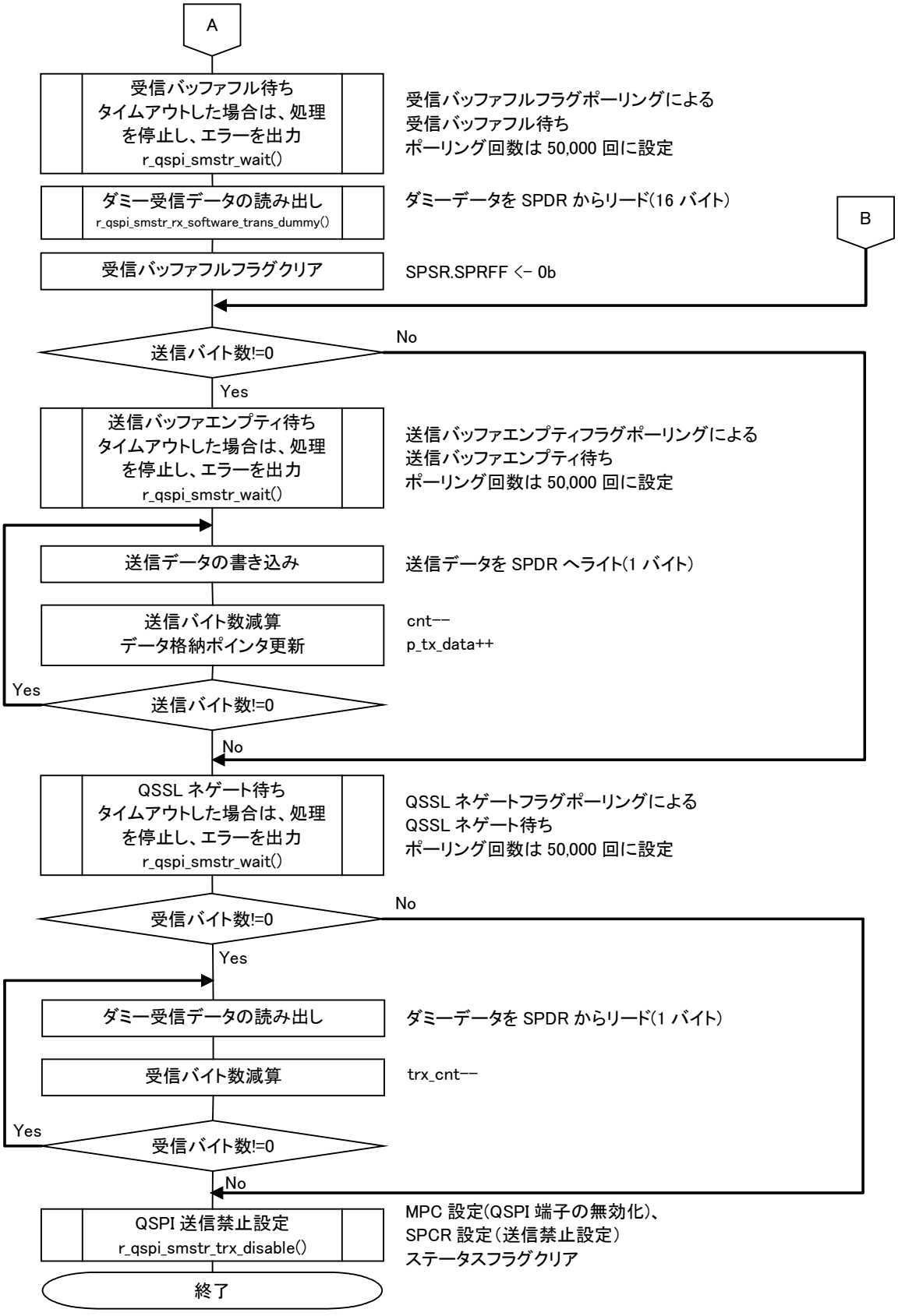


図 1.8 Single-SPI 送信処理 2（ソフトウェア）

(b)　Dual-SPI／Quad-SPI 送信処理（ソフトウェア）

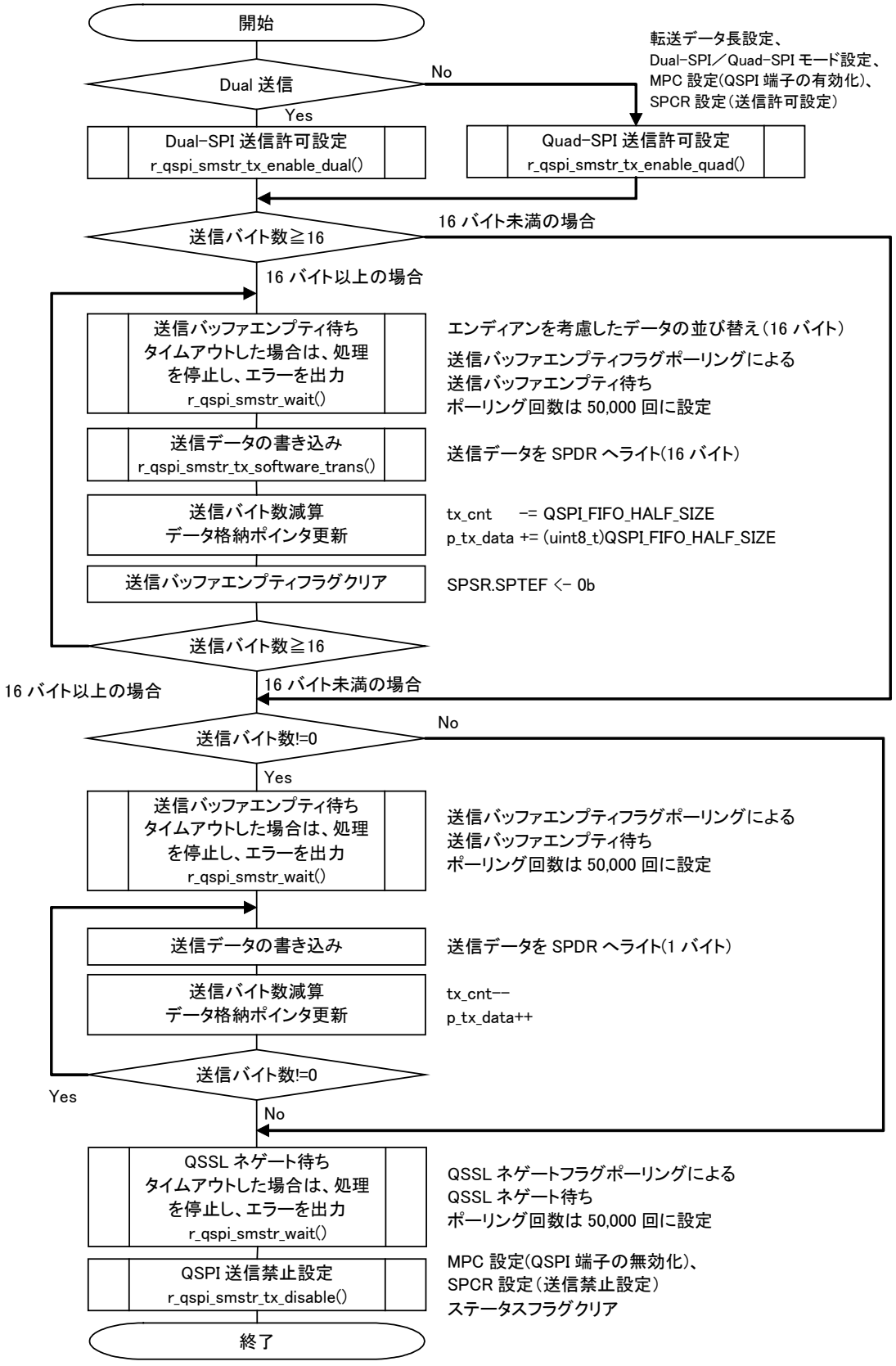


図 1.9　Dual-SPI／Quad-SPI 送信処理（ソフトウェア）

(c)　Single-SPI 送信処理（DMAC／DTC）

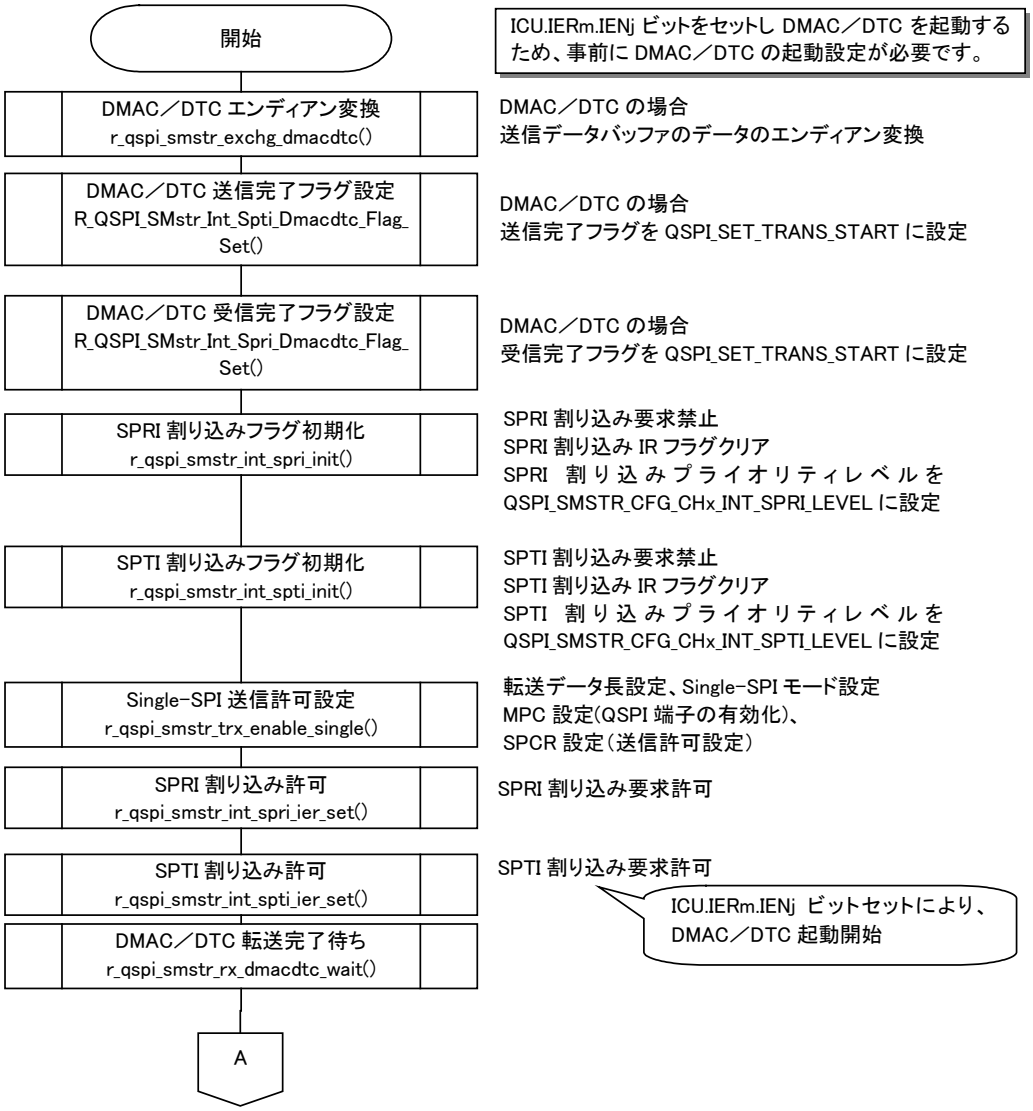


図 1.10　Single-SPI 送信処理 1（DMAC／DTC）

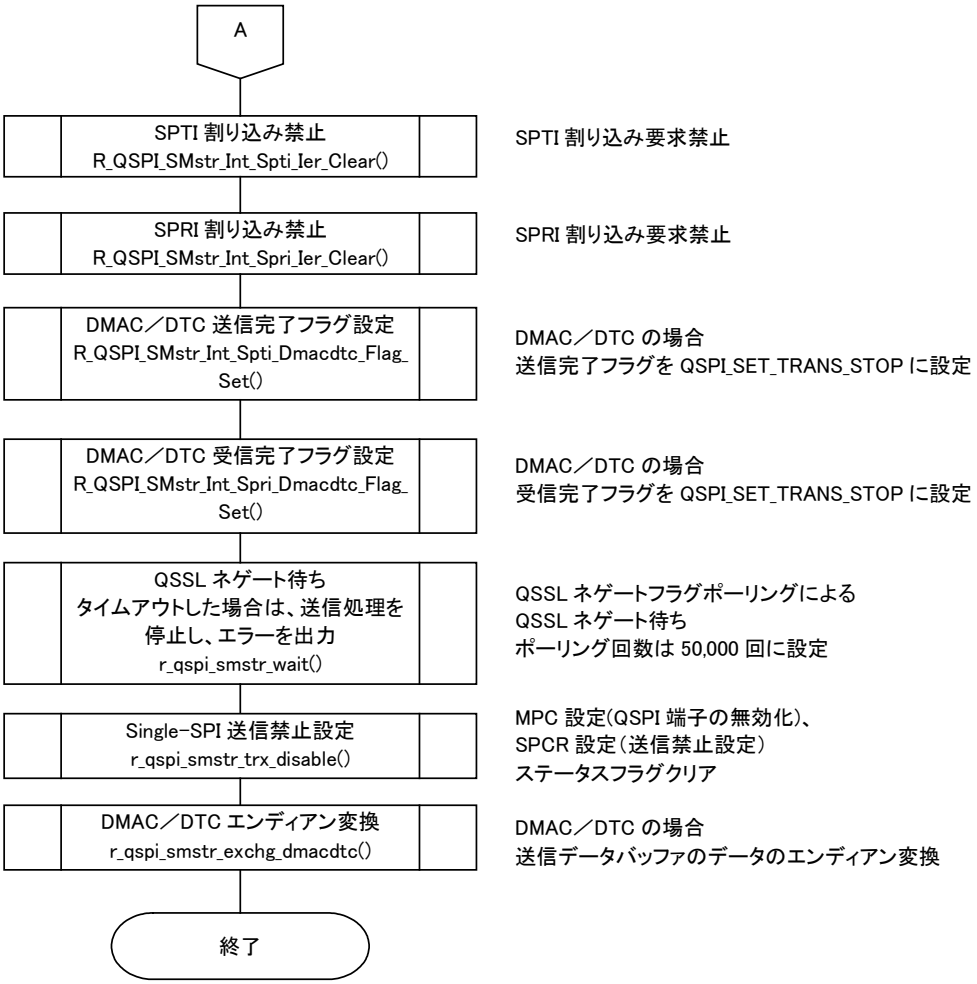


図 1.11　Single-SPI 送信処理 2（DMAC／DTC）

(d)　Dual-SPI／Quad-SPI 送信処理（DMAC／DTC）

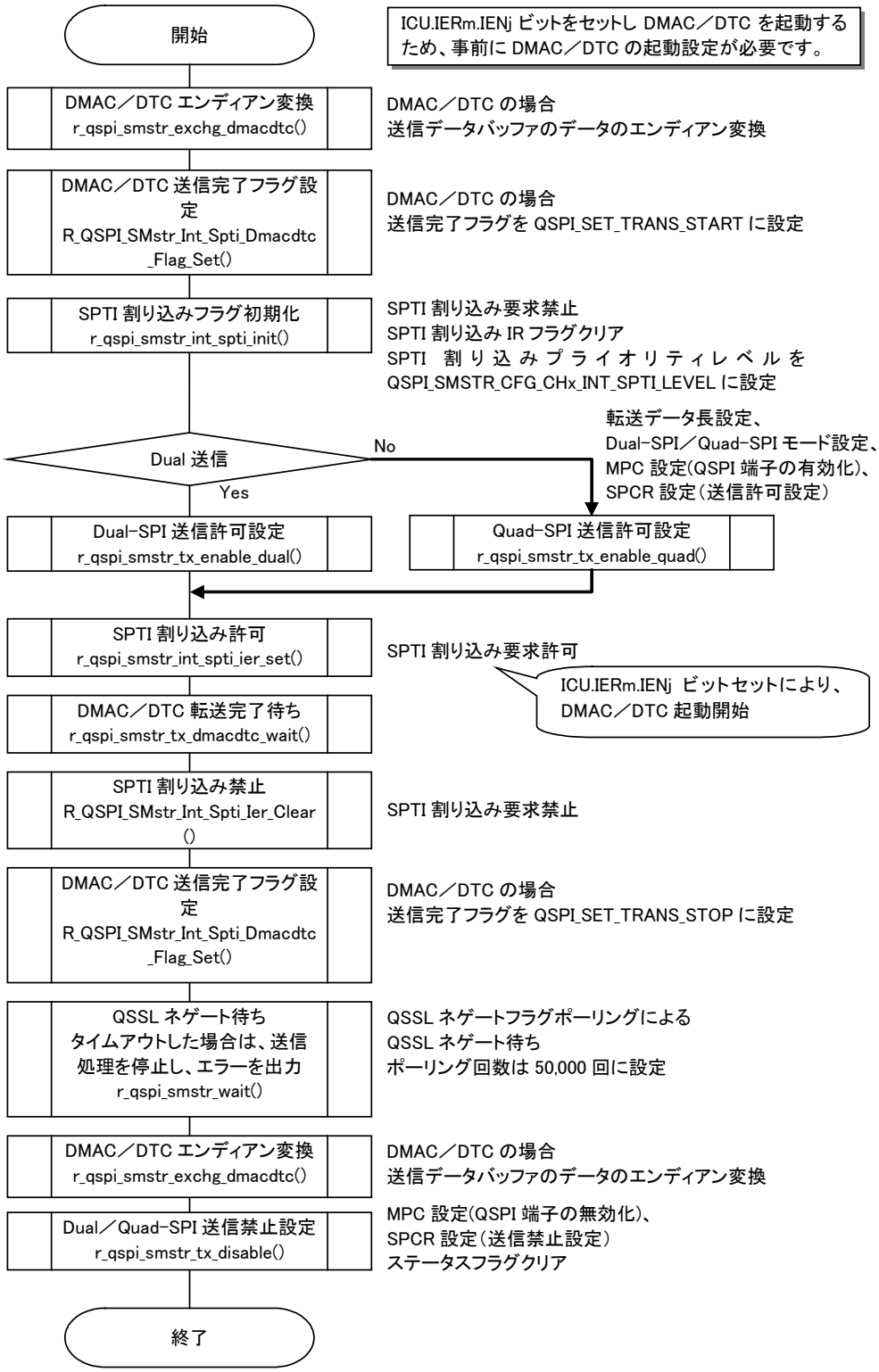


図 1.12　Dual-SPI／Quad-SPI 送信処理（DMAC／DTC）

(7)　シングルマスタ受信処理

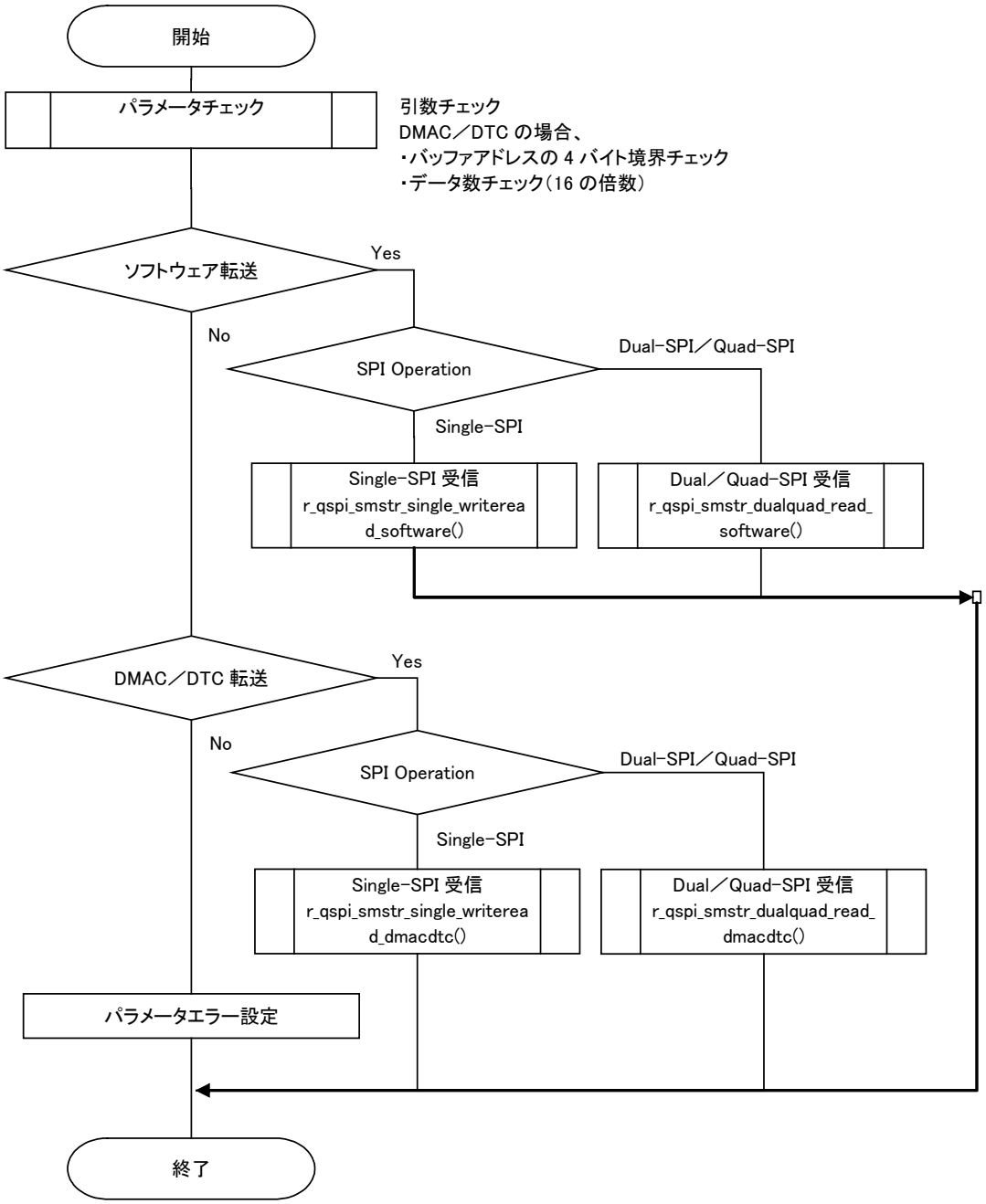


図 1.13　データ受信処理

(a)　Single-SPI 受信処理（ソフトウェア）

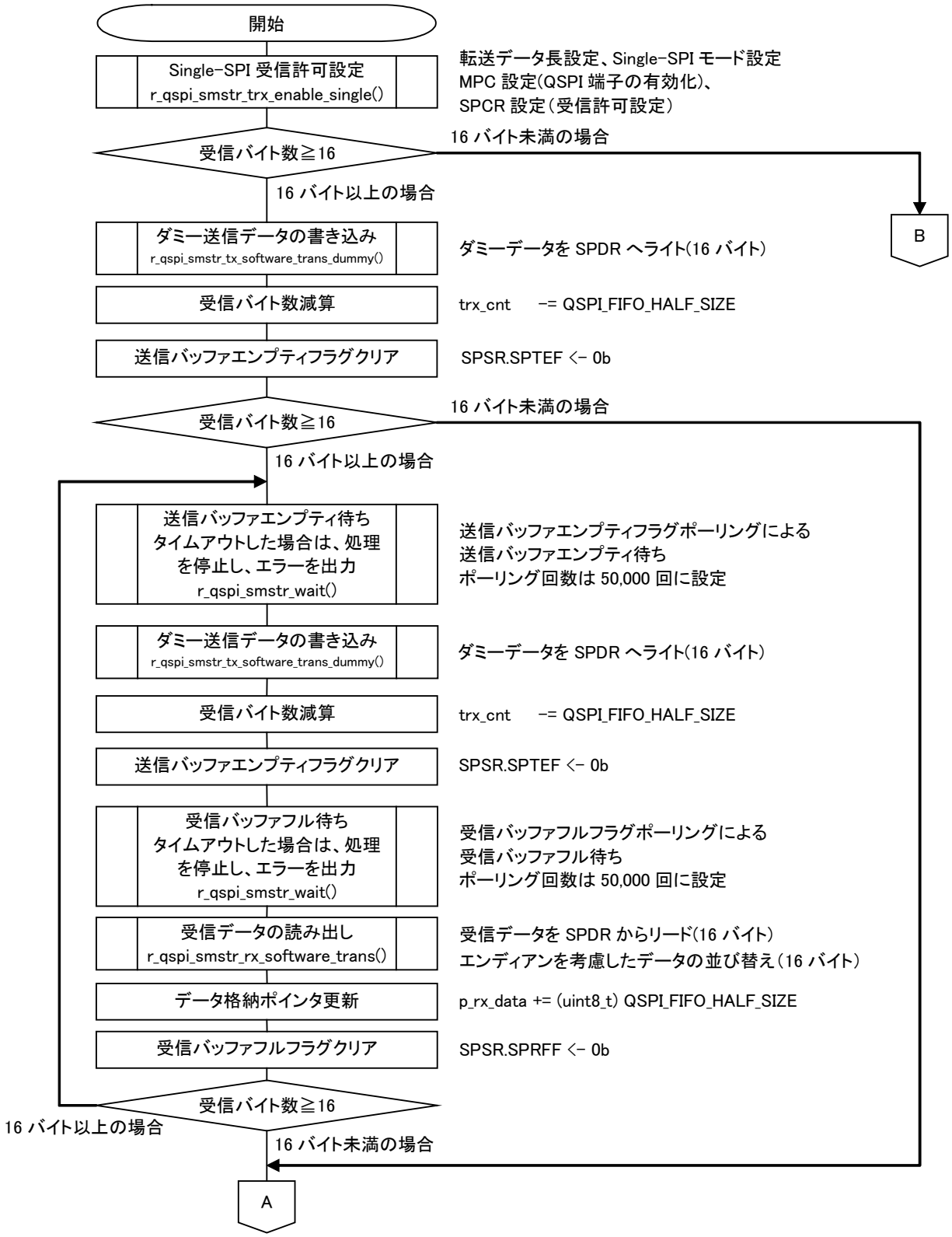


図 1.14　Single-SPI 受信処理 1（ソフトウェア）

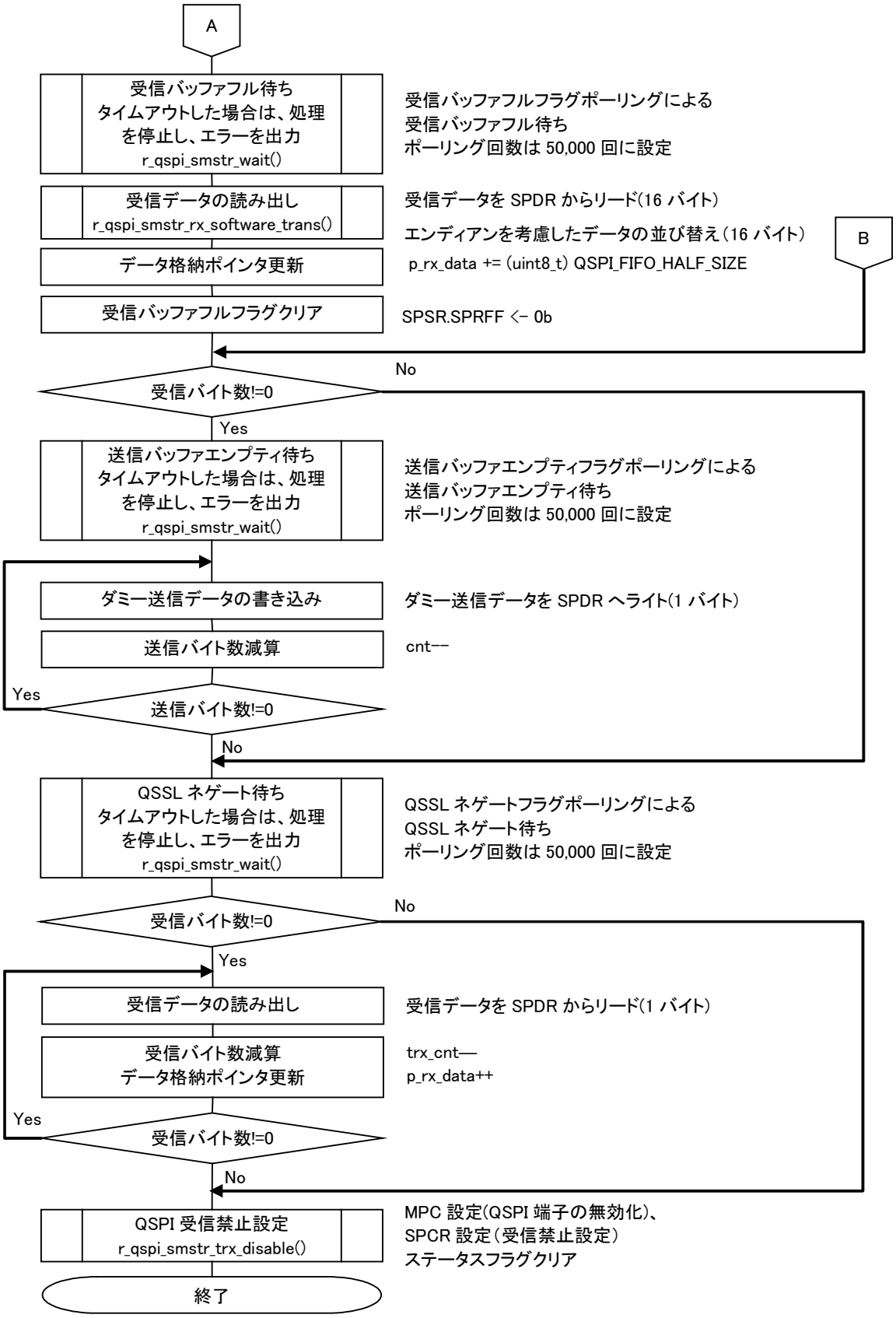


図 1.15 Single –SPI 受信処理 2（ソフトウェア）

(b)　Dual-SPI／Quad-SPI 受信処理（ソフトウェア）

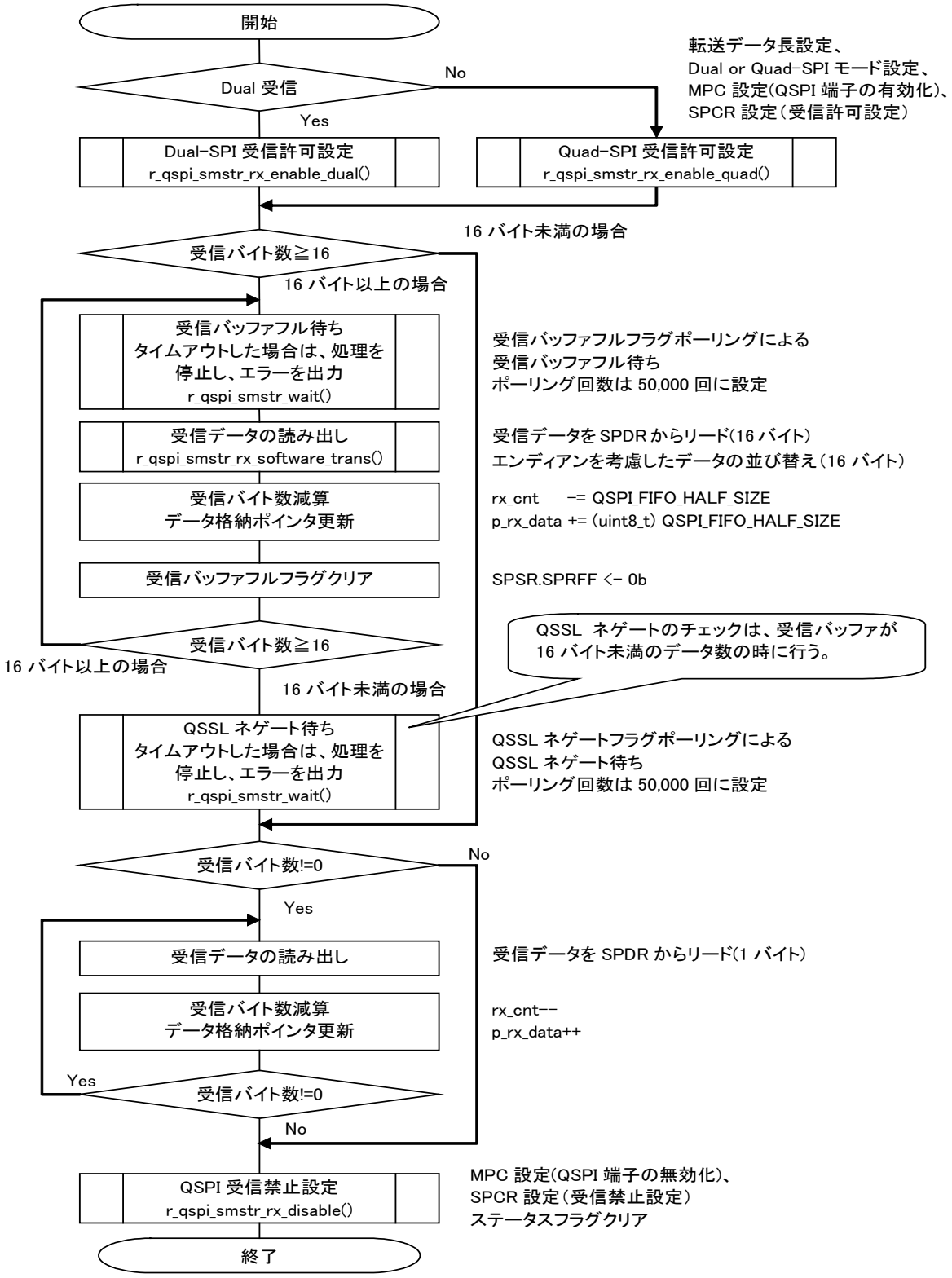


図 1.16　Dual-SPI／Quad-SPI 受信処理（ソフトウェア）

(c)　Single-SPI 受信処理（DMAC／DTC）

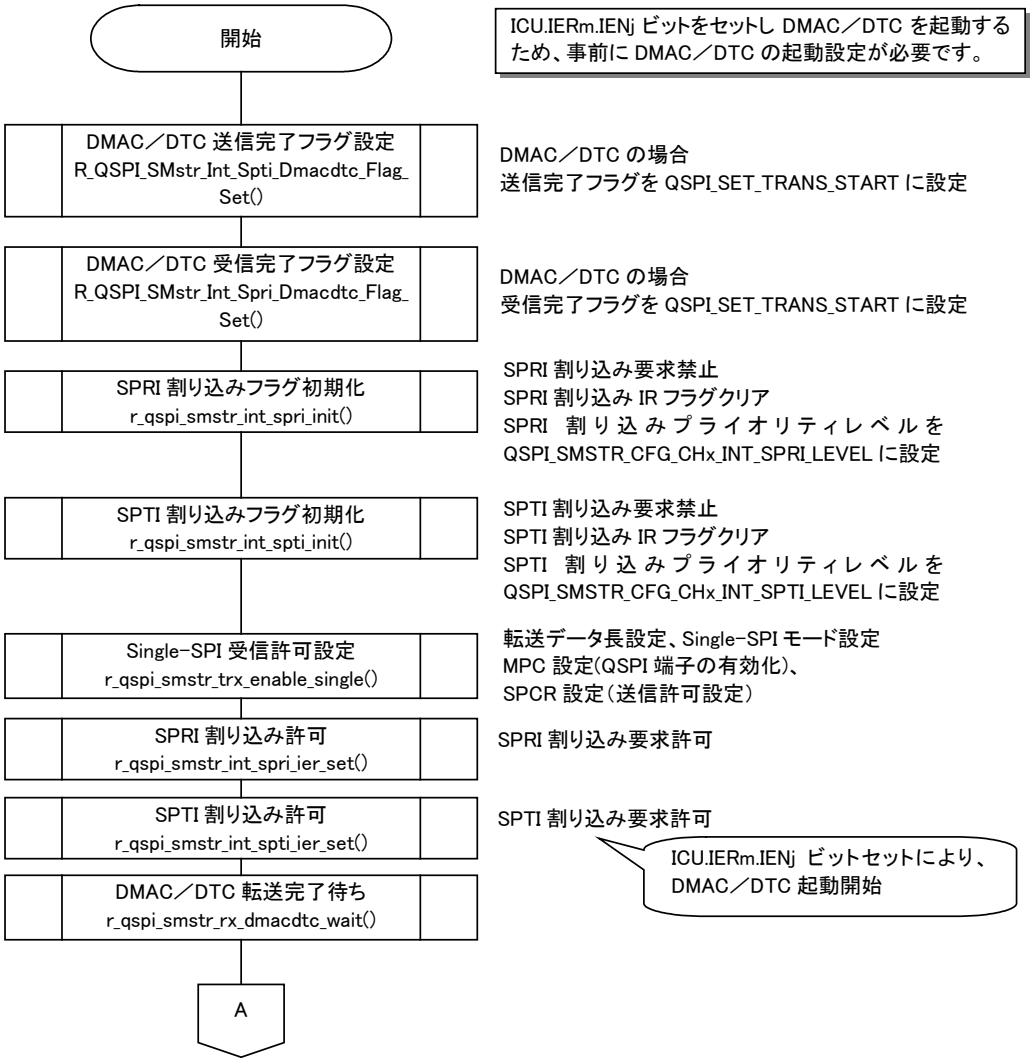


図 1.17　Single-SPI 受信処理 1（DMAC／DTC）

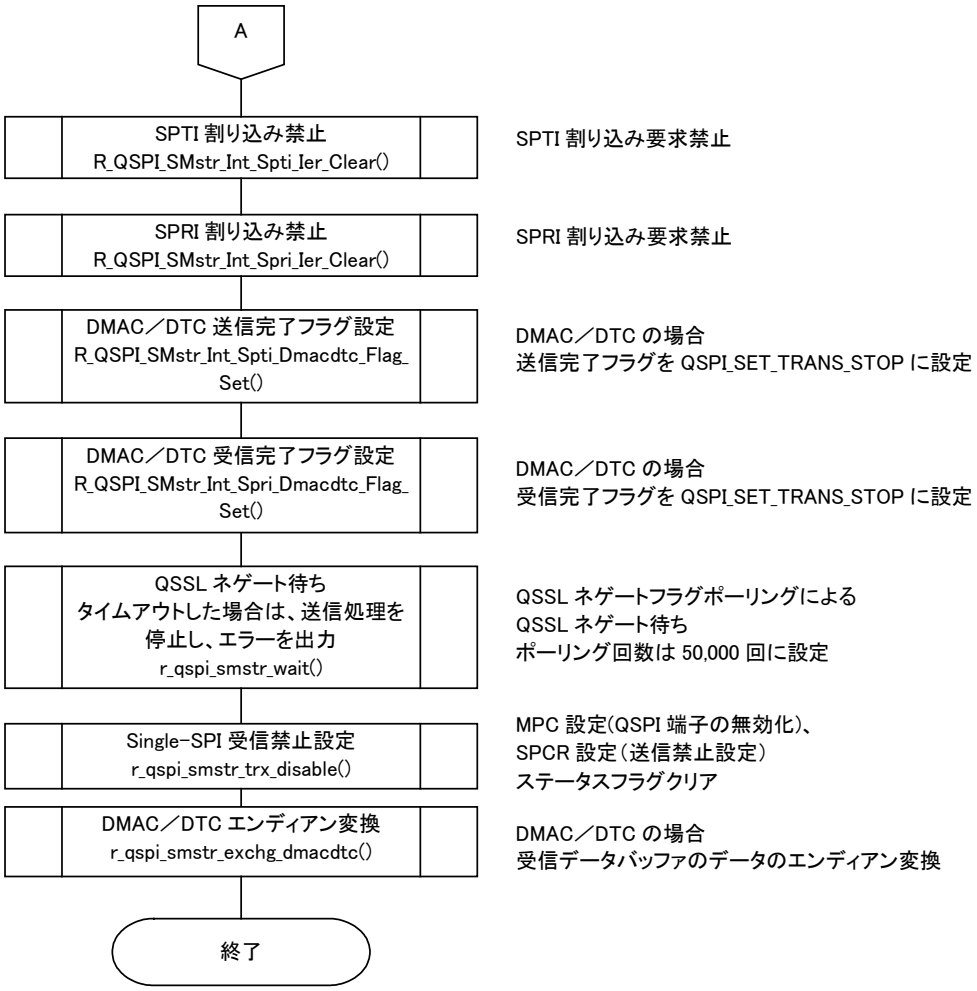


図 1.18　Single-SPI 受信処理 2（DMAC／DTC）

(d)　Dual-SPI／Quad-SPI 受信処理（DMAC／DTC）

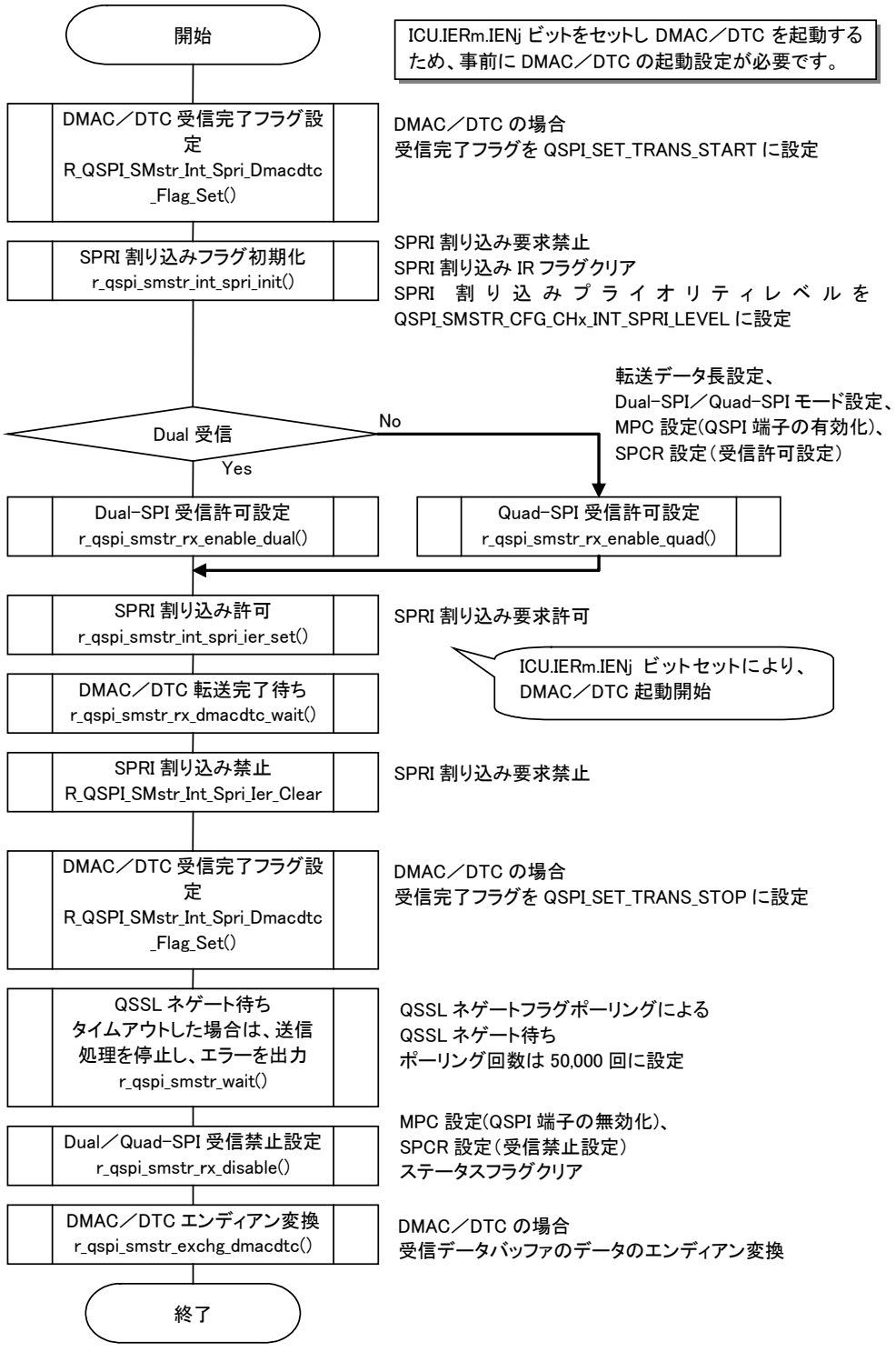


図 1.19　Dual-SPI／Quad-SPI 受信処理（DMAC／DTC）

1.5　状態遷移図

図 1.20に本モジュールの状態遷移図を示します。

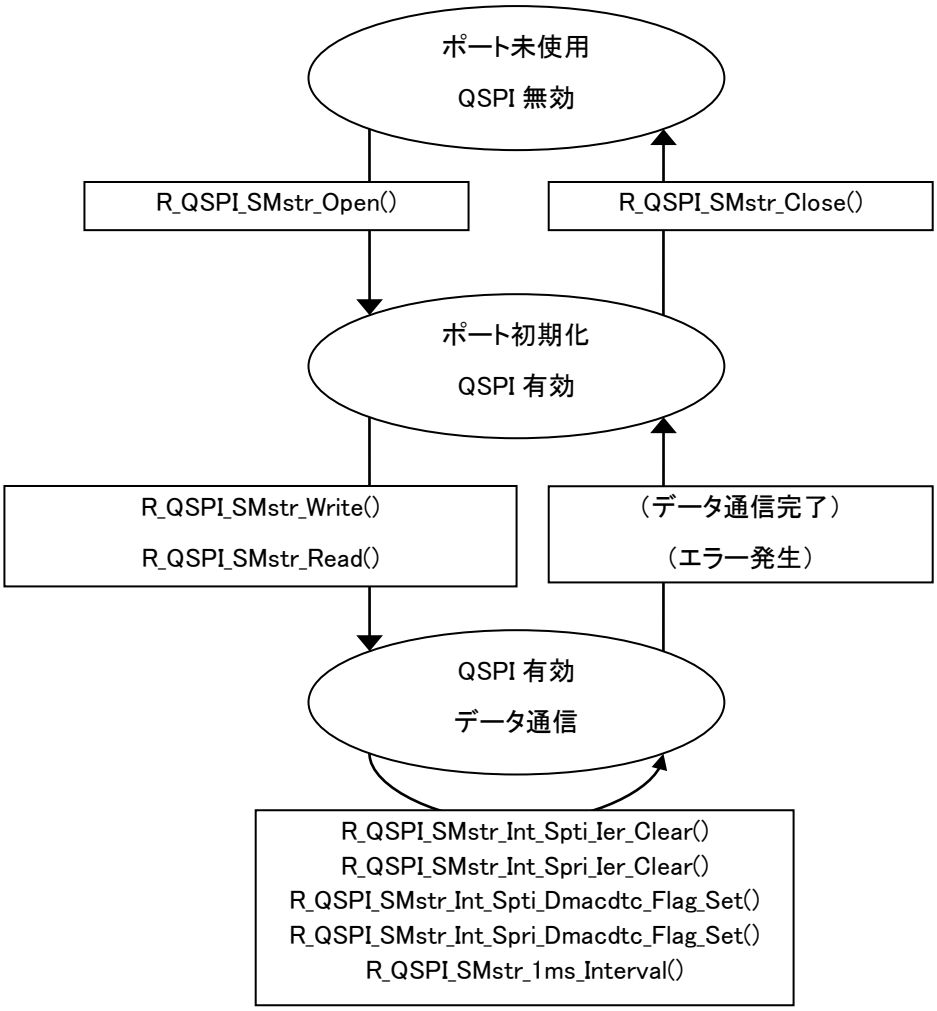


図 1.20　状態遷移図

2. API 情報

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

2.1 ハードウェアの要求

ご使用になる MCU が以下の機能をサポートしている必要があります。

- QSPI

2.2 ソフトウェアの要求

このドライバは以下のパッケージに依存しています。

- r_bsp Rev.5.20 以上
- r_dmaca_rx（DMACA FIT モジュールを用いて、DMAC 転送を使用する場合のみ）
- r_dtc_rx（DTC FIT モジュールを用いて、DTC 転送を使用する場合のみ）
- r_cmt_rx（DMAC 転送もしくは DTC 転送を使用し、かつコンペアマッチタイマ CMT FIT モジュールを使用する場合のみ）
  他タイマやソフトウェアタイマで代用できます。
- r_gpio_rx（GPIO, MPC FIT モジュールを用いて、GPIO を制御する場合のみ）
- r_mpc_rx（GPIO, MPC FIT モジュールを用いて、MPC を制御する場合のみ）

2.3 サポートされているツールチェーン

QSPI FIT モジュールは、「6.1動作確認環境」に示すツールチェーンで動作確認を行っています。

2.4 使用する割り込みベクタ

表 2-1に示す条件が成立した場合、API 関数内で対象の割り込みが有効になります。DMAC を使用する場合は、通信終了後 R_QSPI_SMstr_Int_Spti_Ier_Clear()関数、R_QSPI_SMstr_Int_Spri_Ier_Clear()関数を用いて、割り込みを禁止する必要があります。詳細は「2.13.3 DMAC／DTC」を参照してください。

表 2-1に本 FIT モジュールが使用する割り込みベクタを示します。

表 2-1　条件と有効になる割り込み

条件			有効になる割り込み
API 関数	引数 op_mode	引数 tran_mode	
R_QSPI_SMstr_Write()	QSPI_SMSTR_SINGLE_SPI_WRITE	QSPI_SMSTR_DMAC	送信バッファエンプティ（SPTI） 受信バッファフル（SPRI）
		QSPI_SMSTR_DTC	送信バッファエンプティ（SPTI） 受信バッファフル（SPRI）
	QSPI_SMSTR_DUAL_SPI	QSPI_SMSTR_DMAC	送信バッファエンプティ（SPTI）
		QSPI_SMSTR_DTC	送信バッファエンプティ（SPTI）
	QSPI_SMSTR_QUAD_SPI	QSPI_SMSTR_DMAC	送信バッファエンプティ（SPTI）
		QSPI_SMSTR_DTC	送信バッファエンプティ（SPTI）
R_QSPI_SMstr_Read()	QSPI_SMSTR_SINGLE_SPI_WRITE	QSPI_SMSTR_DMAC	送信バッファエンプティ（SPTI） 受信バッファフル（SPRI）
		QSPI_SMSTR_DTC	送信バッファエンプティ（SPTI） 受信バッファフル（SPRI）
	QSPI_SMSTR_DUAL_SPI	QSPI_SMSTR_DMAC	受信バッファフル（SPRI）
		QSPI_SMSTR_DTC	受信バッファフル（SPRI）
	QSPI_SMSTR_QUAD_SPI	QSPI_SMSTR_DMAC	受信バッファフル（SPRI）
		QSPI_SMSTR_DTC	受信バッファフル（SPRI）

表 2-2　使用する割り込みベクター覧

デバイス	割り込みベクタ
RX64M	SPRI 割り込み（ベクタ番号：42）
	SPTI 割り込み（ベクタ番号：43）
RX65N／RX651	SPRI 割り込み（ベクタ番号：42）
	SPTI 割り込み（ベクタ番号：43）
RX66N	SPRI 割り込み（ベクタ番号：42）
	SPTI 割り込み（ベクタ番号：43）
RX71M	SPRI 割り込み（ベクタ番号：42）
	SPTI 割り込み（ベクタ番号：43）
RX72M	SPRI 割り込み（ベクタ番号：42）
	SPTI 割り込み（ベクタ番号：43）
RX72N	SPRI 割り込み（ベクタ番号：42）
	SPTI 割り込み（ベクタ番号：43）

2.5 ヘッダファイル

すべての API 呼び出しと使用されるインタフェース定義は r_qspi_smstr_rx_if.h に記載しています。

ビルド毎の構成オプションは、r_qspi_smstr_rx_config.h で選択します。

```
#include "r_qspi_smstr_rx_if.h"
```

2.6 整数型

このプロジェクトは ANSI C99 を使用しています。これらの型は stdint.h で定義されています。

2.7 コンパイル時の設定

QSPI FIT モジュールのコンフィギュレーションオプションの設定は、r_qspi_smstr_rx_config.h と r_qspi_smstr_rx_pin_config.h で行います。

RX64M RSK をオプション名および設定値に関する説明を下表に示します。

Configuration options in *r_qspi_smstr_rx_config.h*	
#define QSPI_SMSTR_CFG_USE_FIT ※デフォルト値は“有効”	QSPI FIT モジュールの BSP 環境での使用有無を選択できます。 無効にした場合、r_bsp 等の FIT モジュール制御を無効します。また、別途、処理を組み込む必要があります。詳細は「2.12 FITモジュールの追加方法」を参照ください。 有効にした場合、r_bsp 等の FIT モジュール制御を有効にします。
#define QSPI_SMSTR_CFG_CHx_INCLUDED ※デフォルト値は“有効” ※“x”はチャネル番号	該当チャネルを使用するかを選択できます。 無効にした場合、該当チャネルに関する処理をコードから省略します。 有効にした場合、該当チャネルに関する処理をコードに含めます。
#define QSPI_SMSTR_CFG_LONGQ_ENABLE ※デフォルトは“無効”	デバッグ用のエラーログ取得処理を使用するか選択できます。 無効にした場合、処理をコードから省略します。 有効にした場合、処理をコードに含めます。 使用するためには、別途 FIT の LONGQ FIT モジュールが必要です。

Configuration options in *r_qspi_smstr_rx_pin_config.h*	
#define R_QSPI_SMSTR_CFG_QSPI_QSPCLK_PORT ※RX64M QSPI チャネル 0 のデフォルト値は “‘7’” ※“x”はチャネル番号	QSPI の QSPCLK 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QSPCLK_BIT ※RX64M QSPI チャネル 0 のデフォルト値は “‘7’” ※“x”はチャネル番号	QSPI の QSPCLK 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO0_PORT ※RX64M QSPI チャネル 0 のデフォルト値は “‘C’” ※“x”はチャネル番号	QSPI の QIO0 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO0_BIT ※RX64M QSPI チャネル 0 のデフォルト値は “‘3’” ※“x”はチャネル番号	QSPI の QIO0 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO1_PORT ※RX64M QSPI チャネル 0 のデフォルト値は “‘C’” ※“x”はチャネル番号	QSPI の QIO1 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO1_BIT ※RX64M QSPI チャネル 0 のデフォルト値は “‘4’” ※“x”はチャネル番号	QSPI の QIO1 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO2_PORT ※RX64M QSPI チャネル 0 のデフォルト値は “‘8’” ※“x”はチャネル番号	QSPI の QIO2 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO2_BIT ※RX64M QSPI チャネル 0 のデフォルト値は “‘0’” ※“x”はチャネル番号	QSPI の QIO2 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO3_PORT ※RX64M QSPI チャネル 0 のデフォルト値は “‘8’” ※“x”はチャネル番号	QSPI の QIO3 端子に割り付けるポート番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。
#define R_QSPI_SMSTR_CFG_QSPI_QIO3_BIT ※RX64M QSPI チャネル 0 のデフォルト値は “‘1’” ※“x”はチャネル番号	QSPI の QIO3 端子に割り付けるビット番号を設定してください。 設定値の前後にシングルコーテーション「‘」「’」をつけてください。

2.8　コードサイズ

表 2-3コードサイズに最新バージョンのモジュールを使用した場合のコードサイズを示します。

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

下表の値は下記条件で確認しています。

モジュールリビジョン: r_qspi_rx rev1.14

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
(統合開発環境のデフォルト設定に”-lang = c99”オプションを追加)
GCC for Renesas RX 4.08.04.201902
(統合開発環境のデフォルト設定に” -std=gnu99”オプションを追加)
IAR C/C++ Compiler for Renesas RX version 4.12.1
(統合開発環境のデフォルト設定)

コンフィグレーションオプション: デフォルト設定

表 2-3　コードサイズ

ROM、RAM およびスタックのコードサイズ（注１、注２、注３）				
デバイス	分類	使用メモリ		
		Renesas Compiler	GCC	IAR Compiler
RX65N	ROM	5,254 バイト	1,0584 バイト	7,862 バイト
	RAM	22 バイト	16 バイト	26 バイト
	最大使用ユーザスタック	160 バイト	-	196 バイト
	最大使用割り込みスタック	48 バイト	-	68 バイト
RX71M	ROM	5,254 バイト	10,584 バイト	7,862 バイト
	RAM	22 バイト	16 バイト	26 バイト
	最大使用ユーザスタック	160 バイト	-	196 バイト
	最大使用割り込みスタック	48 バイト	-	68 バイト

注１：動作条件は以下のとおりです。
- r_qspi_smstr_rx.c
- r_qspi_smstr_rx_target.c

注２：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。
注３：リトルエンディアン時の値です。エンディアンにより、上記のメモリサイズは、異なります。

2.9 引数

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

```
typedef volatile struct
{
   uint32_t  data_cnt;                /* Number of data (byte unit)       */
   uint8_t * p_tx_data;               /* Pointer to transmit data buffer  */
   uint8_t * p_rx_data;               /* Pointer to receive data buffer   */
   qspi_smstr_opmode_t   op_mode;     /* SPI operation mode               */
   qspi_smstr_tranmode_t tran_mode;   /* Data transfer mode               */
} qspi_smstr_info_t;
```

2.10 戻り値

API 関数の戻り値を示します。この列挙型は API 関数のプロトタイプ宣言とともに r_qspi_smstr_rx_if.h で記載されています。

```
typedef enum e_qspi_smstr_status
{
    QSPI_SMSTR_SUCCESS    = 0,          /* Successful operation              */
    QSPI_SMSTR_ERR_PARAM  = -1,         /* Parameter error                   */
    QSPI_SMSTR_ERR_HARD   = -2,         /* Hardware error                    */
    QSPI_SMSTR_ERR_OTHER  = -7          /* Other error                       */
} qspi_smstr_status_t;
```

2.11 コールバック関数

コールバック関数はありません。

2.12 FIT モジュールの追加方法

本モジュールは、使用するプロジェクトごとに追加する必要があります。ルネサスでは、スマート・コンフィグレータを使用した(1)、(3)、(5)の追加方法を推奨しています。ただし、スマート・コンフィグレータは、一部の RX デバイスのみサポートしています。サポートされていない RX デバイスについては(2)、(4)の方法を使用してください。

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

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

(3) CS+上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
CS+上で、スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: CS+編 (R20AN0470)」を参照してください。

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

(5) IAREW 上でスマート・コンフィグレータを使用して FIT モジュールを追加する場合
スタンドアロン版スマート・コンフィグレータを使用して、自動的にユーザプロジェクトに FIT モジュールを追加します。詳細は、アプリケーションノート「RX スマート・コンフィグレータ ユーザーガイド: IAREW 編 (R20AN0535)」を参照してください。

2.13 QSPI 以外の周辺機能とモジュール

QSPI FIT モジュールは QSPI 以外に以下の周辺機能を制御します。

- I/O ポート（以下、GPIO と略す）
- マルチファンクションピンコントローラ（以下、MPC と略す）
- DMA コントローラ（以下、DMAC と略す）
- データトランスファコントローラ（以下、DTC と略す）
- ロングキュー（以下、LONGQ と略す）－ソフトウェアモジュール

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

対象のソースコードは“r_qspi_smstr_rx64m_dev_port.c”に含まれます。「5.4 ターゲットMCU Devレイヤの関数詳細」を参考にしてください。

2.13.1 GPIO

GPIO FIT モジュールを使用していません。

QSPI FIT モジュール内の GPIO を制御する関数と制御対象レジスタを以下に示します。レジスタ名はユーザーズマニュアル ハードウェア編を参照してください。

表 2-4　制御関数と制御対象レジスタ

関数名	レジスタ
r_qspi_smstr_mpc_enable()	PMR
r_qspi_smstr_mpc_disable()	PMR
r_qspi_smstr_datao0_init()	ODR, DSCR, PODR, PDR
r_qspi_smstr_datao0_reset()	ODR, DSCR, PODR, PDR
r_qspi_smstr_datao1_init()	ODR, DSCR, PODR, PDR
r_qspi_smstr_datao1_reset()	ODR, DSCR, PODR, PDR
r_qspi_smstr_datao2_init()	ODR, DSCR, PODR, PDR
r_qspi_smstr_datao2_reset()	ODR, DSCR, PODR, PDR
r_qspi_smstr_datao3_init()	ODR, DSCR, PODR, PDR
r_qspi_smstr_datao3_reset()	ODR, DSCR, PODR, PDR
r_qspi_smstr_clk_init()	ODR, DSCR, PODR, PDR
r_qspi_smstr_clk_reset()	ODR, DSCR, PODR, PDR

2.13.2 MPC

MPC FIT モジュールを使用していません。

QSPI FIT モジュール内の MPC を制御する関数と制御対象レジスタを以下に示します。レジスタ名はユーザーズマニュアル ハードウェア編を参照してください。

制御関数と制御対象レジスタ

関数名	レジスタ
r_qspi_smstr_mpc_enable()	MPC
r_qspi_smstr_mpc_disable()	MPC

2.13.3　DMAC／DTC

DMAC 転送もしくは DTC 転送を使用する場合の制御方法を説明します。

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

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

表 2-5に示す制御関数を使って、各々の処理を実現してください。

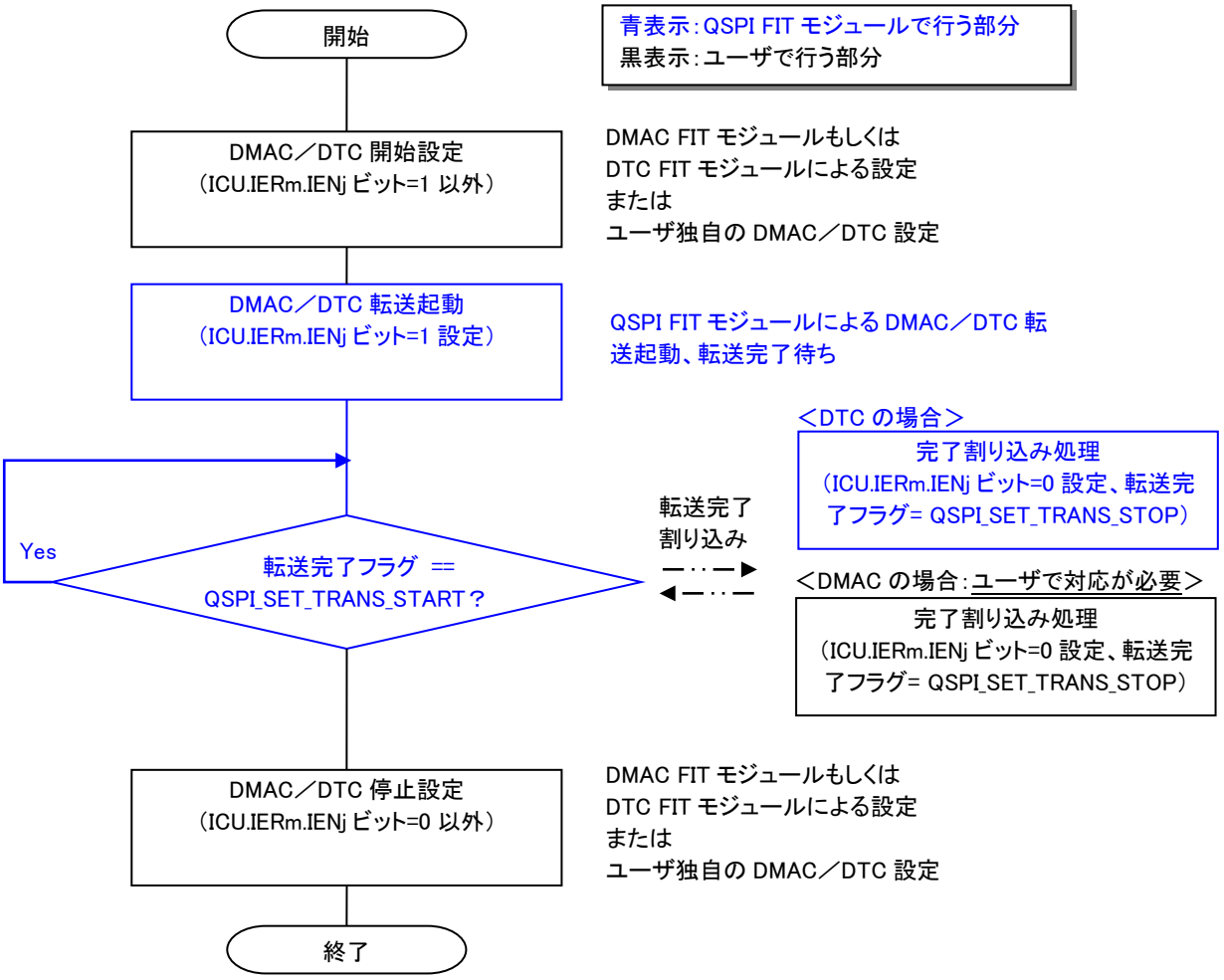


図 2.1　DMAC 転送および DTC 転送設定時の処理

DMAC／DTC 制御に関連する制御関数と処理内容について表 2-5に示します。

データ送信完了待ち処理 r_qspi_smstr_tx_dmacdtc_wait()、およびデータ受信完了待ち処理 r_qspi_smstr_rx_dmacdtc_wait()は、1 ミリ秒（ms）タイマを用いて完了待ちを行います。そのため、事前にユーザシステムにて CMT 等を用いて 1 ミリ秒（ms）タイマを起動してください。そして、コールバック関数等を用いて 1 ミリ秒（ms）毎に R_QSPI_SMstr_1ms_Interval()をコールしてください。

表 2-5　制御関数と処理内容

関数名	処理内容
r_qspi_smstr_spti_isrX()	QSPI channel “X” SPTI 割り込みハンドラ処理（X はチャネル番号）
r_qspi_smstr_spri_isrX()	QSPI channel “X” SPRI 割り込みハンドラ処理（X はチャネル番号）
r_qspi_smstr_tx_dmacdtc_wait()	DMAC／DTC 送信完了処理
r_qspi_smstr_rx_dmacdtc_wait()	DMAC／DTC 受信完了処理
r_qspi_smstr_int_spti_init()	SPTI 割り込み初期化処理
r_qspi_smstr_int_spri_init()	SPRI 割り込み初期化処理
r_qspi_smstr_int_spti_ier_set()	SPTI 割り込みの ICU.IERm.IENj ビットをセット
r_qspi_smstr_int_spri_ier_set ()	SPRI 割り込みの ICU.IERm.IENj ビットをセット
R_QSPI_Int_Spti_Ier_Clear()	SPTI 割り込みの ICU.IERm.IENj ビットをクリア
R_QSPI_Int_Spri_Ier_Clear()	SPRI 割り込みの ICU.IERm.IENj ビットをクリア
R_QSPI_SMstr_Int_Spti_Dmacdtc_flag_Set()	送信用の DMAC／DTC 転送完了フラグを設定
R_QSPI_SMstr_Int_Spri_Dmacdtc_flag_Set()	受信用の DMAC／DTC 転送完了フラグを設定
R_QSPI_SMstr_1ms_Interval()	各チャネルの内部タイマカウンタをインクリメント

2.13.4　CMT

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

他タイマやソフトウェアタイマで代用できます。

2.13.5　LONGQ

エラーログ取得機能で使用する FIT モジュールです。

QSPI FIT モジュールに LONGQ FIT モジュールを使用した制御例が含まれています。QSPI FIT モジュールのコンフィギュレーションオプションのデフォルトは、エラーログ取得機能無効設定です。「2.7 コンパイル時の設定」を参照してください。

(1)　R_LONGQ_Open()の設定

LONGQ FIT モジュールの R_LONGQ_Open()の第三引数 ignore_overflow を“1”に設定してください。これによりエラーログバッファは、リングバッファとして使用することが可能です。

(2)　制御手順

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

1. R_LONGQ_Open()
2. R_QSPI_SMstr_Set_LogHdlAddress()

2.14 FIT モジュール以外の環境で使用する場合の組み込み方法

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

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

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

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

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

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

2.15　for 文、while 文、do while 文について

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

以下に記述例を示します。

while 文の例

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

for 文の例

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

do while 文の例

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

3. API 関数

R_QSPI_SMstr_Open()

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

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Open(
    uint8_t channel,
    uint8_t spbr_data
)
```

Parameters

channel

QSPI チャネル番号

spbr_data

QSPI Bit Rate Register(SPBR) 設定値

Return Values

QSPI_SMSTR_SUCCESS	/* 正常終了した場合 */
QSPI_SMSTR_ERR_PARAM	/* パラメータ異常の場合 */
QSPI_SMSTR_ERR_OTHER	/* 他タスクが QSPI リソース取得済の場合 */

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

引数 channel で指定したチャネル番号の QSPI レジスタを初期化します。
引数 spbr_data で指定した値を QSPI Bit Rate Register(SPBR)に設定します。QSPI FIT モジュールでのビットレート分周設定ビット（SPCMD0.BRDV[1:0]）の設定値は“0”です。使用する MCU のユーザーズマニュアルハードウェア編を参考にして、動作環境に適した spbr_data を設定してください。
QSPCLK の極性 CPOL=1／位相 CPHA=1 に設定します。
正常終了時には、QSPI のモジュールストップ状態は解除され、QSPCLK 端子は汎用出力ポート H 出力状態、QIO0-3 端子は汎用入力ポート状態になります。
また、引数 channel で指定したチャネル番号の QSPI リソースを占有します。リソースを解放するためには R_QSPI_SMstr_Close()をコールしてください。
通信中は本関数をコールしないでください。通信中にコールした場合の通信は保証しません。

Example

```
qspi_smstr_status_t  ret = QSPI_SMSTR_SUCCESS;
uint8_t              channel;
uint8_t              spbr_data;

channel     = 0;
spbr_data   = 1;
ret = R_QSPI_SMstr_Open(channel, spbr_data);
```

Special Notes:

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

R_QSPI_SMstr_Close()

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

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Close(
    uint8_t channel
)
```

Parameters

channel

QSPI チャネル番号

Return Values

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

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

引数 channel で指定したチャネル番号の QSPI をモジュールストップ状態に設定します。
正常終了時には、QSPCLK 端子は汎用出力ポート H 出力状態、QIO0-3 端子は汎用入力ポート状態になります。
また、引数 channel で指定したチャネル番号の QSPI リソースを解放します。再度通信を行う場合、R_QSPI_SMstr_Open()をコールしてください。
通信中は本関数をコールしないでください。通信中にコールした場合の通信は保証しません。

Example

```
qspi_smstr_status_t  ret = QSPI_SMSTR_SUCCESS;
uint8_t              channel;

channel = 0;
ret = R_QSPI_SMstr_Close(channel);
```

Special Notes:

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

R_QSPI_SMstr_Control()

ビットレートや QSPCLK 位相／極性を変更する際に使用する関数です。

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Control(
    uint8_t channel,
    uint8_t clk_mode ,
    uint8_t spbr_data
)
```

Parameters

channel

QSPI チャネル番号

clk_mode

QSPCLK モード。以下を設定してください。

clk_mode=0 : CPOL=0、CPHA=0
clk_mode=1 : CPOL=0、CPHA=1
clk_mode=2 : CPOL=1、CPHA=0
clk_mode=3 : CPOL=1、CPHA=1

spbr_data

QSPI Bit Rate Register(SPBR) 設定値

Return Values

```
QSPI_SMSTR_SUCCESS          /* 正常終了した場合 */
QSPI_SMSTR_ERR_PARAM        /* パラメータ異常の場合 */
```

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

引数 channel で指定したチャネル番号の QSPI のビットレートや QSPCLK 位相／極性を変更します。
引数 spbr_data で指定した値を QSPI Bit Rate Register(SPBR)に設定します。QSPI FIT モジュールでのビットレート分周設定ビット（SPCMD0.BRDV[1:0]）の設定値は“0”です。使用する MCU のユーザーズマニュアルハードウェア編を参考にして、動作環境に適した spbr_data を設定してください。
通信中は本関数をコールしないでください。通信中にコールした場合の通信は保証しません。

Example

```
qspi_smstr_status_t  ret = QSPI_SMSTR_SUCCESS;
uint8_t              channel;
uint8_t              spbr_data;
uint8_t              clk_mode;

channel     = 0;
spbr_data   = 1;
ret = R_QSPI_SMstr_Open(channel, spbr_data);    /* Set CPOL=1 and CPHA=1. */

/* Change SPI clock mode and QSPI bit rate. */
clk_mode    = 1;     /* Set CPOL=0 and CPHA=1. */
spbr_data   = 3;
ret = R_QSPI_SMstr_Control(channel, clk_mode, spbr_data);
```

Special Notes:

なし

R_QSPI_SMstr_Write()

データを送信する際に使用する関数です。

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Write(
    uint8_t channel,
    qspi_smstr_info_t * p_qspi_smstr_info
)
```

Parameters

channel

QSPI チャネル番号

** p_qspi_smstr_info*

QSPI 情報構造体。

data_cnt

設定可能範囲は 1～4,294,967,295 です。0 を設定した場合、エラーを返します。また、DMAC 転送もしくは DTC 転送を指定する場合、設定値は 16 の倍数としてください。

*p_tx_data

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

*p_rx_data

未使用

op_mode

SPI モードを設定してください。

QSPI_SMSTR_SINGLE_SPI_WRITE : Single-SPI mode for writing
QSPI_SMSTR_DUAL_SPI : Dual-SPI mode
QSPI_SMSTR_QUAD_SPI : Quad-SPI mode

tran_mode

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

QSPI_SMSTR_SW : ソフトウェア転送.
QSPI_SMSTR_DMAC : DMAC 転送.
QSPI_SMSTR_DTC : DTC 転送

Return Values

QSPI_SMSTR_SUCCESS /* 正常終了した場合 */
QSPI_SMSTR_ERR_PARAM /* パラメータ異常の場合 */
QSPI_SMSTR_ERR_HARD /* ハードウェア異常の場合 */

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

引数 channel で指定したチャネル番号の QSPI を使って、データを送信します。
引数 tran_mode で DMAC 転送もしくは DTC 転送を指定した場合、転送可能なバイト数は 16 の倍数の値でです。
そうでない場合、エラー終了となり転送は行いません。

Example

```
#define DATA_CNT (uint32_t)(256)

uint8_t               buf[DATA_CNT];
qspi_smstr_info_t     tx_info;
qspi_smstr_status_t   ret = QSPI_SMSTR_SUCCESS;
uint8_t               channel;

channel = 0;
tx_info.data_cnt      = DATA_CNT;
tx_info.p_tx_data     = &buf[0];
tx_info.op_mode       = QSPI_SMSTR_SINGLE_SPI_WRITE;
tx_info.tran_mode     = QSPI_SMSTR_SW;
ret = R_QSPI_SMstr_Write(channel, &tx_info);
```

Special Notes:

DMAC 転送もしくは DTC 転送を指定する場合、以下の点にご注意ください。
・別途、DMAC FIT モジュール／DTC FIT モジュール／タイマモジュール（CMT FIT モジュール等）を入手してください。
・バッファのアドレスは 4 バイトの境界値としてください。
・転送データ数として 16 の倍数を指定しコールしてください。転送データ数として 1～15 の端数が発生する場合、別途ソフトウェア転送を指定しコールしてください。
・転送モードはブロック転送として、1 回の起動要因に対して、16 バイトの転送が行われるように設定してください。例えば、データ転送サイズを 4 バイトにした場合、ブロック転送回数を 4 回に設定してください。
・データ送信完了待ち処理 r_qspi_smstr_tx_dmacdtc_wait()はタイマを使用します。本関数をコールする前に CMT 等を用いて 1 ミリ秒（ms）タイマを起動してください。そして、1 ミリ秒（ms）毎に R_QSPI_SMstr_1ms_Interval()をコールしてください。
・本関数をコールする前に DMAC もしくは DTC を起動可能状態に設定してください。
・DMAC もしくは DTC を起動可能状態に設定していない場合、転送は行われません。戻り値は QSPI_SMSTR_ERR_HARD が返ります。
・リトルエンディアン設定、かつ DMAC 転送もしくは DTC 転送の設定で本関数をコールすると、送信データ格納バッファ内のデータ並びを変更し、データを送信します。送信完了後に元のデータ並びに戻します。

R_QSPI_SMstr_Read()

データを受信する際に使用する関数です。

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Read(
    uint8_t channel,
    qspi_smstr_info_t * p_qspi_smstr_info
)
```

Parameters

channel

QSPI チャネル番号

** p_qspi_smstr_info*

QSPI 情報構造体。

data_cnt

設定可能範囲は 1～4,294,967,295 です。0 を設定した場合、エラーを返します。また、DMAC 転送もしくは DTC 転送を指定する場合、設定値は 16 の倍数としてください。

*p_tx_data

未使用

*p_rx_data

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

op_mode

SPI モードを設定してください。

```
QSPI_SMSTR_SINGLE_SPI_READ      : Single-SPI mode for reading
QSPI_SMSTR_DUAL_SPI             : Dual-SPI mode
QSPI_SMSTR_QUAD_SPI             : Quad-SPI mode
```

tran_mode

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

```
QSPI_SMSTR_SW       : ソフトウェア転送
QSPI_SMSTR_DMAC     : DMAC 転送
QSPI_SMSTR_DTC      : DTC 転送
```

Return Values

```
QSPI_SMSTR_SUCCESS              /* 正常終了した場合 */
QSPI_SMSTR_ERR_PARAM            /* パラメータ異常の場合 */
QSPI_SMSTR_ERR_HARD             /* ハードウェア異常の場合 */
```

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description
引数 channel で指定したチャネル番号の QSPI を使って、データを受信します。
引数 tran_mode で DMAC 転送もしくは DTC 転送を指定した場合､転送可能なバイト数は 16 の倍数の値でです。
そうでない場合、エラー終了となり転送は行いません。

Example
```
#define DATA_CNT (uint32_t)(256)

uint8_t               buf[DATA_CNT];
qspi_smstr_info_t     rx_info;
qspi_smstr_status_t   ret = QSPI_SMSTR_SUCCESS;
uint8_t               channel;

channel = 0;
rx_info.data_cnt      = DATA_CNT;
rx_info.p_rx_data     = &buf[0];
rx_info.op_mode       = QSPI_SMSTR_SINGLE_SPI_READ;
rx_info.tran_mode     = QSPI_SMSTR_SW;
ret = R_QSPI_SMstr_Read(channel, &rx_info);
```

Special Notes:
DMAC 転送もしくは DTC 転送を指定する場合、以下の点にご注意ください。
・別途、DMAC FIT モジュール／DTC FIT モジュール／タイマモジュール（CMT FIT モジュール等）を入手してください。
・バッファのアドレスは 4 バイトの境界値としてください。
・転送データ数として 16 の倍数を指定しコールしてください。転送データ数として 1～15 の端数が発生する場合、別途ソフトウェア転送を指定しコールしてください。
・転送モードはブロック転送として、1 回の起動要因に対して、16 バイトの転送が行われるように設定してください。例えば、データ転送サイズを 4 バイトにした場合、ブロック転送回数を 4 回に設定してください。
・データ送信完了待ち処理 r_qspi_smstr_rx_dmacdtc_wait()はタイマを使用します。本関数をコールする前に CMT 等を用いて 1 ミリ秒（ms）タイマを起動してください。そして、1 ミリ秒（ms）毎に R_QSPI_SMstr_1ms_Interval()をコールしてください。
・本関数をコールする前に DMAC もしくは DTC を起動可能状態に設定してください。
・DMAC もしくは DTC を起動可能状態に設定していない場合、転送は行われません。戻り値は QSPI_SMSTR_ERR_HARD が返ります。

R_QSPI_SMstr_Get_BuffRegAddress()

QSPI データレジスタ（SPDR）のアドレスを取得する関数です。

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Get_BuffRegAddress (
    uint8_t channel,
    uint32_t * p_spdr_adr
)
```

Parameters

channel

QSPI チャネル番号

** p_spdr_adr*

SPDR のアドレス格納用ポインタ。格納先のアドレスを設定してください。

Return Values

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

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

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

Example

```
uint32_t             reg_buff;
qspi_smstr_status_t  ret = QSPI_SMSTR_SUCCESS;
uint8_t              channel;

channel = 0;
ret = R_QSPI_SMstr_Get_BuffRegAddress(channel, &reg_buff);
```

Special Notes:

なし

R_QSPI_SMstr_Int_Spti_Ier_Clear()

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

Format

```
void R_QSPI_SMstr_Int_Spti_Ier_Clear (
    uint8_t channel
)
```

Parameters

channel

QSPI チャネル番号

Return Values

なし

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

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

Example

```
DMA_Handler_W()
{
    R_QSPI_SMstr_Int_Spti_Ier_Clear(0);
    R_QSPI_SMstr_Int_Spti_Dmacdtc_Flag_Set(0, QSPI_SET_TRANS_STOP);
}
```

Special Notes:

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

R_QSPI_SMstr_Int_Spri_Ier_Clear()

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

Format

```
void R_QSPI_SMstr_Int_Spri_Ier_Clear (
    uint8_t channel
)
```

Parameters

channel

QSPI チャネル番号

Return Values

なし

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

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

Example

```
DMA_Handler_R()
{
    R_QSPI_SMstr_Int_Spri_Ier_Clear(0);
    R_QSPI_SMstr_Int_Spri_Dmacdtc_Flag_Set(0, QSPI_SET_TRANS_STOP);
}
```

Special Notes:

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

R_QSPI_SMstr_Int_Spti_Dmacdtc_Flag_Set()

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

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Int_Spti_Dmacdtc_Flag_Set (
    uint8_t channel,
    qspi_smstr_trans_flg_t flg
)
```

Parameters

channel

QSPI チャネル番号

flg

フラグ。以下を設定してください。

QSPI_SET_TRANS_STOP　：DMAC もしくは DTC 転送完了

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

Return Values

QSPI_SMSTR_SUCCESS　　*/* 正常終了した場合 */*

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

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

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

Example

```
DMA_Handler_W()
{
    R_QSPI_SMstr_Int_Spti_Ier_Clear(0);
    R_QSPI_SMstr_Int_Spti_Dmacdtc_Flag_Set(0, QSPI_SET_TRANS_STOP);
}
```

Special Notes:

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

R_QSPI_SMstr_Int_Spri_Dmacdtc_Flag_Set()

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

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Int_Spri_Dmacdtc_Flag_Set (
    uint8_t channel,
    qspi_smstr_trans_flg_t flg
)
```

Parameters

channel

QSPI チャネル番号

flg

フラグ。以下を設定してください。

QSPI_SET_TRANS_STOP　：DMAC もしくは DTC 転送完了

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

Return Values

QSPI_SMSTR_SUCCESS　　*/* 正常終了した場合 */*

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

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

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

Example

```
DMA_Handler_R()
{
    R_QSPI_SMstr_Int_Spri_Ier_Clear(0);
    R_QSPI_SMstr_Int_Spri_Dmacdtc_Flag_Set(0, QSPI_SET_TRANS_STOP);
}
```

Special Notes:

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

R_QSPI_SMstr_GetVersion()

ドライバのバージョン情報を取得する際に使用する関数です。

Format
uint32_t R_QSPI_SMstr_GetVersion(void)

Parameters
なし

Return Values
バージョン番号　　*上位 2 バイト：メジャーバージョン、下位 2 バイト：マイナーバージョン*

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

Description
バージョン情報を返します。

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

Special Notes:
なし

R_QSPI_SMstr_Set_LogHdlAddress()

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

Format

```
qspi_smstr_status_t R_QSPI_SMstr_Set_LogHdlAddress(
    uint32_t user_long_que
)
```

Parameters

user_long_que

LONGQ FIT モジュールのハンドラアドレスを設定してください。

Return Values

QSPI_SMSTR_SUCCESS　　*/* 正常終了した場合 */*

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

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

Example

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

qspi_smstr_status_t  ret = QSPI_SMSTR_SUCCESS;
uint32_t             MtlLogTbl[ERR_LOG_SIZE];   /*Err log buffer */
longq_err_t          err;
longq_hdl_t          p_qspi_user_long_que;   /* Pointer of LONGQ hndler */
uint32_t             long_que_hndl_address;  /* Address of LONGQ hndler */


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

long_que_hndl_address = (uint32_t)p_qspi_user_long_que;
ret = R_QSPI_SMstr_Set_LogHdlAddress(long_que_hndl_address);
```

Special Notes:

別途 LONGQ FIT モジュールを組み込んでください。また、r_qspi_smstr_rx_config.h の#define QSPI_SMSTR_CFG_LONGQ_ENABLE を有効にしてください。
QSPI_SMSTR_CFG_LONGQ_ENABLE が無効のときにこの関数が呼び出された場合、この関数はなにもしません。

R_QSPI_SMstr_Log()

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

Format

```
uint32_t R_QSPI_SMstr_Log(
      uint32_t flg,
      uint32_t fid,
      uint32_t line/
)
```

Parameters

flg

0x00000001（固定値）を設定してください。

fid

0x0000003f（固定値）を設定してください。

line

0x0001ffff（固定値）を設定してください。

Return Values

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

Properties

r_qspi_smstr_rx_if.h にプロトタイプ宣言されています。

Description

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

Example

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

uint8_t              buf[DATA_CNT];
qspi_smstr_info_t    tx_info;
qspi_smstr_status_t  ret = QSPI_SMSTR_SUCCESS;
uint8_t              channel;

channel = 0;
tx_info.data_cnt     = DATA_CNT;
tx_info.p_tx_data    = &buf[0];
tx_info.op_mode      = QSPI_SMSTR_QUAD_SPI;
tx_info.tran_mode    = QSPI_SMSTR_SW;

ret = R_QSPI_SMstr_Write(channel, &tx_info);
if (QSPI_SMSTR_SUCCESS != ret)
{
   /* Set last error log to buffer. */
   R_QSPI_SMstr_Log(
      USER_DRIVER_ID,
      USER_LOG_MAX,
      USER_LOG_ADR_MAX
   );
   R_QSPI_SMstr_Close(channel);
}
```

Special Notes:

別途 LONGQ FIT モジュールを組み込んでください。また、r_qspi_smstr_rx_config.h の#define QSPI_SMSTR_CFG_LONGQ_ENABLE を有効にしてください。
QSPI_SMSTR_CFG_LONGQ_ENABLE が無効のときにこの関数が呼び出された場合、この関数はなにもしません。

R_QSPI_SMstr_1ms_Interval()

関数が呼ばれる毎に内部タイマカウンタをインクリメントします。

Format
void R_QSPI_SMstr_1ms_Interval(void)

Parameters
なし

Return Values
なし

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

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

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

    channel = (uint32_t)pdata;
    if (channel == gs_cmt_channel)
    {
        R_QSPI_SMstr_1ms_Interval();
    }
}
```

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

4. 端子設定

QSPI FIT モジュールを使用するためには、マルチファンクションピンコントローラ（MPC）で周辺機能の入出力信号を端子に割り付ける（以下、端子設定と称す）必要があります。ただし、端子設定は、R_QSPI_SMstr_Write／R_QSPI_SMstr_Read 関数内で実行するため、ユーザ処理は不要です。

e² studio の場合は「FIT Configurator」または「Smart Configurator」の端子設定機能を使用することができます。FIT Configurator、Smart Configurator の端子設定機能を使用すると、端子設定画面で選択した端子を使用することができます。

選択した端子情報は r_qspi_smstr_rx_pin_config.h に反映され、「表 4-1 端子設定マクロ定義」に応じたマクロ定義に上書きされます。

表 4-1　端子設定マクロ定義

選択したオプション	マクロ定義
チャネル 0	R_QSPI_SMSTR_CFG_QSPI_QSPCLK_PORT R_QSPI_SMSTR_CFG_QSPI_QSPCLK_BIT R_QSPI_SMSTR_CFG_QSPI_QIO0_PORT R_QSPI_SMSTR_CFG_QSPI_QIO0_BIT R_QSPI_SMSTR_CFG_QSPI_QIO1_PORT R_QSPI_SMSTR_CFG_QSPI_QIO1_BIT R_QSPI_SMSTR_CFG_QSPI_QIO2_PORT R_QSPI_SMSTR_CFG_QSPI_QIO2_BIT R_QSPI_SMSTR_CFG_QSPI_QIO3_PORT R_QSPI_SMSTR_CFG_QSPI_QIO3_BIT

表 4-2に Power on Reset 後、および API 関数実行による端子の状態を示します。本モジュールは「(2) 制御可能なスレーブデバイス」に示すとおり、SPI モード 3（CPOL=1、CPHA=1）をサポートします。また、R_QSPI_SMstr_Close()後の QSPCLK 端子の状態は GPIO H 出力です。必要に応じて端子設定を見直してください。

表 4-2　関数実行後の端子の状態

関数名	QSPCLK 端子	QIO0-QIO3 端子（注 1）
（Power on Reset 後）	GPIO 入力状態	GPIO 入力状態
R_QSPI_SMstr_Open()後	GPIO H 出力状態	GPIO 入力状態
R_QSPI_SMstr_Write()中 R_QSPI_SMstr_Read()中 R_QSPI_SMstr_WriteRead()中	周辺機能 H/L 出力状態	周辺機能 H/L 出力状態
R_QSPI_SMstr_Write()後 R_QSPI_SMstr_Read()後 R_QSPI_SMstr_WriteRead()後	GPIO H 出力状態	GPIO 入力状態
R_QSPI_SMstr_Close()後	GPIO H 出力状態	GPIO 入力状態

注 1 ：QIO0-QIO3 端子は、外付け抵抗でプルアップ処理してください。「(1) ハードウェア構成例」を参照してください

5. デモプロジェクト

デモプロジェクトには、FIT モジュールとそのモジュールが依存するモジュール（例 : r_bsp）を使用する main()関数が含まれます。本 FIT モジュールには以下のデモプロジェクトが含まれます。

5.1 qspi_demo_rskrx64m, qspi_demo_rskrx65n_2m, qspi_demo_rskrx72n, qspi_demo_rskrx64m_gcc, qspi_demo_rskrx65n_2m_gcc, qspi_demo_rskrx72n_gcc

これらの QSPI FIT モジュールデモプログラムは、ルネサス RSKRX64M、RSKRX65N-2M、RSKRX72N デモボード用に設計されています。このプログラムは、マスタデバイスとして動作する RX MCU で QSPI FIT モジュールの API を使用して、スレーブデバイスとして動作するシリアルフラッシュ メモリ (MX25L3233F シリアル NOR フラッシュ) に読み書きする方法を示します。

コードをコンパイルした後、ターゲットボードにダウンロードし実行すると、初期化後に LED0 が点灯します。QSPI モジュールが正常にオープンされると、LED1 が点灯します。スレーブデバイスへのデータ書き込みが成功すると、LED2 が点灯します。スレーブデバイスからデータが正常に読み込まれると、LED3 が点灯します。QSPI モジュールが正常に閉じられると、すべての LED が消灯します。

セットアップと実行

1. r_qspi_smstr_rx_config.h でチャネル 0 のドライバサポートが有効になっていることを確認します。

```
#define QSPI_SMSTR_CFG_CH0_INCLUDED
```

2. データ転送モジュールの選択:

DMACA FIT モジュールを使用する場合、rskrx64m.h (RSKRX64M の場合)、rskrx65n_2m.h (RSKRX65N-2M の場合)、rskrx72n.h (RSKRX72N の場合) のマクロ定義「USE_DMAC_FIT」を以下の条件に変更することで有効にします。

```
#if 0
#define USE_DMAC_FIT
#endif
```

上記を以下のように変更 :

```
#if 1
#define USE_DMAC_FIT
#endif
```

DTC FIT モジュールを使用する場合、rskrx64m.h (RSKRX64M の場合)、rskrx65n_2m.h (RSKRX65N-2M の場合)、rskrx72n.h (RSKRX72N の場合) のマクロ定義「USE_DTC_FIT」は、条件を変更することで有効になります。

```
#if 0
#define USE_DTC_FIT
#endif
```

上記を以下のように変更 :

```
#if 1
#define USE_DTC_FIT
#endif
```

デフォルトでは、送信モードはソフトウェア転送モードになっています。

3. 端子設定：r_qspi_smstr_rx_pin_config.h で端子設定を確認

4. RSK ボードを PC に接続します (ルネサス E1 エミュレータを使用)。このサンプルアプリケーションをビルドし、ボードにダウンロードします。

5. ルネサス e2 studio IDE で、[Renesas Views] タブをクリックし、[デバッグ] 項目にマウスを移動して、[Renesas Debug Virtual Console] を選択して表示します。

6. ログと LED を確認して、アプリケーションがスレーブ デバイス (MX25L3233F シリアル NOR フラッシュ) にデータを読み書きすることを確認し、Renesas Debug Virtual Console ビューで送受信された 256 バイトデータを検証します。

サポートされるボード

RSKRX64M、RXKRX65N-2M、RSKRX72N

5.2　ワークスペースにデモを追加する

デモプロジェクトは、本アプリケーションノートで提供されるファイルの FITDemos サブディレクトリにあります。ワークスペースにデモプロジェクトを追加するには、「ファイル」 >> 「インポート」を選択し、「インポート」ダイアログから「一般」の「既存プロジェクトをワークスペースへ」を選択して「次へ」ボタンをクリックします。「インポート」ダイアログで「アーカイブ・ファイルの選択」ラジオボタンを選択し、「参照」ボタンをクリックして FITDemos サブディレクトリを開き、使用するデモの zip ファイルを選択して「終了」をクリックします。

5.3　デモのダウンロード方法

デモプロジェクトは、RX Driver Package には同梱されていません。デモプロジェクトを使用する場合は、個別に各 FIT モジュールをダウンロードする必要があります。「スマートブラウザ」の「アプリケーションノート」タブから、本アプリケーションノートを右クリックして「サンプル・コード（ダウンロード）」を選択することにより、ダウンロードできます。

6. 付録

6.1 動作確認環境

本 FIT モジュールの動作確認環境を以下に示します。

表 6-1 動作確認環境 (Rev.1.10)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e² studio V6.0.0
Cコンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.2.07.00
	コンパイルオプション : 統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.1.10
使用ボード	Renesas Starter Kit for RX64M (型名：R0K50564MSxxxBE) Renesas Starter Kit for RX65N (型名：RTK500565NSxxxxxBE) Renesas Starter Kit for RX65N-2MB (型名：RTK50565N2SxxxxxBE) Renesas Starter Kit for RX71M (型名：R0K50571MSxxxBE)

表 6-2 動作確認環境 (Rev.1.11)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e² studio V7.3.0
Cコンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00
	コンパイルオプション : 統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.1.11

表 6-3　動作確認環境 (Rev.1.12)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e² studio V7.3.0 IAR Embedded Workbench for Renesas RX 4.10.01
Cコンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201803 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.10.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.1.12
使用ボード	Renesas Starter Kit+ for RX65N（型名：RTK500565Nxxxxxx）

表 6-4　動作確認環境 (Rev.1.13)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e² studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
Cコンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.1.13
使用ボード	Renesas Starter Kit+ for RX72M (型名：RTK5572Mxxxxxxxxxx)

表 6-5　動作確認環境 (Rev.1.14)

項目	内容
統合開発環境	ルネサス エレクトロニクス製 e2 studio V7.4.0 IAR Embedded Workbench for Renesas RX 4.12.01
Cコンパイラ	ルネサス エレクトロニクス製 C/C++ compiler for RX family V.3.01.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 4.08.04.201902 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.12.01 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン／リトルエンディアン
モジュールのバージョン	Ver.1.14
使用ボード	Renesas Starter Kit+ for RX72N (型名：RTK5572Nxxxxxxxxxx)

表 6-6　動作確認環境 (Rev.1.20)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e2 studio 2022-07 IAR Embedded Workbench for Renesas RX 4.20.3
C コンパイラ	ルネサスエレクトロニクス製 C/C++ Compiler for RX Family V3.04.00 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -lang = c99
	GCC for Renesas RX 8.3.0.202202 コンパイルオプション：統合開発環境のデフォルト設定に以下のオプションを追加 -std=gnu99 リンクオプション：「Optimize size (サイズ最適化) (-Os)」を使用する場合、統合開発環境のデフォルト設定に以下のオプションを追加 -Wl,--no-gc-sections これは、FIT 周辺機器モジュール内で宣言されている割り込み関数をリンカが誤って破棄（discard）することを回避（work around）するための対策です。
	IAR C/C++ Compiler for Renesas RX version 4.20.3 コンパイルオプション：統合開発環境のデフォルト設定
エンディアン	ビッグエンディアン/リトルエンディアン
モジュールのリビジョン	Rev1.20
使用ボード	Renesas Starter Kit for RX64M (型名：R0K50564Mxxxxxx) Renesas Starter Kit+ for RX65N-2MB (型名：RTK50565N2Cxxxxxxx) Renesas Starter Kit+ for RX72N (型名：RTK5572NNDCxxxxxx)

6.2 トラブルシューティング

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

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

- CS+を使用している場合
  アプリケーションノート RX ファミリ CS+に組み込む方法 Firmware Integration Technology (R01AN1826)」
- e² studio を使用している場合
  アプリケーションノート RX ファミリ e² studio に組み込む方法 Firmware Integration Technology (R01AN1723)」

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

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

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

6.3　ターゲット MCU QSPI レイヤの関数詳細

以降で説明するレジスタ名の略語を以下に示します。以降の説明に記載する各レジスタの詳細およびビット名については、ユーザーズマニュアル ハードウェア編を参照してください。

以下に、QSPI FIT モジュールで使用するターゲット MCU QSPI レイヤの関数の詳細を示します。

表 6-7　レジスタ名の略語

略語	レジスタ名
SPCR	QSPI 制御レジスタ
SSLP	QSPI スレーブセレクト極性レジスタ
SPPCR	QSPI 端子制御レジスタ
SPSR	QSPI ステータスレジスタ
SPDR	QSPI データレジスタ
SPSCR	QSPI シーケンス制御レジスタ
SPSSR	QSPI シーケンスステータスレジスタ
SPBR	QSPI ビットレートレジスタ
SPDCR	QSPI データコントロールレジスタ
SPCKD	QSPI クロック遅延レジスタ
SSLND	QSPI スレーブセレクトネゲート遅延レジスタ
SPND	QSPI 次アクセス遅延レジスタ
SPCMD0 -SPCMD3	QSPI コマンドレジスタ 0-3
SPBFCR	QSPI バッファコントロールレジスタ
SPBDCR	QSPI バッファデータカウントセットレジスタ
SPBMUL0 - SPBMUL3	QSPI データ転送長倍数設定レジスタ 0-3

6.3.1　r_qspi_smstr_ch_check()

(1)　目的

指定したチャネルが定義されているかどうかを確認します。

(2)　機能

指定したチャネルが定義されている場合、正常終了します。

(3)　備考

r_qspi_smstr_rx_config.h の#define QSPI_SMSTR_CFG_CHx_INCLUDED（x=チャネル番号）を有効にすることで、使用するチャネルを指定してください。

6.3.2　r_qspi_smstr_enable()

(1)　目的

QSPI 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

1. モジュールストップ状態を解除

   r_qspi_smstr_module_enable()を参照してください。

2. 送信設定／受信設定／送受信設定の有効化手順の共通の処理を実行

   ・SPCR <- 08h : QSPI 機能無効、QSPI 機能の一部を初期化、マスタモード（QSPCLK 端子の出力許可）

   ・SSLP <- 00h：QSSL 信号 L アクティブ（※１）

   ・SPPCR <- 36h：Single/Dual 時 QIO2-3 は 1 固定、データ出力アイドル値は 1 固定

   ・SPBR ビットレート設定

   ・SPCKD <- 00h：SPCKD 遅延値設定（初期値）

   ・SSLND <- 00h：QSSL ネゲート遅延値（初期値）

   ・SPND <- 00h：次アクセス遅延値（初期値）

   ・SPDCR <- 00h：ダミーデータ送信禁止

   ・SPSR の PSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）

   ・SPSCR <- 00h：シーケンス長 SPCMD0 を使用

   ・SPBFCR <- 30h：送信バッファトリガ数 0 バイト、受信バッファトリガ数 1 バイト

   ・SPCMD0 <- E203h：

   CPHA=1、CPOL=1、ビットレート、Single-SPI、転送終了時 QSSL 信号をネゲート、32 ビット、MSB ファースト、次アクセス遅延有、QSSL ネゲート遅延有、クロック遅延有

   ・SPCMD1 <-E203h：

   CPHA=1、CPOL=1、ビットレート、Single-SPI、転送終了時 QSSL 信号をネゲート、32 ビット、MSB ファースト、次アクセス遅延有、QSSL ネゲート遅延有、クロック遅延有

   ・SPCMD2 <- E203h：

   CPHA=1、CPOL=1 ビットレート、Single-SPI、転送終了時 QSSL 信号をネゲート、32 ビット、MSB ファースト、次アクセス遅延有、QSSL ネゲート遅延有、クロック遅延有

(3)　備考

r_qspi_smstr_disable()と対となるものです。

6.3.3 r_qspi_smstr_disable()

(1) 目的

QSPI を無効化します。

(2) 機能

送信設定／受信設定／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。

1. モジュールストップ状態を解除
   r_qspi_smstr_module_enable()を参照してください。
2. SPSR の SPSSLF/SPTEF/SPRFF クリア
   r_qspi_smstr_spsr_clear()を参照してください。
3. モジュールストップ状態に設定
   r_qspi_smstr_module_disable()を参照してください。

(3) 備考

r_qspi_smstr_enable()と対となるものです。

6.3.4 r_qspi_smstr_change()

(1) 目的

ビットレートと QSPCLK 位相／極性を変更します。

(2) 機能

SPBR にビットレートを設定し、SPCMD に QSPCLK 位相／極性を設定します。

1. SPBR にビットレートを設定
2. SPCMD0-2 の CPHA（位相）と CPOL（極性）を設定

(3) 備考

なし

6.3.5 r_qspi_smstr_data_set_long()

(1) 目的

32 ビット送信データを設定します。

(2) 機能

SPDR にデータを書き込みます。

(3) 備考

なし

6.3.6 r_qspi_smstr_data_set_byte()

(1) 目的

8 ビット送信データを設定します。

(2) 機能

SPDR にデータを書き込みます。

(3) 備考

なし

6.3.7 r_qspi_smstr_data_get_long()

(1) 目的

32 ビット受信データを取得します。

(2) 機能

SPDR からデータを取出し、その値を戻り値に設定します。

(3) 備考

なし

6.3.8 r_qspi_smstr_data_get_byte()

(1) 目的

8 ビット受信データを取得します。

(2) 機能

SPDR からデータを取出し、その値を戻り値に設定します。

(3) 備考

なし

6.3.9 r_qspi_smstr_spsr_clear()

(1) 目的

SPSR の SPSSLF フラグ、SPTEF フラグ、SPRFF フラグをクリアします。

(2) 機能

SPSR の SPSSLF フラグ、SPTEF フラグ、SPRFF フラグをクリアします。

1. いずれかのフラグが 1 であれば、0 クリア

(3) 備考

なし

6.3.10 r_qspi_smstr_sptef_clear()

(1) 目的

SPSR の SPTEF フラグをクリアします。

(2) 機能

SPSR の SPTEF フラグをクリアします。

1. フラグが 1 であれば、0 クリア

(3) 備考

なし

6.3.11 r_qspi_smstr_sprff_clear()

(1) 目的

SPSR の SPRFF フラグをクリアします。

(2) 機能

SPSR の SPRFF フラグをクリアします。

1. フラグが 1 であれば、0 クリア

(3) 備考

なし

6.3.12 r_qspi_smstr_spsslf_clear()

(1) 目的

SPSR の SPSSLF フラグをクリアします。

(2) 機能

SPSR の SPSSLF フラグをクリアします。

1. フラグが 1 であれば、0 クリア

(3) 備考

なし

6.3.13 r_qspi_smstr_spsr_addr()

(1) 目的

SPSR のアドレスを取得します。

(2) 機能

SPSR のアドレスを戻り値に設定します。

(3) 備考

なし

6.3.14　r_qspi_smstr_trx_enable_single()

(1)　目的

Single-SPI モードでの送受信を許可します。

(2)　機能

端子を汎用入出力ポート機能から QSPI 機能に切り替えた後､Single-SPI モード QSPI 機能を有効にします。

本処理では、r_qspi_smstr_enable()の後の初期化手順として、Single-SPI モードの送受信設定専用の初期化処理を行います。

1. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
2. 送信バッファと受信バッファをリセット（r_qspi_smstr_buffer_reset()を参照）
3. 送信／受信のバッファデータ数トリガとデータ数を設定（r_qspi_smstr_datasize_set()を参照）
4. SPSCR の参照シーケンス番号、SPI 動作モード、QSSL 信号の制御方法を設定（※１）

＜データ数が QSPI の送信／受信バッファ個数の半分（※２）以上、かつ 16 の倍数でない時、＞

- SPSCR <- 01h：SPCMD0 -> SPCMD1
- SPCMD0.SPIMOD <- 00b：Single-SPI
- SPCMD0.SSLKP <- 1b：転送終了後から次アクセス開始まで QSSL 信号レベルを保持
- SPCMD1.SPIMOD < 00b：Single-SPI
- SPCMD1.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート

＜上記以外の時、＞

- SPSCR <- 00b：SPCMD0
- SPCMD0.SPIMOD <- 00b：Single-SPI
- SPCMD0.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート

5. 使用端子を QSPI 機能に設定

   r_qspi_smstr_mpc_enable()を参照してください。

6. SPCR の SPE を設定（機能有効設定）

   SPE をセットし、QSPI 機能を有効にします。

   - SPCR <- 48h：QSPI 機能有効、マスタモード（QSPCLK 端子の出力許可）

(3)　備考

r_qspi_smstr_trx_disable()と対となるものです。

※１：QSSL 制御機能を使用しません。QSSL 端子は他機能に割り当てることが可能であるため、QSSL 端子を汎用出力ポートに設定し、スレーブデバイスセレクト出力に割り当てることができます。

※２：RX64M、RX71M、RX65N、RX72M、RX72N、RX66N では送信／受信バッファはそれぞれ 8 ビット×32 個であり、その半分の 16 になります。

6.3.15　r_qspi_smstr_trx_disable()

(1)　目的

QSPI の送受信を禁止します。

(2)　機能

送受信を禁止します。端子を QSPI 機能から汎用入出力ポート機能に切り替えた後、送受信停止設定処理を行います。

1. 端子の周辺機能を無効化（r_qspi_smstr_trx_mpc_disable()を参照）
2. SPCR の SPE 設定（機能無効設定）

   SPE をクリアし、QSPI 機能を無効にします。

   ・SPCR <-08h : QSPI 機能無効、QSPI 機能の一部を初期化、マスタモード（QSPCLK 端子の出力許可）
3. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
4. SPSCR をリセット後の値に設定

   ・SPSCR <- 00b：SPCMD0

(3)　備考

r_qspi_smstr_trx_enable()と対となるものです。

6.3.16　r_qspi_smstr_tx_enable_dual()

(1)　目的

Dual-SPI モードでの送信を許可します。

(2)　機能

端子を汎用入出力ポート機能から QSPI 機能に切り替えた後､Dual-SPI モード QSPI 機能を有効にします。

本処理では、r_qspi_smstr_enable()の後の初期化手順として、Dual-SPI モードの送信設定専用の初期化処理を行います。

1. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
2. 送信バッファと受信バッファをリセット（r_qspi_smstr_buffer_reset()を参照）
3. 送信／受信のバッファデータ数トリガとデータ数を設定（r_qspi_smstr_datasize_set()を参照）
4. SPSCR の参照シーケンス番号、SPI リードライトアクセス設定、SPI 動作モード、QSSL 信号の制御方法を設定（※１）

   ＜データ数が QSPI の送信／受信バッファ個数の半分（※２）以上、かつ 16 の倍数でない時、＞
   - SPSCR <- 01h：SPCMD0 -> SPCMD1
   - SPCMD0.SPRW <- 0b：ライト動作
   - SPCMD0.SPIMOD <- 01b：Dual-SPI
   - SPCMD0.SSLKP <- 1b：転送終了後から次アクセス開始まで QSSL 信号レベルを保持
   - SPCMD1.SPRW <- 0b：ライト動作
   - SPCMD1.SPIMOD <- 01b：Dual-SPI
   - SPCMD1.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート

   ＜上記以外の時、＞
   - SPSCR <- 00b：SPCMD0
   - SPCMD0.SPRW <- 0b：ライト動作
   - SPCMD0.SPIMOD <- 01b：Dual-SPI
   - SPCMD0.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート
5. 使用端子を QSPI 機能に設定

   r_qspi_smstr_mpc_enable()を参照してください。
6. SPCR の SPE 設定（機能有効設定）

   SPE をセットし、QSPI 機能を有効にします。
   - SPCR <- 48h：QSPI 機能有効、マスタモード（QSPCLK 端子の出力許可）

(3)　備考

r_qspi_smstr_tx_disable()と対となるものです。

※１：QSSL 制御機能を使用しません。QSSL 端子は他機能に割り当てることが可能であるため、QSSL 端子を汎用出力ポートに設定し、スレーブデバイスセレクト出力に割り当てることができます。

※２：RX64M、RX71M、RX65N、RX72M、RX72N、RX66N では送信／受信バッファはそれぞれ 8 ビット×32 個であり、その半分の 16 になります。

6.3.17　r_qspi_smstr_tx_enable_quad()

(1)　目的

Quad-SPI モードでの送信を許可します。

(2)　機能

端子を汎用入出力ポート機能から QSPI 機能に切り替えた後､Quad-SPI モード QSPI 機能を有効にします。

本処理では、r_qspi_smstr_enable()の後の初期化手順として、Quad-SPI モードの送信設定専用の初期化処理を行います。

1. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
2. 送信バッファと受信バッファをリセット（r_qspi_smstr_buffer_reset()を参照）
3. 送信／受信のバッファデータ数トリガとデータ数を設定（r_qspi_smstr_datasize_set()を参照）
4. SPSCR の参照シーケンス番号、SPI リードライトアクセス設定、SPI 動作モード、QSSL 信号の制御方法を設定（※１）

   ＜データ数が QSPI の送信／受信バッファ個数の半分（※２）以上、かつ 16 の倍数でない時、＞
   - SPSCR <- 01h：SPCMD0 -> SPCMD1
   - SPCMD0.SPRW <- 0b：ライト動作
   - SPCMD0.SPIMOD <- 10b：Quad-SPI
   - SPCMD0.SSLKP <- 1b：転送終了後から次アクセス開始まで QSSL 信号レベルを保持
   - SPCMD1.SPRW <- 0b：ライト動作
   - SPCMD1.SPIMOD <- 10b：Quad-SPI
   - SPCMD1.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート

   ＜上記以外の時、＞
   - SPSCR <- 00b：SPCMD0
   - SPCMD0.SPRW <- 0b：ライト動作
   - SPCMD0.SPIMOD <- 10b：Quad-SP
   - SPCMD0.SSLKP <- 00b：転送終了後に QSSL 信号をネゲート
5. 使用端子を QSPI 機能に設定

   r_qspi_smstr_mpc_enable()を参照してください。
6. SPCR の SPE 設定（機能有効設定）

   SPE をセットし、QSPI 機能を有効にします。
   - SPCR <- 48h：QSPI 機能有効、マスタモード（QSPCLK 端子の出力許可）

(3)　備考

r_qspi_smstr_tx_disable()と対となるものです。

※１：QSSL 制御機能を使用しません。QSSL 端子は他機能に割り当てることが可能であるため、QSSL 端子を汎用出力ポートに設定し、スレーブデバイスセレクト出力に割り当てることができます。

※２：RX64M、RX71M、RX65N、RX72M、RX72N、RX66N では送信／受信バッファはそれぞれ 8 ビット×32 個であり、その半分の 16 になります。

6.3.18　r_qspi_smstr_tx_disable()

(1)　目的

QSPI の送受信を禁止します。

(2)　機能

送受信を禁止します。端子を QSPI 機能から汎用入出力ポート機能に切り替えた後、送受信停止設定処理を行います。

1. 端子の周辺機能を無効化（r_qspi_smstr_trx_mpc_disable()を参照）
2. SPCR の SPE 設定（機能無効設定）

   SPE をクリアし、QSPI 機能を無効にします。

   ・SPCR <-08h : QSPI 機能無効、QSPI 機能の一部を初期化、マスタモード（QSPCLK 端子の出力許可）
3. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
4. SPSCR をリセット後の値に設定

   ・SPSCR <- 00b：SPCMD0

(3)　備考

r_qspi_smstr_tx_enable_dual()もしくは r_qspi_smstr_tx_enable_quad()と対となるものです。

6.3.19　r_qspi_smstr_rx_enable_dual()

(1)　目的

Dual-SPI モードでの送信を許可します。

(2)　機能

端子を汎用入出力ポート機能から QSPI 機能に切り替えた後､Dual-SPI モード QSPI 機能を有効にします。

本処理では、r_qspi_smstr_enable()の後の初期化手順として、Dual-SPI モードの受信設定専用の初期化処理を行います。

1. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
2. 送信バッファと受信バッファをリセット（r_qspi_smstr_buffer_reset()を参照）
3. 送信／受信のバッファデータ数トリガとデータ数を設定（r_qspi_smstr_datasize_set()を参照）
4. SPSCR の参照シーケンス番号、SPI リードライトアクセス設定、SPI 動作モード、QSSL 信号の制御方法を設定（※１）

   ＜データ数が QSPI の送信／受信バッファ個数の半分（※２）以上、かつ 16 の倍数でない時、＞
   - SPSCR <- 02h：SPCMD0 -> SPCMD1 -> SPCMD2
   - SPCMD0.SPRW <- 1b：リード動作
   - SPCMD0.SPIMOD <- 01b：Dual-SPI
   - SPCMD0.SSLKP <- 1b：転送終了後から次アクセス開始まで QSSL 信号レベルを保持
   - SPCMD1.SPRW <- 1b：リード動作
   - SPCMD1.SPIMOD <- 01b：Dual-SPI
   - SPCMD1.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート
   - SPCMD2.SPRW <- 0b：ライト動作（転送完了のためのダミーのライトシーケンス設定）
   - SPCMD2.SPIMOD <- 10b：Quad-SPI

   ＜上記以外の時、＞
   - SPSCR <- 01b：SPCMD0 -> SPCMD1
   - SPCMD0.SPRW <- 1b：リード動作
   - SPCMD0.SPIMOD <- 01b：Dual-SPI
   - SPCMD0.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート
   - SPCMD1.SPRW <- 0b：ライト動作（転送完了のためのダミーのライトシーケンス設定）
   - SPCMD1.SPIMOD <- 10b：Quad-SPI
5. 使用端子を QSPI 機能に設定

   r_qspi_smstr_mpc_enable()を参照してください。
6. SPCR の SPE 設定（機能有効設定）

   SPE をセットし、QSPI 機能を有効にします。
   - SPCR <- 48h：QSPI 機能有効、マスタモード（QSPCLK 端子の出力許可）

(3)　備考

r_qspi_smstr_rx_disable()と対となるものです。

※１：QSSL 制御機能を使用しません。QSSL 端子は他機能に割り当てることが可能であるため、QSSL 端子を汎用出力ポートに設定し、スレーブデバイスセレクト出力に割り当てることができます。

※２：RX64M、RX71M、RX65N、RX72M、RX72N、RX66N では送信／受信バッファはそれぞれ 8 ビット×32 個であり、その半分の 16 になります。

6.3.20　r_qspi_smstr_rx_enable_quad()

(1)　目的

Quad-SPI モードでの送信を許可します。

(2)　機能

端子を汎用入出力ポート機能から QSPI 機能に切り替えた後､Quad-SPI モード QSPI 機能を有効にします。

本処理では、r_qspi_smstr_enable()の後の初期化手順として、Quad-SPI モードの受信設定専用の初期化処理を行います。

1. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
2. 送信バッファと受信バッファをリセット（r_qspi_smstr_buffer_reset()を参照）
3. 送信／受信のバッファデータ数トリガとデータ数を設定（r_qspi_smstr_datasize_set()を参照）
4. SPSCR の参照シーケンス番号、SPI リードライトアクセス設定、SPI 動作モード、QSSL 信号の制御方法を設定（※１）

＜データ数が QSPI の送信／受信バッファ個数の半分（※２）以上、かつ 16 の倍数でない時、＞
- ・SPSCR <- 02h：SPCMD0 -> SPCMD1 -> SPCMD2
- ・SPCMD0.SPRW <- 1b：リード動作
- ・SPCMD0.SPIMOD <- 10b：Quad-SPI
- ・SPCMD0.SSLKP <- 1b：転送終了後から次アクセス開始まで QSSL 信号レベルを保持
- ・SPCMD1.SPRW <- 1b：リード動作
- ・SPCMD1.SPIMOD <- 10b：Quad-SPI
- ・SPCMD1.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート
- ・SPCMD2.SPRW <- 0b：ライト動作（転送完了のためのダミーのライトシーケンス設定）
- ・SPCMD2.SPIMOD <- 10b：Quad-SPI

＜上記以外の時、＞
- ・SPSCR <- 01b：SPCMD0 -> SPCMD1
- ・SPCMD0.SPRW <- 1b：リード動作
- ・SPCMD0.SPIMOD <- 10b：Quad-SPI
- ・SPCMD0.SSLKP <- 0b：転送終了後に QSSL 信号をネゲート
- ・SPCMD1.SPRW <- 0b：ライト動作（転送完了のためのダミーのライトシーケンス設定）
- ・SPCMD2.SPIMOD <- 10b：Quad-SPI

5. 使用端子を QSPI 機能に設定

   r_qspi_smstr_mpc_enable()を参照してください。
6. SPCR の SPE 設定（機能有効設定）

   SPE をセットし、QSPI 機能を有効にします。
   - ・SPCR <- 48h：QSPI 機能有効、マスタモード（QSPCLK 端子の出力許可）

(3)　備考

r_qspi_smstr_rx_disable()と対となるものです。

※１：QSSL 制御機能を使用しません。QSSL 端子は他機能に割り当てることが可能であるため、QSSL 端子を汎用出力ポートに設定し、スレーブデバイスセレクト出力に割り当てることができます。

※２：RX64M、RX71M、RX65N、RX72M、RX72N、RX66N では送信／受信バッファはそれぞれ 8 ビット×32 個であり、その半分の 16 になります。

6.3.21　r_qspi_smstr_rx_disable()

(1)　目的

QSPI の送受信を禁止します。

(2)　機能

送受信を禁止します。端子を QSPI 機能から汎用入出力ポート機能へ切り替えた後、送受信停止設定処理を行います。

1. 端子の周辺機能を無効化（r_qspi_smstr_trx_mpc_disable()を参照）
2. SPCR の SPE 設定（機能無効設定）

   SPE をクリアし、QSPI 機能を無効にします。

   ・SPCR <-08h : QSPI 機能無効、QSPI 機能の一部を初期化、マスタモード（QSPCLK 端子の出力許可）
3. SPSR の SPSSLF/SPTEF/SPRFF クリア（r_qspi_smstr_spsr_clear()を参照）
4. SPSCR をリセット後の値に設定

   ・SPSCR <- 00b：SPCMD0

(3)　備考

r_qspi_smstr_rx_enable()_dual()もしくは r_qspi_smstr_rx_enable_quad()と対となるものです。

6.3.22　r_qspi_smstr_buffer_reset()

(1)　目的

送信バッファと受信バッファをリセットします。

(2)　機能

SPBFCR を使って、送信バッファと受信バッファをリセットし、その後、共に通常動作状態に設定します。

(3)　備考

なし

6.3.23　r_qspi_smstr_datasize_set()

(1)　目的

送信バッファデータ数トリガと受信バッファデータ数トリガと送信／受信のデータ数を設定します。

(2)　機能

SPBFCR を使って、送信バッファデータ数トリガと受信バッファデータ数トリガを設定します。その後、SPCMDn（n は、0-2）の SPB に転送データ長と SPBMULn（n は、0-2）に転送データ長倍数を設定することにより、送信／受信のデータ数を設定します。

1.　送信／受信バッファデータ数トリガ、転送データ長、転送データ長倍数を設定

＜データ数が QSPI の送信／受信バッファ個数の半分（※１）以上、かつ 16 の倍数でない時、＞

・SPBFCR <- 1Dh：送信バッファデータ数トリガ 16 バイト、受信バッファデータ数トリガ 16 バイト

・SPCMD0.SPB <- 0010b：シーケンス 0 転送データ長 32 ビット（4 バイト）

・SPBMUL0 シーケンス 0 転送データ長倍数設定：

＝（全データ数／QSPI のバッファ個数の半分）*（QSPI のバッファ個数の半分／転送データ長）

・SPCMD1.SPB <- 0000b：シーケンス 1 転送データ長 8 ビット（1 バイト）

・SPBMUL1 シーケンス 1 転送データ長倍数設定：

＝（全データ数／QSPI のバッファ個数の半分）の余り

＜データ数が QSPI の送信／受信バッファ個数の半分（※１）以上、かつ 16 の倍数の時、＞

・SPBFCR <- 1Dh：送信バッファデータ数トリガ 16 バイト、受信バッファデータ数トリガ 16 バイト

・SPCMD0.SPB <- 0010b：シーケンス 0 転送データ長 32 ビット（4 バイト）

・SPBMUL0 シーケンス 0 転送データ長倍数設定：

＝（全データ数／QSPI のバッファ個数の半分）*（QSPI のバッファ個数の半分／転送データ長）

＜データが QSPI の送信／受信バッファ個数の半分（※１）よりも少ない場合、＞

・SPBFCR <- 30h：送信バッファデータ数トリガ 0 バイト、受信バッファデータ数トリガ 1 バイト

・SPCMD0.SPB <- 0000b：シーケンス 0　転送データ長 8 ビット（1 バイト）

・SPBMUL0 シーケンス 0 転送データ長倍数設定：

＝全データ数

(3)　備考

※１：RX64M、RX71M、RX65N、RX72M、RX72N、RX66N では送信／受信バッファはそれぞれ 8 ビット×32 個であり、その半分の 16 になります。

6.4　ターゲット MCU Dev レイヤの関数詳細

以下に、QSPI FIT モジュールで使用するターゲット MCU Dev レイヤの関数の詳細を示します。

6.4.1　r_qspi_smstr_io_init()

(1)　目的

使用端子を汎用入出力ポートにした後、QIO0-3 端子をポート入力状態、QSPCLK 端子をポート出力状態にします。

(2)　機能

1. 使用端子を汎用入出力ポート機能に設定

   r_qspi_smstr_mpc_disable()を参照してください。

2. QSPCLK 端子をポート”H”出力に設定

   r_qspi_smstr_clk_init()を参照してください。

3. QIO0-3 端子をポート入力に設定

   r_qspi_smstr_dataio0_init()、r_qspi_smstr_dataio1_init()、r_qspi_smstr_dataio2_init()、r_qspi_smstr_dataio3_init()を参照してください。

(3)　備考

端子を周辺機能から汎用入出力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。

6.4.2　r_qspi_smstr_io_reset()

(1)　目的

使用端子を汎用入出力ポートにした後、入力端子と出力端子をポート入力状態にします。

(2)　機能

使用端子を汎用入出力ポートにした後、QIO0-3 端子と QSPCLK 端子をポート入力状態にします。

1. 使用端子を汎用入出力ポート機能に設定

   r_qspi_smstr_mpc_disable()を参照してください。

2. QSPCLK 端子をポート入力に設定

   r_qspi_smstr_clk_reset()を参照してください。

3. QIO0-3 端子をポート入力に設定

   r_qspi_smstr_dataio0_reset()、r_qspi_smstr_dataio1_reset()、r_qspi_smstr_dataio2_reset()、r_qspi_smstr_dataio3_reset()を参照してください。

(3)　備考

r_qspi_smstr_io_init()と対となるものです。

6.4.3 r_qspi_smstr_mpc_enable()

(1) 目的

使用端子を QSPI 機能に設定します。

(2) 機能

ユーザーズマニュアル ハードウェア編のマルチファンクションピンコントローラ(MPC)項の「端子入出力機能設定手順」に従い、以下の手順でレジスタの設定を行います。

1. ポートモードレジスタ(PMR)を”0”にし、端子を汎用入出力ポートに設定
   ・QIO0-3 端子、QSPCLK 端子 PMR <- 0b：汎用入出力ポートとして使用
2. 書き込みプロテクトレジスタ(PWPR)を設定し、Pxn 端子機能制御レジスタ(PxnPFS)を書き込み有効化
   ・PWPR.B0WI <- 0b：PFSWE ビットへの書き込み許可
   ・PWPR.PFSWE <- 1b：PFS レジスタへの書き込み許可
3. PxnPFS.PSEL[4:0]ビットにより QSPI 端子機能に設定
   ・QIO0-3 端子 PxnPFS <- 1Bh：QIO0-3 端子として使用可能な状態（※１）
   ・QSPCLK 端子 PxnPFS <- 1Bh：QSPCLK 端子として使用可能な状態（※１）
4. PWPR.PFSWE ビットを”0”にし、PxnPFS レジスタへの書き込みを禁止
   ・PWPR.PFSWE <- 0b：PFS レジスタへの書き込み禁止
   ・PWPR.B0WI <- 1b：PFSWE ビットへの書き込み禁止
5. 各端子の PMR 設定値を”1”にし、QSPI 端子機能に切り替え
   ・QIO0-3 端子、QSPCLK 端子 PMR <- 1b：QSPI 機能として使用

(3) 備考

Pxn 端子機能制御レジスタ(PxnPFS)を設定するときは、当該端子の PMR レジスタが”0”の状態で設定します。当該端子の PMR レジスタが”1”の状態で PxnPFS レジスタを設定すると、入力機能の場合は意図しないエッジが入力されたり、出力機能の場合は意図しないパルスが出力されたりする可能性があります。

※１：使用する MCU によっては、設定値が異なる場合があります。

6.4.4　r_qspi_smstr_mpc_disable()

(1)　目的

使用端子を汎用入出力ポート機能に設定します。

(2)　機能

ユーザーズマニュアル ハードウェア編のマルチファンクションピンコントローラ(MPC)項の「端子入出力機能設定手順」に従い、以下の手順でレジスタの設定を行います。

1. ポートモードレジスタ(PMR)を”0”にし、端子を汎用入出力ポートに設定
   ・QIO0-3 端子、QSPCLK 端子の PMR <- 0b：汎用入出力ポートとして使用
2. 書き込みプロテクトレジスタ(PWPR)を設定し、Pxn 端子機能制御レジスタ(PxnPFS)を書き込み有効化
   ・PWPR.B0WI <- 0b：PFSWE ビットへの書き込み許可
   ・PWPR.PFSWE <- 1b：PFS レジスタへの書き込み許可
3. PxnPFS.PSEL[4:0]ビットによりポート端子入出力機能に設定
   ・QIO0-3 端子、QSPCLK 端子 PxnPFS <- 00h：Hi-Z（リセット後の値）
4. PWPR.PFSWE ビットを”0”にし、PxnPFS レジスタへの書き込みを禁止
   ・PWPR.PFSWE <- 0b：PFS レジスタへの書き込み禁止
   ・PWPR.B0WI <- 1b：PFSWE ビットへの書き込み禁止

(3)　備考

Pxn 端子機能制御レジスタ(PxnPFS)を設定するときは、当該端子の PMR レジスタが”0”の状態で設定します。当該端子の PMR レジスタが”1”の状態で PxnPFS レジスタを設定すると、入力機能の場合は意図しないエッジが入力されたり、出力機能の場合は意図しないパルスが出力されたりする可能性があります。

6.4.5　r_qspi_smstr_dataio0_init()

6.4.6　r_qspi_smstr_dataio1_init()

6.4.7　r_qspi_smstr_dataio2_init()

6.4.8　r_qspi_smstr_dataio3_init()

(1)　目的

QIOn（n は、0-3 を示す。）端子をポート入力状態にします。

(2)　機能

1. オープンドレイン制御レジスタ(ODRn)を使って、QIOn 端子の出力形態を CMOS 出力に設定

・QIOn 端子 ODR <- 0b：CMOS 出力

2. プルアップ制御レジスタ(PCR)を使って、QIOn 端子の入力プルアップ抵抗を無効化

・QIOn 端子 PCR <- 0b：入力プルアップ無効

3. 駆動能力制御レジスタ(DSCR)を使って、QIOn 端子のポート駆動能力を設定

使用する MCU の AC タイミング特性の条件により、以下の設定を推奨します。（※１）（※２）

RX64M、RX71M、RX65N、RX72M、RX72N、RX66N の場合
QIOn 端子のポート駆動能力を「高駆動出力」に設定します。

・QIOn 端子 DSCR <- 1b：高駆動出力

4. ポート方向レジスタ(PDR)を使って、QIOn 端子をポート入力に設定

・QIOn 端子 PDR <- 0b：入力ポート

(3)　備考

※１：使用する MCU により、通常出力時と高駆動出力時の出力 Low レベル許容電流値（I_{OL}）と出力 Low レベル（V_{OL}）の特性が異なります。接続するデバイスに合わせて、適正な値を設定してください。

※２：駆動能力制御レジスタ(DSCR)は、使用できる端子が限定されています。

6.4.9　r_qspi_smstr_dataio0_reset()

6.4.10　r_qspi_smstr_dataio1_reset()

6.4.11　r_qspi_smstr_dataio2_reset()

6.4.12　r_qspi_smstr_dataio3_reset()

(1)　目的

QIOn（n は、0-3 を示す。）端子をポート入力状態にします。

(2)　機能

1. ポート方向レジスタ(PDR)を使って、QIOn 端子をポート入力に設定

・QIOn 端子 PDR <- 0b：入力ポート

2. 駆動能力制御レジスタ(DSCR)を使って、QIOn 端子のポート駆動能力を通常出力（リセット後の値）に設定

RX64M、RX71M、RX65N、RX72M、RX72N、RX66N の場合
QIOn 端子のポート駆動能力を「通常出力」に設定します。

・QIOn 端子 DSCR <- 0b：通常出力

3. プルアップ制御レジスタ(PCR)を使って、QIOn 端子の入力プルアップ抵抗を無効化

・QIOn 端子 PCR <- 0b：入力プルアップ無効

4. オープンドレイン制御レジスタ(ODRn)を使って、QIOn 端子の出力形態を CMOS 出力（リセット後の値）に設定

・QIOn 端子 ODR <- 0b：CMOS 出力

(3)　備考

なし

6.4.13　r_qspi_smstr_clk_init()

(1)　目的

QSPCLK 端子をポート”H”出力にします。

(2)　機能

1. オープンドレイン制御レジスタ(ODRn)を使って、QSPCLK 端子の出力形態を CMOS 出力に設定
   ・QSPCLK 端子 ODR <- 0b：CMOS 出力
2. 駆動能力制御レジスタ(DSCR)を使って、QSPCLK 端子のポート駆動能力を設定
   使用する MCU の AC タイミング特性の条件により、以下の設定を推奨します。（※１）（※２）
   RX64M、RX71M、RX65N、RX72M、RX72N、RX66N の場合
   QSPCLK 端子のポート駆動能力を「高駆動出力」に設定します。
   ・QSPCLK 端子 DSCR <- 1b：高駆動出力
3. ポート出力データレジスタ(PODR)を使って、QSPCLK 端子を H 出力に設定
   ・QSPCLK 端子 PODR <- 1b：H 出力
4. ポート方向レジスタ(PDR)を使って、QSPCLK 端子をポート出力に設定
   ・QSPCLK 端子 PDR <- 1b：出力ポート

(3)　備考

※１：使用する MCU により、通常出力時と高駆動出力時の出力 Low レベル許容電流値（I_{OL}）と出力 Low レベル（V_{OL}）の特性が異なります。接続するデバイスに合わせて、適正な値を設定してください。

※２：駆動能力制御レジスタ(DSCR)は、使用できる端子が限定されています。

6.4.14　r_qspi_smstr_clk_reset()

(1)　目的

QSPCLK 端子をポート入力状態にします。

(2)　機能

1. ポート方向レジスタ(PDR)を使って、QSPCLK 端子をポート入力に設定

   ・QSPCLK 端子 PDR <- 0b：入力ポート

2. ポート出力データレジスタ(PODR)を使って、QSPCLK 端子を L 出力（リセット後の値）に設定

   ・QSPCLK 端子 PODR <- 0b：L 出力

3. 駆動能力制御レジスタ(DSCR)を使って、QSPCLK 端子のポート駆動能力を通常出力（リセット後の値）に設定

   RX64M、RX71M、RX65N、RX72M、RX72N、RX66N の場合
   QSPCLK 端子のポート駆動能力を「通常出力」に設定します。

   ・QSPCLK 端子 DSCR <- 0b：通常出力

4. オープンドレイン制御レジスタ(ODRn)を使って、QSPCLK 端子の出力形態を CMOS 出力（リセット後の値）に設定

   ・QSPCLK 端子 ODR <- 0b：CMOS 出力

(3)　備考

なし

6.4.15 r_qspi_smstr_module_enable()

(1) 目的

モジュールストップ状態を解除します。

(2) 機能

1. プロテクトレジスタ(PRCR)と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

・PRCR <- A502h：モジュールストップコントロールレジスタのプロテクト解除

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

・PRCR <- A500h：モジュールストップコントロールレジスタのプロテクト許可

(3) 備考

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、書き込みともにできません。本関数の実行後に、QSPI レジスタにアクセスすることができます。なお、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。

6.4.16 r_qspi_smstr_module_disable()

(1) 目的

モジュールストップ状態にします。

(2) 機能

1. QSPI 関連のレジスタ設定を行うため、プロテクトレジスタ(PRCR)と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

・PRCR <- A502h：モジュールストップコントロールレジスタのプロテクト解除

・MSTPCRB.MSTPBxx <- 1b：モジュールストップ、QSPI レジスタへのリード／ライト不可能

・PRCR <- A500h：モジュールストップコントロールレジスタのプロテクト許可

(3) 備考

なし

6.4.17　r_qspi_smstr_tx_dmadtc_wait()

(1)　目的

DMAC／DTC 送信の完了待ち処理を行います。

(2)　機能

ソフトウェアループを使用して DMAC／DTC 送信の完了待ち処理を行います。ループ回数は 50,000 回です。1 回に転送するデータ数が大きい場合、正常に転送完了する前にループ回数が上限値を超える可能性があります。ループ回数を増やす等、必要に応じて処理を見直してください。

(3)　備考

QSPI FIT モジュールは、DMAC FIT モジュールもしくは DTC FIT モジュールを含みません。別途入手してください。

処理を見直す場合、エラー時の戻り値は QSPI_SMSTR_ERR_PARAM または QSPI_SMSTR_ERR_HARD を返すようにしてください。

6.4.18　r_qspi_smstr_rx_dmadtc_wait()

(1)　目的

DMAC／DTC 受信の完了待ち処理を行います。

(2)　機能

ソフトウェアループを使用して DMAC／DTC 送信の完了待ち処理を行います。ループ回数は 50,000 回です。1 回に転送するデータ数が大きい場合、正常に転送完了する前にループ回数が上限値を超える可能性があります。ループ回数を増やす等、必要に応じて処理を見直してください。

(3)　備考

QSPI FIT モジュールは、DMAC FIT モジュールもしくは DTC FIT モジュールを含みません。別途入手してください。

処理を見直す場合、エラー時の戻り値は QSPI_SMSTR_ERR_PARAM または QSPI_SMSTR_ERR_HARD を返すようにしてください。

6.4.19　r_qspi_smstr_int_spti_init()

(1)　目的

SPTI 割り込みを初期化します。

(2)　機能

1.　IENx 設定

・SPTI 割り込み要求許可ビット：禁止

2.　IR 設定

・SPTI 割り込みステータスフラグ：クリア

(3)　備考

なし

6.4.20　r_qspi_smstr_int_spri_init()

(1)　目的

SPRI 割り込みを初期化します。

(2)　機能

1.　IENx 設定

・SPRI 割り込み要求許可ビット：禁止

2.　IR 設定

・SPRI 割り込みステータスフラグ：クリア

(3)　備考

なし

6.4.21　r_qspi_smstr_int_spti_ier_set()

(1)　目的

SPTI 割り込みの ICU.IERm.IENj ビットをセットします。

(2)　機能

1.　IENx 設定

・SPTI 割り込み要求許可ビット：許可

(3)　備考

なし

6.4.22　r_qspi_smstr_int_spri_ier_set()

(1)　目的

SPRI 割り込み要求の ICU.IERm.IENj ビットをセットします。

(2)　機能

1.　IENx 設定

・SPRI 割り込み要求許可ビット：許可

(3)　備考

なし

6.5　駆動能力制御レジスタ（DSCR）設定の注意事項

使用する MCU により、通常出力時と高駆動出力時の出力 Low レベル許容電流値（I_{OL}）と出力 Low レベル（V_{OL}）の特性が異なります。接続するデバイスに合わせて、適正な値を設定してください。

6.6　各 MCU の QSPI 端子のポート機能

使用する MCU により、オープンドレイン制御レジスタ（ODR）と駆動能力制御レジスタ（DSCR）の制御方法が異なります。各 MCU の QSPI 端子のポート機能を表 6-8に示します。

表 6-8　各 MCU の QSPI 端子のポート機能

MCU	チャネル	端子	ポート番号	オープンドレイン制御レジスタ（ODR）	駆動能力制御レジスタ（DSCR）
RX64M RX65N RX71M	QSPI0	QSPCLK	P77	CMOS/オープンドレイン	高駆動固定
			PD5	CMOS/オープンドレイン	通常/高駆動
		QMO/QIO0	PC3	CMOS/オープンドレイン	通常/高駆動
			PD6	CMOS/オープンドレイン	通常/高駆動
		QMO/QIO1	PC4	CMOS/オープンドレイン	通常/高駆動
			PD7	CMOS/オープンドレイン	通常/高駆動
		QIO2	P80	CMOS/オープンドレイン	高駆動固定
			PD2	CMOS/オープンドレイン	通常/高駆動
		QIO3	P81	CMOS/オープンドレイン	高駆動固定
			PD3	CMOS/オープンドレイン	通常/高駆動
RX72M RX72N RX66N	QSPI0	QSPCLK	P77	CMOS/オープンドレイン	通常/高駆動
			PD5	CMOS/オープンドレイン	通常/高駆動
			PM0	CMOS/オープンドレイン	通常/高駆動
			PN4	CMOS/オープンドレイン	通常/高駆動
		QMO/QIO0	PC3	CMOS/オープンドレイン	通常/高駆動
			PD6	CMOS/オープンドレイン	通常/高駆動
			PJ3	CMOS/オープンドレイン	高駆動固定
			PM2	CMOS/オープンドレイン	通常/高駆動
		QMO/QIO1	PC4	CMOS/オープンドレイン	通常/高駆動
			PD7	CMOS/オープンドレイン	通常/高駆動
			PJ5	CMOS/オープンドレイン	高駆動固定
			PM3	CMOS/オープンドレイン	通常/高駆動
		QIO2	P00	CMOS/オープンドレイン	通常/高駆動
			P80	CMOS/オープンドレイン	通常/高駆動
			PD2	CMOS/オープンドレイン	通常/高駆動
			PM4	CMOS/オープンドレイン	通常/高駆動
		QIO3	P01	CMOS/オープンドレイン	通常/高駆動
			P81	CMOS/オープンドレイン	通常/高駆動
			PD3	CMOS/オープンドレイン	通常/高駆動
			PM5	CMOS/オープンドレイン	通常/高駆動

7. 参考ドキュメント

ユーザーズマニュアル：ハードウェア
（最新版をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース
（最新版をルネサス エレクトロニクスホームページから入手してください。）

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

テクニカルアップデート情報

本モジュールは以下のテクニカルアップデートの内容を反映しています。

- TN-RX*-A154A/J
  Rev.1.00 から対応済みです。

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

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.07.31	—	初版発行
1.06	2014.08.29	—	アプリケーションノートの Rev.をソースコードの Ver.に合わせた。
		—	src フォルダの r_qspi_smstr_private.h を修正した。
		—	src フォルダの r_qspi_smstr.c を修正した。
		—	demo フォルダに DTC 使用例のデモソースを追加した。
		7	1.4 関連アプリケーションノート：内容を追加した。
1.08	2014.12.26	—	「DMAC/DTC」を「DMAC もしくは DTC」に変更した。
		1	対象デバイス　に、RX71M を追加。
		1	FIT 関連ドキュメント　に、「CS+に組み込む方法(R01AN1826JU)」を追加。
		5	1.2.1 API の関数　表 1-2 API 関数　に、R_QSPI_SMstr_1ms_Interval()と注２を追加した。
		6	1.2.2 動作環境とメモリサイズ　(1)RX64M の場合　表 1-3 動作確認条件 使用ボード　型名　元は、Renesas Starter Kit for RX64M であった。
		6	1.2.2 動作環境とメモリサイズ　(1)RX64M の場合　表 1-4 必要メモリサイズ　最大使用割り込みスタック　元は、”-“であった。
		7	1.2.2 動作環境とメモリサイズ　に、(2)RX71M の場合　を追加。
		8	1.4 関連アプリケーションノート　に、以下を追加した。 -DMA コントローラ DMACA 制御モジュール Firmware Integration Technology（R01AN2063JJ） -DTC Module Using Firmware Integration Technology（R01AN1819EJ） -コンペアマッチタイマ (CMT)モジュール Firmware Integration Technology（R01AN1856JU） -EEPROM アクセス クロック同期式制御モジュール Firmware Integration Technology（R01AN2325JJ） - General Purpose Input/Output Driver Module Using Firmware Integration Technology（R01AN1721EU） - Multi-Function Pin Controller Module Using Firmware Integration Technology（R01AN1724EU）
		11	1.6.4 ソフトウェア構成 (b)r_qspi_smstr_target.c　元は、r_qspi_smstr_xxx.c であった。
		11	1.6.4 ソフトウェア構成 (c)r_qspi_smstr_target_dev_port.c　元は、r_qspi_smstr_xxx_dev_port.c であった。
		11	1.6.4 ソフトウェア構成 (d)スレーブデバイス用制御ソフトウェア　に「（R01AN2325JJ）」と「この Serial EEPROM 制御ソフトウェアには、FIT モジュールとの合わせ込みのためのドライバ I/F 関数（r_eeprom_spi_drvif_devX.c：X=0or1）があります。」を追加した。
		27	1.6.8 状態遷移図　「ポート初期化　QSPI 有効」　元は、「ポート初期化　QSPI 無効」であった。
		28	2.2 ソフトウェアの要求　に、r_cgc_rx を変更した。
		28	2.2 ソフトウェアの要求　に、r_dmaca_rx、r_dtc_rx、r_cmt_rx、r_gpio_rx、r_mpc_rx を追加した。
		28	2.4 ヘッダファイル　を修正した。

		29	2.6 コンパイル時の設定　Configuration options in r_qspi_smstr_rx_pin_config.h　元は、Configuration options in r_qspi_smstr_rx_config.h に記載されていた。
		29	2.6 コンパイル時の設定　QSPI_SMSTR_CFG_USE_FIT　元は、QSPI_SMSTR_CFG_USE_BSP だった。
		31	2.9.1 QSPI FIT モジュールの追加方法（手動で追加する場合） フォルダ名 r_qspi_smstr_rx　元は、r_qspi_smstr であった。
		31	2.9.1 QSPI FIT モジュールの追加方法（手動で追加する場合） r_qspi_smstr_rx_pin_config_reference.h を追加。
		34	2.10.3 DMAC／DTC　データ送信完了待ち、およびデータ受信完了待ち方法をタイマに変更。　元は、ソフトウェアループによる完了待ちであった。
		34	2.10.4 CMT　を追加。
		35	2.11 FIT モジュール以外の環境で使用する場合の組み込み QSPI_SMSTR_CFG_USE_FIT　元は、QSPI_SMSTR_CFG_USE_BSP だった。
		35	2.12 端子の状態　を追加。
		36	3.1 R_QSPI_SMstr_Open()　Description　QSPCLK 端子の状態　説明を追加。
		38	3.2 R_QSPI_SMstr_Close()　Description および Special Notes QSPCLK 端子の状態　説明を追加。
		42	3.4 R_QSPI_SMstr_Write()　Special Notes　タイマ、リトルエンディアン設定を追加。
		44	3.5 R_QSPI_SMstr_Read()　Special Notes　タイマの記述を追加。
		53	3.13 R_QSPI_SMstr_Log()　Description　に以下を追加。 -「LONGQ FIT モジュールを使用し、エラーログ取得終了処理を行います。」 -「エラーログサイズ設定方法については、LONGQ FIT モジュールを参照してください。」
		56	3.14 R_QSPI_SMstr_1ms_Interval()　を追加。
		61 63 64 66 67 69	4.1.14 r_qspi_smstr_trx_enable_single() 4.1.16 r_qspi_smstr_tx_enable_dual() 4.1.17 r_qspi_smstr_tx_enable_quad() 4.1.19 r_qspi_smstr_rx_enable_dual() 4.1.20 r_qspi_smstr_rx_enable_quad() 4.1.23 r_qspi_smstr_datasize_set() (3)備考　に、RX71M を追加。
		73	4.2.5 r_qspi_smstr_datao0_init() 4.2.6 r_qspi_smstr_datao1_init() 4.2.7 r_qspi_smstr_datao2_init() 4.2.8 r_qspi_smstr_datao3_init() (2)機能　に、RX71M を追加。
		74	4.2.9 r_qspi_smstr_datao0_reset() 4.2.10 r_qspi_smstr_datao1_reset() 4.2.11 r_qspi_smstr_datao2_reset() 4.2.12 r_qspi_smstr_datao3_reset() (2)機能　に、RX71M を追加。
		75	4.2.13 r_qspi_smstr_clk_init() (2)機能　に、RX71M を追加。
		76	4.2.14 r_qspi_smstr_clk_reset() (2)機能　に、RX71M を追加。
		81	4.4 各 MCU の QSPI 端子のポート機能　に、RX71M を追加。
1.09	2016.09.30	1	対象デバイス　に、RX65N を追加。

		8	1.2.2 動作環境とメモリサイズ　に、(3)RX65N の場合　を追加。
		30	2.6 コンパイル時の設定　表「Configuration options in r_qspi_smstr_rx_pin_config.h」　各#define 名を変更。
		30-31	2.6 コンパイル時の設定　表「Configuration options in r_qspi_smstr_rx_pin_config.h」　「RX64M QSPI」を追加。
		32	2.9 モジュールの追加方法　を更新した。
		5	4.端子設定　元は 2.12 端子状態であった。
		63-71	5.1.14　r_qspi_smstr_trx_enable_single() 5.1.16　r_qspi_smstr_tx_enable_dual() 5.1.17　r_qspi_smstr_tx_enable_quad() 5.1.19　r_qspi_smstr_rx_enable_dual() 5.1.20　r_qspi_smstr_rx_enable_quad() 5.1.23　r_qspi_smstr_datasize_set() RX65N を追加。
		75-78	5.2.5　r_qspi_smstr_dataio0_init() 5.2.6　r_qspi_smstr_dataio1_init() 5.2.7　r_qspi_smstr_dataio2_init() 5.2.8　r_qspi_smstr_dataio3_init() 5.2.9　r_qspi_smstr_dataio0_reset() 5.2.10　r_qspi_smstr_dataio1_reset() 5.2.11　r_qspi_smstr_dataio2_reset() 5.2.12　r_qspi_smstr_dataio3_reset() 5.2.13　r_qspi_smstr_clk_init() 5.2.14　r_qspi_smstr_clk_reset() RX65N を追加。
		83	5.4 各 MCU の QSPI 端子のポート機能 表 5-2　RX65N を追加。
1.10	2017.07.31	1	対象デバイスに、RX651 を追加
		-	下記の章を変更した。 ・1.1 QSPI FIT モジュールとは。 ・1.2 QSPI FIT モジュールの概要。 ・1.4 処理例 ・1.5 状態遷移図 ・2 API 情報 ・5.1 動作確認環境：元は 1.2.2 動作環境とメモリサイズであった。 ・6. 参考ドキュメント：元は 5.参考ドキュメントであった。 下記の章を追加した。 ・2.4 使用する割り込みベクタ。 ・2.8 コードサイズ。 ・2.11 コールバック関数。 ・2.12 FIT モジュールの追加方法。 ・5.2 トラブルシューティング
		26	2.2 ソフトウェアの要求　「r_cgc_rx」を削除
1.11	2019.02.01	57	表 5 2　動作確認環境 (Rev.1.11)を追加
		-	機能関連 Smart Configurator での GUI によるコンフィグオプション設定機能に対応 ■内容 GUI によるコンフィグオプション設定機能に対応するため、設定ファイルを追加。
1.12	2019.05.20	-	以下のコンパイラに対応

			・GCC for Renesas RX ・IAR C/C++ Compiler for Renesas RX
		1	「関連ドキュメント」に、R01AN1723 と R01AN1826 を削除
		1	「対象コンパイラ」を追加
		25	「2.2 ソフトウェアの要求」　依存する r_bsp モジュールのリビジョンを追加
		29	2.8「コードサイズ」を更新
		56	4.端子設定に端子設定マクロ定義を修正
		59	表 5-3 動作確認環境 (Ver.1.12)を追加
1.13	2019.07.30	1	対象デバイスに、RX72M を追加
		2	「関連ドキュメント」に、R01AN1833 を削除
		26	表 2-2 使用する割り込みベクター覧に RX72M を追加
		29	2.8「コードサイズ」を更新
		36	2.15「for 文、while 文、do while 文について」を追加
		37-56	3. API 関数に、各 API の Reentrant を削除
		60	表 5-4 動作確認環境 (Rev.1.13)を追加
		67-75	5.3.14　r_qspi_smstr_trx_enable_single() 5.3.16　r_qspi_smstr_tx_enable_dual() 5.3.17　r_qspi_smstr_tx_enable_quad() 5.3.19　r_qspi_smstr_rx_enable_dual() 5.3.20　r_qspi_smstr_rx_enable_quad() 5.3.23　r_qspi_smstr_datasize_set() (3)備考　に、RX72M を追加。
		79-82	5.4.5　r_qspi_smstr_dataio0_init() 5.4.6　r_qspi_smstr_dataio1_init() 5.4.7　r_qspi_smstr_dataio2_init() 5.4.8　r_qspi_smstr_dataio3_init() 5.4.9　r_qspi_smstr_dataio0_reset() 5.4.10　r_qspi_smstr_dataio1_reset() 5.4.11　r_qspi_smstr_dataio2_reset() 5.4.12　r_qspi_smstr_dataio3_reset() 5.4.13　r_qspi_smstr_clk_init() 5.4.14　r_qspi_smstr_clk_reset() RX72M を追加。
		87	表 5-6 各 MCU の QSPI 端子のポート機能に RX72M を追加
1.14	2019.11.22	1	対象デバイスに、RX72N と RX66N を追加
		27	表 2-2 使用する割り込みベクター覧に RX72N と RX66N を追加
		30	2.8「コードサイズ」を更新
		54-56	「3.12 R_QSPI_SMstr_Set_LogHdlAddress」と「3.13 R_QSPI_SMstr_Log」の「Special Notes」を修正
		62	表 5-5 動作確認環境 (Rev.1.14)を追加
		69-77	5.3.14　r_qspi_smstr_trx_enable_single() 5.3.16　r_qspi_smstr_tx_enable_dual() 5.3.17　r_qspi_smstr_tx_enable_quad() 5.3.19　r_qspi_smstr_rx_enable_dual() 5.3.20　r_qspi_smstr_rx_enable_quad() 5.3.23　r_qspi_smstr_datasize_set() (3)備考　に、RX72N と RX66N を追加。
		81-84	5.4.5　r_qspi_smstr_dataio0_init() 5.4.6　r_qspi_smstr_dataio1_init()

			5.4.7　r_qspi_smstr_dataio2_init() 5.4.8　r_qspi_smstr_dataio3_init() 5.4.9　r_qspi_smstr_dataio0_reset() 5.4.10　r_qspi_smstr_dataio1_reset() 5.4.11　r_qspi_smstr_dataio2_reset() 5.4.12　r_qspi_smstr_dataio3_reset() 5.4.13　r_qspi_smstr_clk_init() 5.4.14　r_qspi_smstr_clk_reset() RX72N と RX66N を追加。
		89	表 5-7 各 MCU の QSPI 端子のポート機能に RX72N と RX66N を追加
1.15	2021.09.30	6	表 1-3 使用端子と機能に QSPI FIT の端子についての制限事項を追加
1.20	2022.09.30	62	「5. デモプロジェクト」を追加。
		62-63	「5. デモプロジェクト」に RSK RX64M、RSK RX65N-2M、RSKRX72N を追加。
		66	「6.1 動作確認環境」： Rev.1.20 に対応する表を追加。
		プログラム	デモプロジェクトの追加

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。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、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 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。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

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

www.renesas.com/contact/

商標について

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