RX ファミリ

R01AN3436JJ0100
Rev.1.00
2018.10.31

Serial NAND Flash memory アクセス クロック同期式制御モジュール

Firmware Integration Technology

要旨

本アプリケーションノートでは、ルネサス エレクトロニクス社製 MCU を使用した Serial NAND Flash memory コマンド制御モジュールの使用方法について説明します。本モジュールは、Firmware Integration Technology（以下、FIT と略す）に準拠しています。以降、本モジュールを NAND SPI FIT モジュールと称します。

なお、本モジュールは Serial NAND Flash memory のコマンドを制御するためのミドルウェアです。ブロック管理制御は含まれません。また、MCU のシリアル通信機能（クロック同期式シングルマスタ）を制御するデバイスドライバは含まないため、別途、以下の URL から入手してください。

- Serial NAND Flash memory コマンド制御、クロック同期式シングルマスタ制御
  https://www.renesas.com/driver/spi_serial_flash

対象デバイス

東芝メモリ社製 Serial Interface NAND

動作確認に使用した MCU

ルネサス エレクトロニクス社製 RX ファミリ MCU

RX100 シリーズ、RX200 シリーズ、RX600 シリーズ、RX700 シリーズ

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。

FIT 関連ドキュメント

- Firmware Integration Technology ユーザーズマニュアル (R01AN1833JU)
- ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685JU)
- e² studio に組み込む方法 Firmware Integration Technology (R01AN1723JU)
- CS+に組み込む方法 Firmware Integration Technology (R01AN1826JJ)
- RX ファミリ RSPI クロック同期式シングルマスタ制御モジュール Firmware Integration Technology（R01AN1914JJ）
- 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（R01AN1940JJ）
- RX ファミリ DMA コントローラ DMACA 制御モジュール Firmware Integration Technology（R01AN2063JJ）
- RX Family DTC モジュール Firmware Integration Technology（R01AN1819JJ）
- RX ファミリ コンペアマッチタイマ（CMT）モジュール Firmware Integration Technology（R01AN1856JJ）
- RX ファミリ GPIO モジュール Firmware Integration Technology（R01AN1721JJ）
- RX ファミリ MPC モジュール Firmware Integration Technology（R01AN1724JJ）
- RX ファミリ ロングワード型キューバッファ（LONGQ）モジュール Firmware Integration Technology（R01AN1889JJ）

目次

1. 概要

1.1 NAND SPI FIT モジュールとは

本モジュールは API として、プロジェクトに組み込んで使用します。本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。

表 1-1 に本アプリケーションノート内で使用する用語／略語を示します。

表 1-1　用語／略語定義

用語／略語	内容
FIT	Firmware Integration Technology の略語
“機能名”FIT モジュール	FIT を使用した他の機能制御モジュールの略語
NAND SPI FIT モジュール	Serial NAND Flash memory アクセス クロック同期式制御モジュール Firmware Integration Technology の略語
RSPI	RX ファミリ MCU シリアルペリフェラルインタフェースの略語
QSPI	RX ファミリ MCU クワッドシリアルペリフェラルインタフェースの略語
Single-SPI	Single SPI mode による通信
Dual-SPI	Dual SPI mode による通信
Quad-SPI	Quad SPI mode による通信

1.2　NAND SPI FIT モジュールの概要

NAND SPI FIT モジュールは、ルネサス エレクトロニクス社製 MCU を使用し、Serial NAND Flash memory を制御します。表 1-2 に概要を示します。

表 1-2　概要

項目		内容
マスタデバイス		ルネサス エレクトロニクス社製 MCU
スレーブデバイス		Serial NAND Flash memory
制御対象		Serial NAND Flash memory のコマンド制御 （注：ブロック管理制御を含みません）
通信方式		シリアル通信（クロック同期式）
通信クロックの極性と位相		SPI モード 3（CPOL=1、CPHA=1）
制御デバイス数		最大 2 個
エンディアン		リトルエンディアン／ビッグエンディアンに対応
Firmware Integration Technology（FIT）		準拠
組み合せて使用する FIT モジュール	基本設定	・BSP FIT モジュール
	シリアル通信（クロック同期式シングルマスタ制御）	・RSPI SMstr FIT モジュール（Single-SPI）（注２） ・QSPI SMstr FIT モジュール（Single/Dual/Quad-SPI）（注３）
	I/O 設定	・GPIO FIT モジュール
	端子設定	・MPC FIT モジュール
	データ転送（注１）	・DMACA FIT モジュール ・DTC FIT モジュール
	タイマ（注１）	・CMT FIT モジュール
	データ管理	・LONGQ FIT モジュール

注１：DMAC 転送もしくは DTC 転送を使用する場合のみ

注２：RX ファミリ RSPI クロック同期式シングルマスタ制御モジュール Firmware Integration Technology（R01AN1914JJ）

注３：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（R01AN1940JJ）

1.3　API の概要

表 1-3 に NAND SPI FIT モジュールに含まれる API 関数を示します。

表 1-3　API 関数

関数名	説明
R_NAND_SPI_Open()	ドライバのオープン処理
R_NAND_SPI_Close()	ドライバのクローズ処理
R_NAND_SPI_SendCmd()	コマンド発行処理
R_NAND_SPI_DataIn()	データ入力処理
R_NAND_SPI_DataOut()	データ出力処理
R_NAND_SPI_SetCS()	チップセレクト制御処理
R_NAND_SPI_GetVersion()	ドライバのバージョン情報取得処理
R_NAND_SPI_1msInterval()	インターバルタイマカウント処理
R_NAND_SPI_SetLogHdlAddress()	LONGQ FIT モジュールのハンドラアドレス設定処理
R_NAND_SPI_Log()	LONGQ FIT モジュールを使ったエラーログ取得処理

1.4　処理例

1.4.1　ソフトウェアとハードウェアの全体構成

Serial NAND Flash memory に MCU のシリアル通信端子とチップセレクト制御端子を接続し、通信を実現します。

図 1-1　システム構成

表 1-4　周辺機能とソフトウェア制御

周辺機能	ソフトウェア制御
シリアル通信	クロック同期式シングルマスタを使用し、クロック出力とデータ入出力を制御。
チップセレクト制御	汎用ポートを使用し、信号出力を制御。

1.4.2　ソフトウェア構成

図 1-2 にソフトウェア構成を示します。また、表 1-5 に各レイヤの概要を示します。

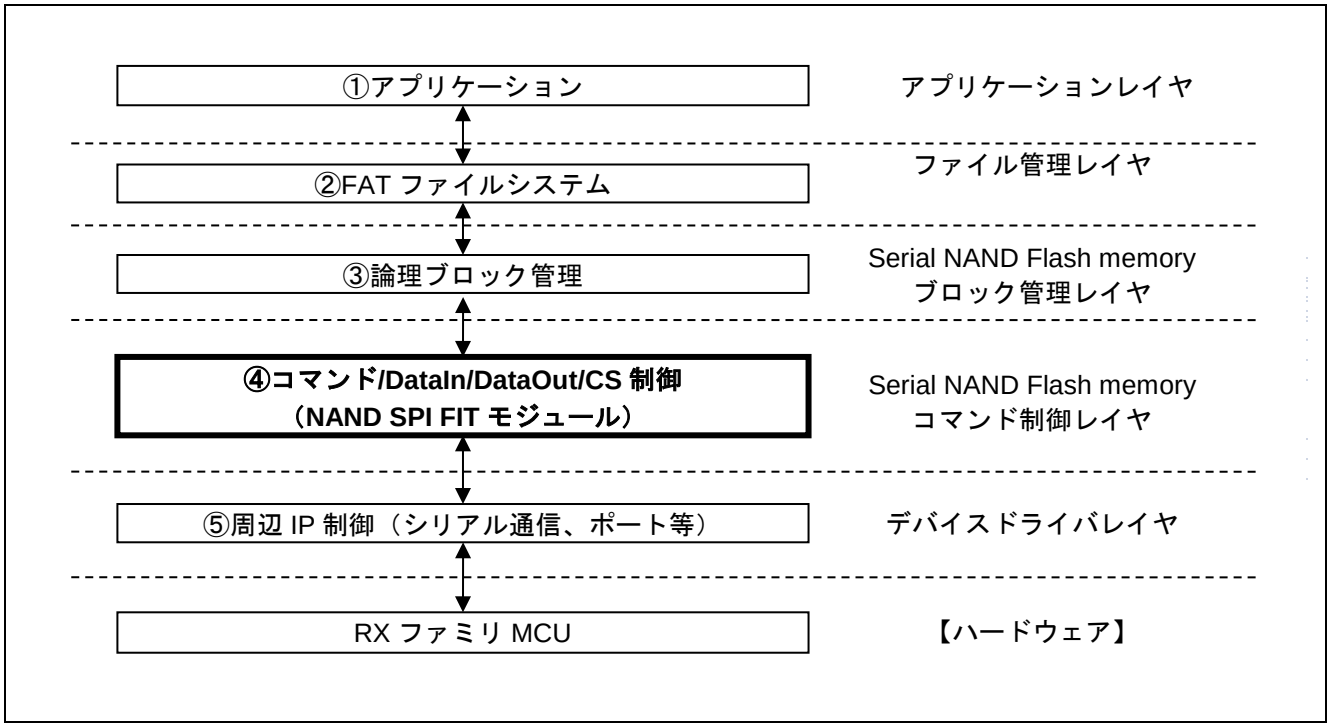


図 1-2　ソフトウェア構成

表 1-5　ソフトウェアブロック概要

ブロック名称	概要
①アプリケーション	ユーザアプリケーション
②FAT ファイルシステム	FAT ファイルシステム
③論理ブロック管理	Serial NAND Flash memory の論理ブロック管理
④コマンド /DataIn/DataOut/CS 制御	**NAND SPI FIT モジュール** **Serial NAND Flash memory にアクセスするためのコマンド制御部。**
⑤周辺 IP 制御（シリアル通信、ポート等）	以下の制御を実行するデバイスドライバレイヤ。各 FIT モジュールを用いて制御が可能。 ・シリアル通信（クロック同期式シングルマスタ制御） ・I/O 制御 ・端子設定 ・データ転送 ・タイマ

1.4.3　ハードウェア構成

以下に Single-SPI、Dual-SPI、Quad-SPI の接続図を示します。

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。

(1)　Single-SPI 構成例

以下に Single-SPI 使用時の接続例を示します。

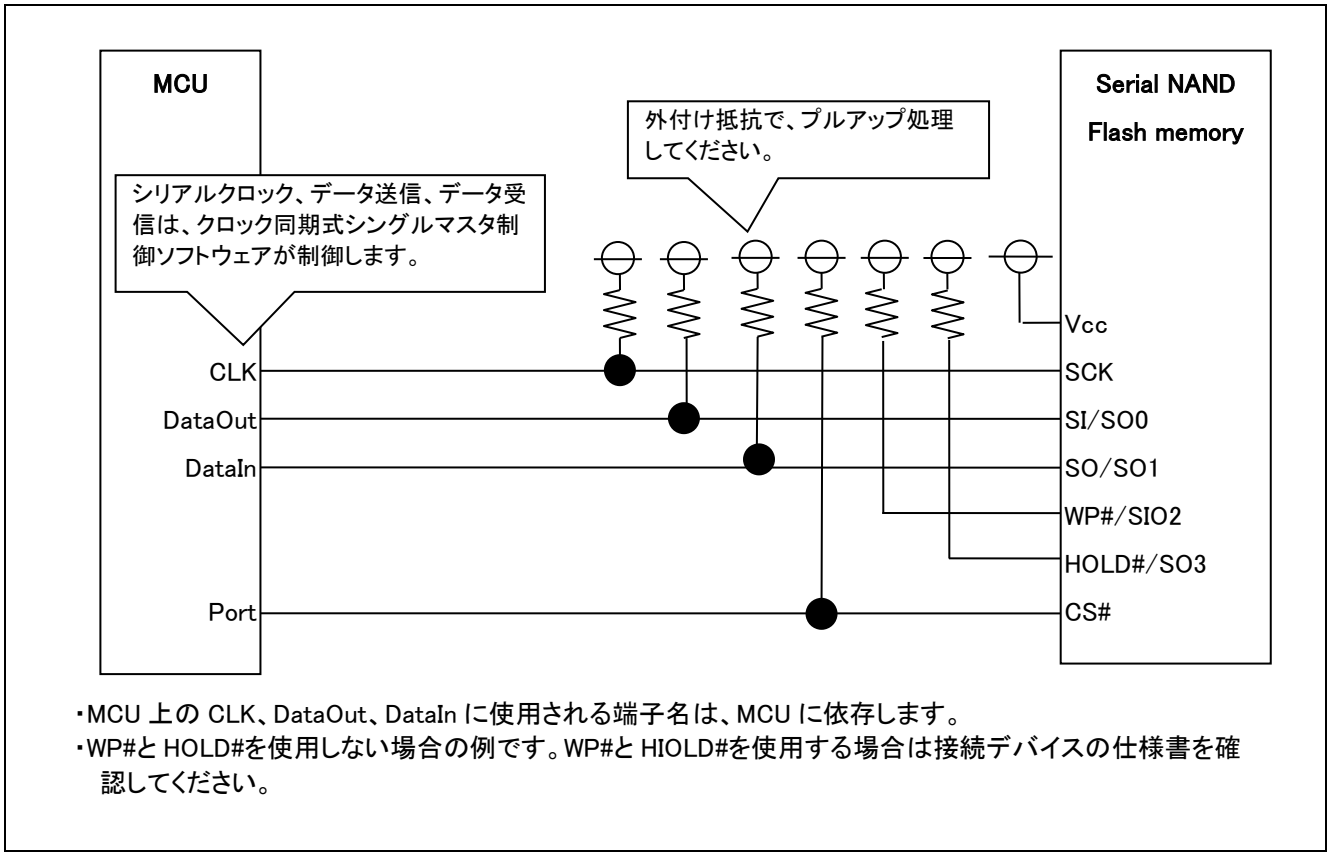


図 1-3　Single-SPI 使用時の接続例

表 1-6　Single-SPI 使用端子と機能

MCU 端子名	入出力	内容
CLK	出力	クロック出力
DataOut	出力	マスタデータ出力
DataIn	入力	マスタデータ入力
Port	出力	チップセレクト端子出力

(2)　Dual-SPI 構成例

以下に Dual-SPI 使用時の接続例を示します。

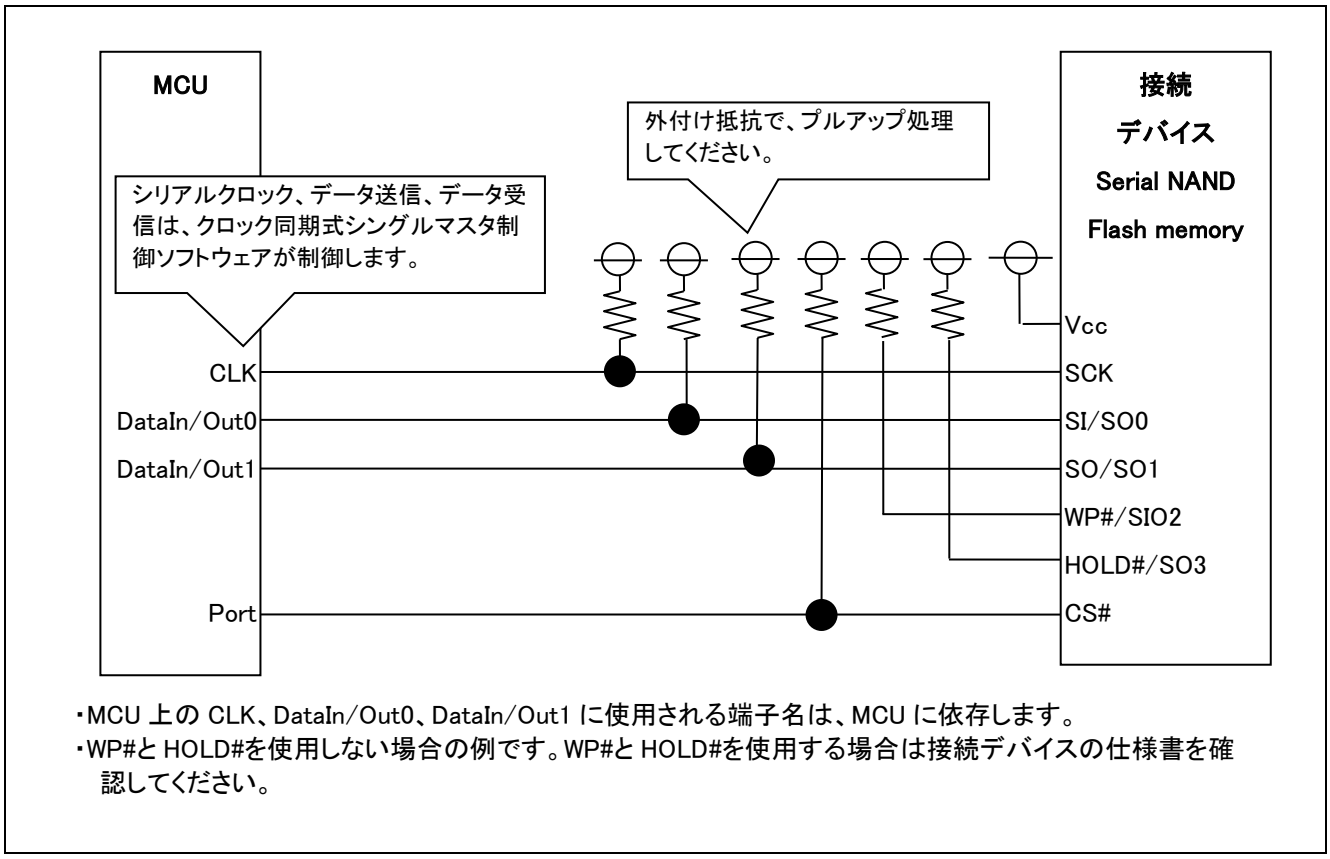


図 1-4　Dual-SPI 使用時の MCU と SPI スレーブデバイスの接続例

表 1-7　Dual-SPI 使用端子と機能

MCU 端子名	入出力	内容
CLK	出力	クロック出力
DataIn/Out0	入出力	マスタデータ入出力 0
DataIn/Out1	入出力	マスタデータ入出力 1
Port	出力	チップセレクト端子出力

(3)　Quad-SPI 構成例

以下に Quad-SPI 使用時の接続例を示します。

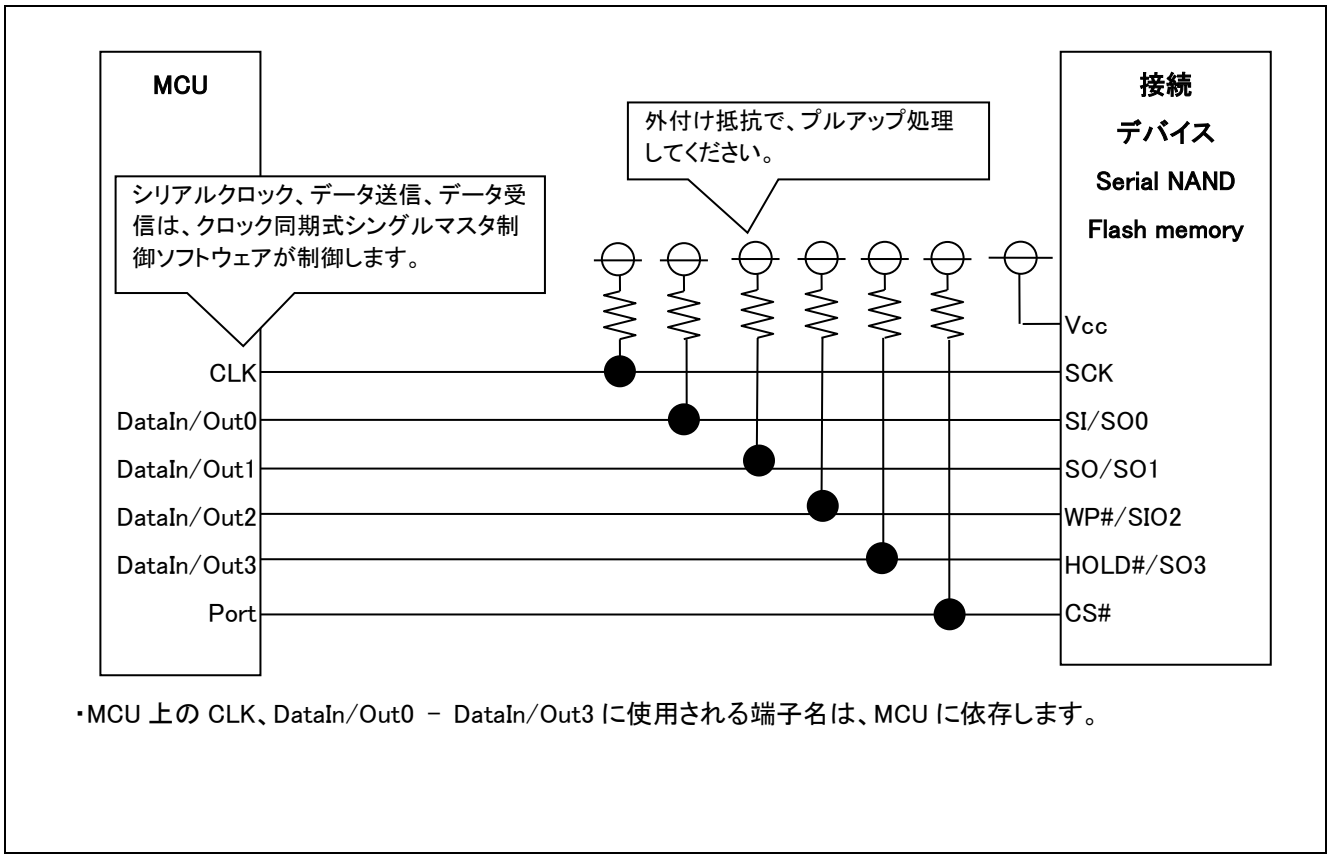


図 1-5　Quad-SPI 使用時の MCU と SPI スレーブデバイスの接続例

表 1-8　Quad-SPI 使用端子と機能

MCU 端子名	入出力	内容
CLK	出力	クロック出力
DataIn/Out0	入出力	マスタデータ入出力０
DataIn/Out1	入出力	マスタデータ入出力１
DataIn/Out2	入出力	マスタデータ入出力２
DataIn/Out3	入出力	マスタデータ入出力３
Port	出力	チップセレクト端子出力

1.4.4　ソフトウェア説明

(1)　API コール手順

NAND SPI FIT モジュールの API は、主に「コマンド発行／データ入力／データ出力／チップセレクト制御」で構成されています。発行するコマンドタイプにより、API を組み合わせて制御を実現してください。

(a)　コマンド発行のみの場合

図 1-6 に「コマンド発行のみ」の API コール例を示します。

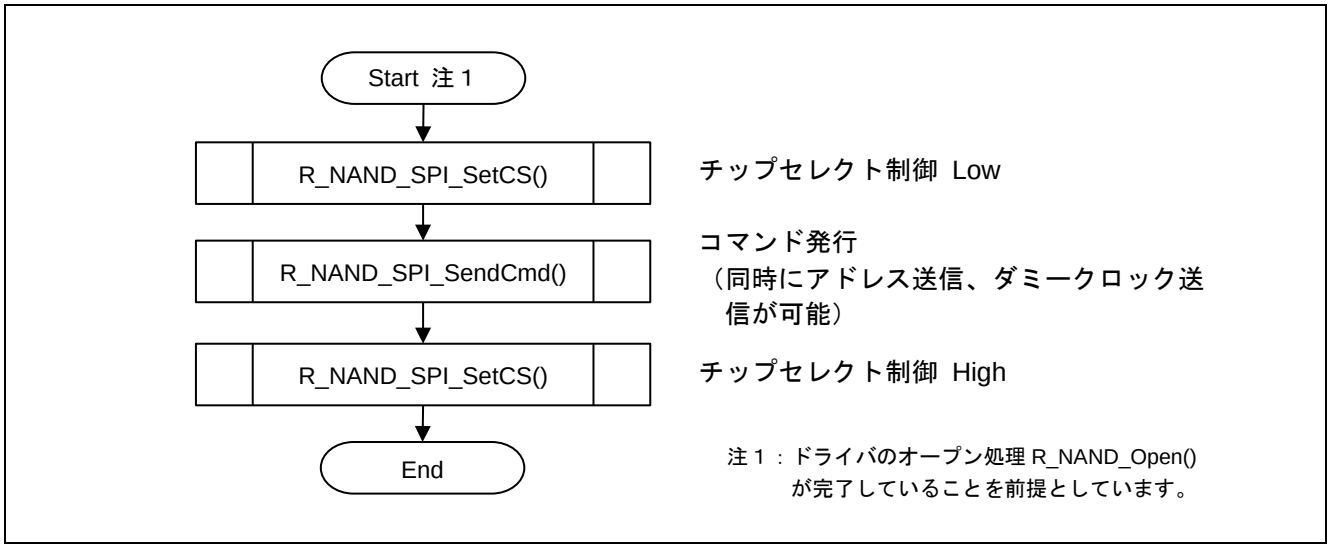


図 1-6　API コール例 「コマンド発行のみ」

(b)　コマンド発行＋データ入力の場合

図 1-7 に「コマンド発行＋データ入力」の API コール例を示します。

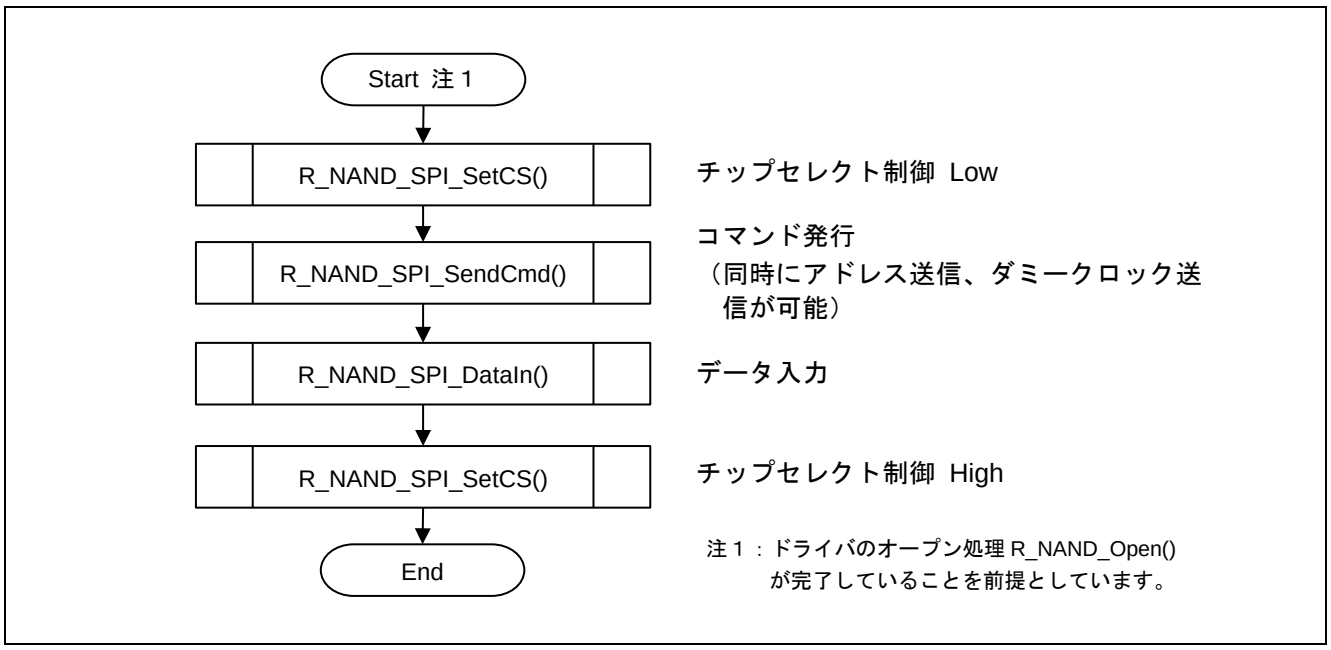


図 1-7　API コール例 「コマンド発行＋データ入力」

(c)　コマンド発行＋データ出力の場合

図 1-8 に「コマンド発行＋データ出力」の API コール例を示します。

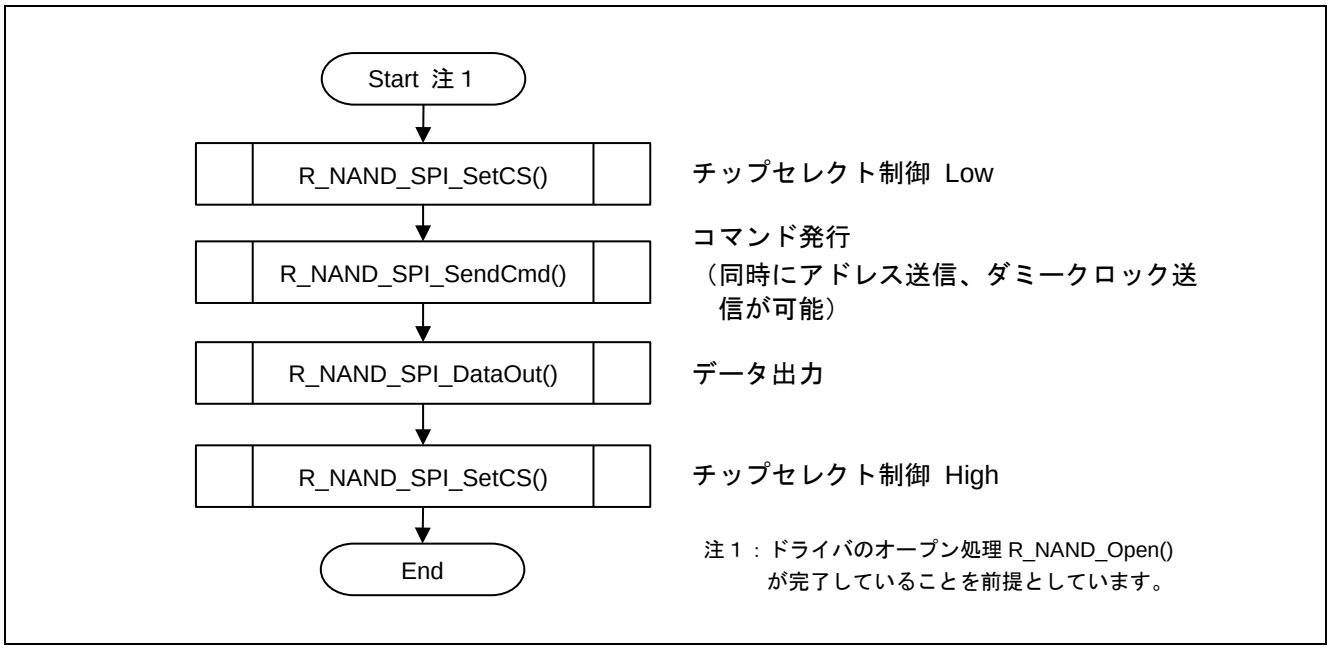


図 1-8　API コール例 「コマンド発行＋データ出力」

(2)　チップセレクト制御

表 1-9 に API コールとチップセレクト制御の関係を示します。

表 1-9　API コールとチップセレクト制御の関係

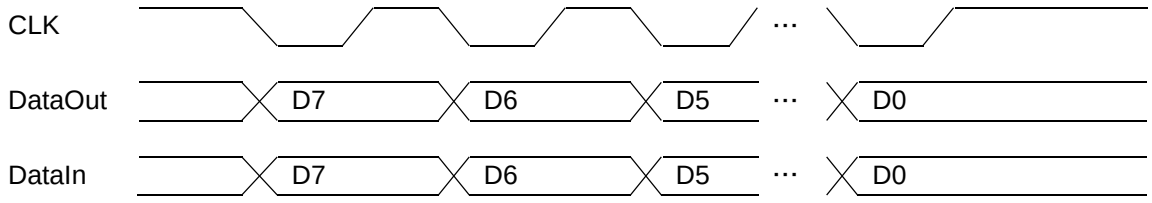
API コール	状態遷移		チップセレクト信号状態	注意事項
	API コール前	API コール後		
R_NAND_SPI_Open()	未初期化	アイドル	Hi-Z -> High	無し
R_NAND_SPI_SetCS()	アイドル	通信	High -> Low	チップセレクトセットアップ時間のウェイト処理は、別途ユーザで行ってください。
R_NAND_SPI_SetCS()	通信	アイドル	Low -> High	チップセレクトホールド時間のウェイト処理は、別途ユーザで行ってください。
R_NAND_SPI_Close()	アイドル	未初期化	High	本 API をコールしても、チップセレクト端子の状態は Hi-Z になりません。端子状態を変更したい場合は、別途ユーザで行ってください。

(3)　シリアル通信制御

MCU のシリアル通信機能を使用し、クロック同期式シングルマスタ制御を実現します。シリアル通信のクロック極性と位相は SPI モード 3（CPOL=1、CPHA=1）で制御します。

(a)　Single-SPI 制御時

図 1-9 に Single-SPI 制御時の通信タイミングを示します。

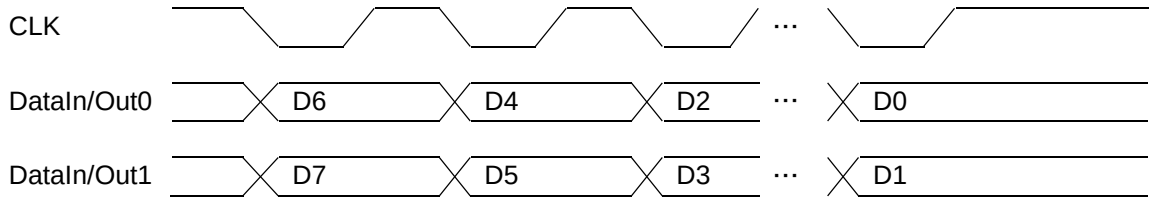


・MCU->スレーブデバイスの送信時：転送クロックの立ち下がりで送信データ出力開始
・スレーブデバイス->MCU の受信時：転送クロックの立ち上がりで受信データの入力取り込み
・MSB ファーストでの転送
・転送を行っていないときの CLK 端子のレベル：”H”

図 1-9　Single-SPI 制御時の通信タイミング

(b)　Dual-SPI 制御時

図 1-10 に Dual-SPI 制御時の通信タイミングを示します。

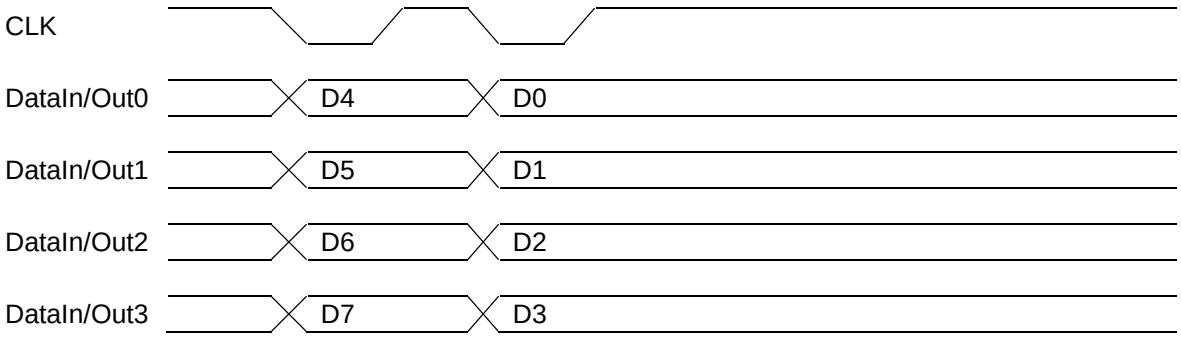


・MCU->スレーブデバイスの送信時：転送クロックの立ち下がりで送信データ出力開始
・スレーブデバイス->MCU の受信時：転送クロックの立ち上がりで受信データの入力取り込み
・MSB ファーストでの転送
・転送を行っていないときの CLK 端子のレベル：”H”

図 1-10　Dual-SPI 制御時の通信タイミング

(c)　Quad-SPI 制御時

図 1-11 に Quad-SPI 制御時の通信タイミングを示します。



・MCU->スレーブデバイスの送信時：転送クロックの立ち下がりで送信データ出力開始
・スレーブデバイス->MCU の受信時：転送クロックの立ち上がりで受信データの入力取り込み
・MSB ファーストでの転送
・転送を行っていないときの CLK 端子のレベル：”H”

図 1-11　Quad-SPI 制御時の通信タイミング

(4)　データバッファと送信／受信データの関係

図 1-12 に MCU の内蔵 RAM に格納するデータ配置と送信／受信の関係を示します。

本制御ソフトウェアは、ブロック型デバイスドライバです。エンディアンや使用するシリアル通信機能に関係なく、内蔵 RAM の先頭番地に配置されたデータから順番に送信し、また、受信した順番に内蔵 RAM の先頭番地から順番にデータを格納します。

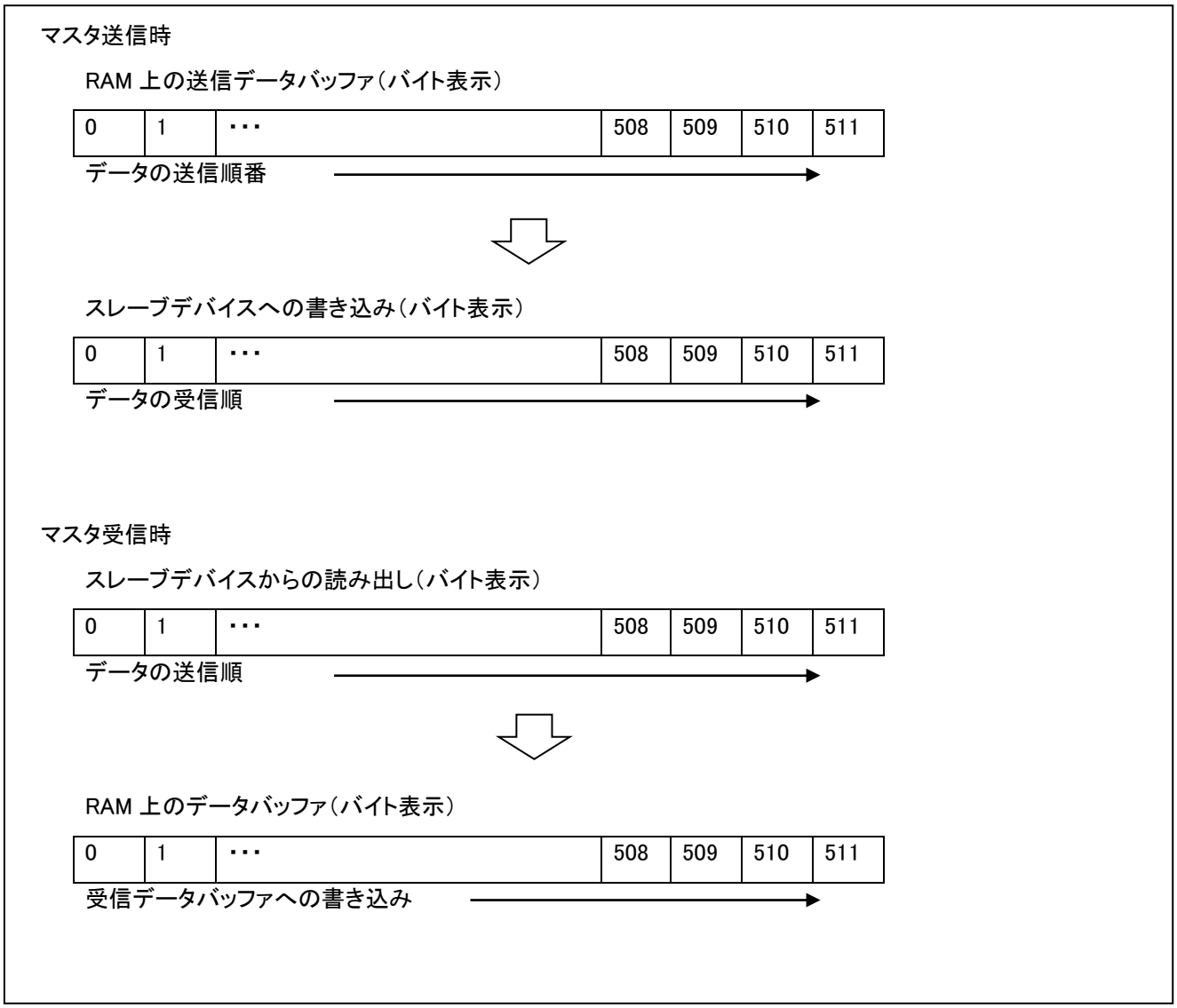


図 1-12　データバッファと送信／受信データの関係

1.5　状態遷移

図 1-13 に API の状態遷移を示します。

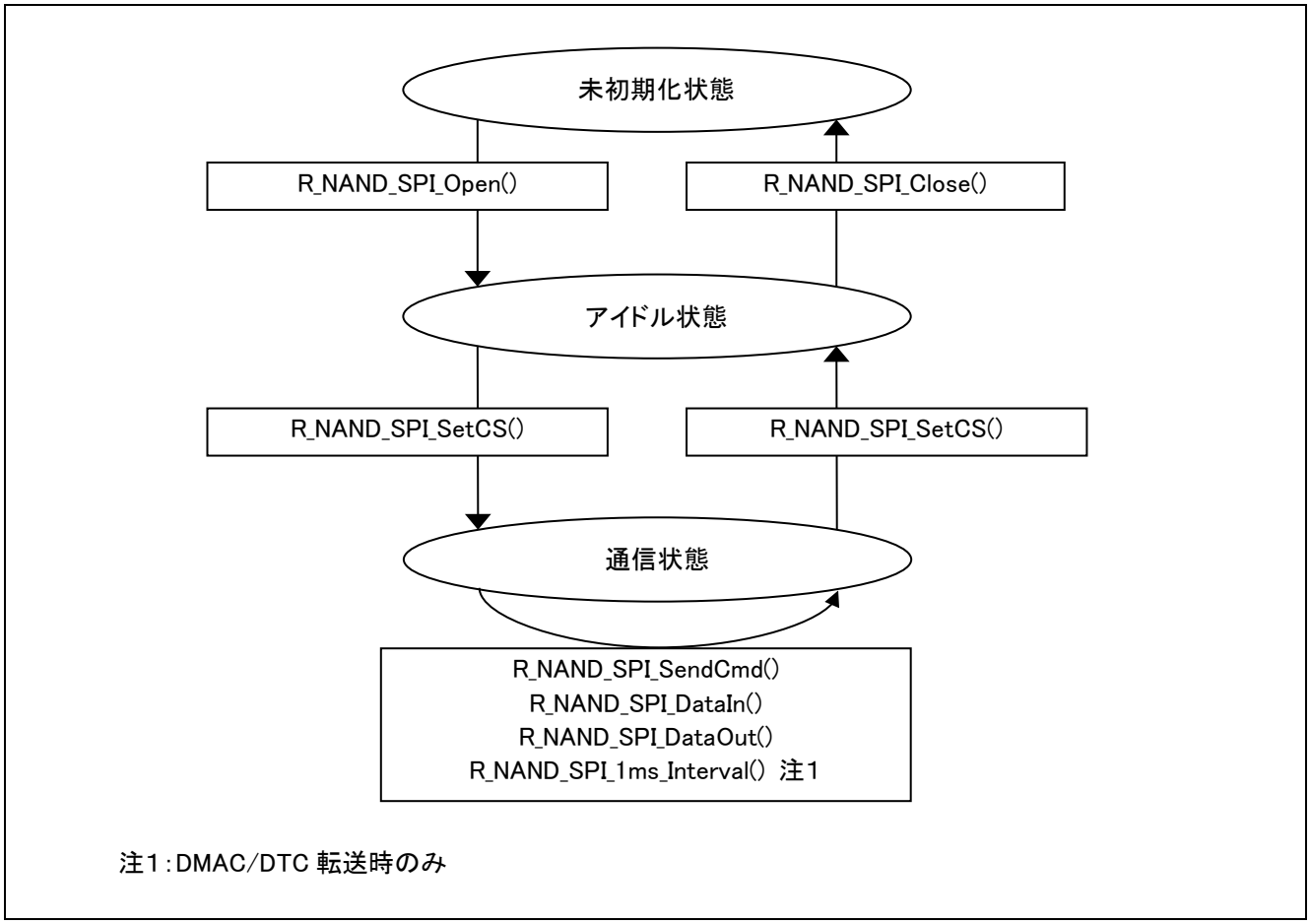


図 1-13　状態遷移図

2. API 情報

本制御ソフトウェアの API は、ルネサス エレクトロニクス社の API 命名基準に従っています。

2.1 ハードウェアの要求

ご使用になるマイコンが以下の機能をサポートしている必要があります。

- シリアル通信（クロック同期式シングルマスタ制御）
- I/O ポート
- データ転送（DMAC／DTC を使用する場合）
- タイマ（DMAC／DTC を使用する場合）

2.2 ソフトウェアの要求

このドライバは以下のパッケージに依存しています。

- r_bsp
- r_rspi_smstr_rx（クロック同期式シングルマスタ制御に RSPI SMstr FIT モジュールを使用する場合）
- r_qspi_smstr_rx（クロック同期式シングルマスタ制御に QSPI SMstr FIT モジュールを使用する場合）
- r_gpio_rx
- r_mpc_rx
- r_dtc_rx（データ転送に DTC FIT モジュールを使用する場合のみ）
- r_dmaca_rx（データ転送に DMACA FIT モジュールを使用する場合のみ）
- r_cmt_rx（データ転送に DTC FIT モジュールもしくは DMACA FIT モジュールを使用し、かつ、タイマに 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_longq（エラーログ取得機能を有効にし、LONGQ FIT モジュールを使用する場合のみ）

2.3 サポートされているツールチェイン

本制御ソフトウェアは、「6.1 動作確認環境詳細」に示すツールチェイ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

DMAC 転送または DTC 転送を行う場合、割り込みが有効になります。割り込みの詳細については、MCU のシリアル通信機能（クロック同期式シングルマスタ）を制御するデバイスドライバのアプリケーションノートをご参照ください。

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_nand_spi_if.h に記載しています。
ビルド毎の構成オプションは、r_nand_spi_config.h で選択します。以下の順番でインクルードしてください。

```
#include "r_nand_spi_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

本制御ソフトウェアのコンフィギュレーションオプションの設定は、r_nand_spi_config.h と r_nand_spi_pin_config.h で行います。

オプション名および設定値に関する説明を下表に示します。なお、表中の DEVx は Serial NAND Flash memory のデバイス番号を示します（x=0 or 1）。

表 2-1 Configuration options(pin_config.h)

Configuration options in *r_nand_spi_pin_config.h*	
NAND_SPI_DEVx_GPIO_CS ※デフォルトは“GPIO_PORT_C_PIN_0”	チップセレクト端子を制御する汎用ポート番号を設定します。デフォルト値はポート C0 を使用する場合です。制御するポートに合わせて設定してください。

表 2-2 Configuration options(config.h)

Configuration options in *r_nand_spi_config.h*	
#define NAND_SPI_CFG_DEVx_INCLUDED ※DEV0 のデフォルト値は、“1（有効）”	DEVx 制御に関する定義です。（1：有効、0：無効） 最低でも 1 デバイスは有効にしてください。
#define NAND_SPI_CFG_DEVx_MODE_TRNS ※DEV0 のデフォルト値は以下のとおりです。 NAND_SPI_TRNS_CPU	MCU 内蔵 RAM へのデータ転送方式を定義します。 以下から 1 つ設定してください。 NAND_SPI_TRNS_CPU：CPU 転送（Software 転送） NAND_SPI_TRNS_DMAC：DMAC 転送（注 1） NAND_SPI_TRNS_DTC：DTC 転送（注 2）
#define NAND_SPI_CFG_DEVx_MODE_DRVR ※DEV0 のデフォルト値は以下のとおりです。 NAND_SPI_DRVR_RX_FIT_RSPI_SMSTR	使用するデバイスドライバを定義します。 以下から 1 つ設定してください。 NAND_SPI_DRVR_RX_FIT_RSPI_SMSTR： RSPI クロック同期式シングルマスタ制御 FIT モジュール NAND_SPI_DRVR_RX_FIT_QSPI_SMSTR： QSPI クロック同期式シングルマスタ制御 FIT モジュール
#define NAND_SPI_CFG_DEVx_MODE_DRVR_CH ※DEV0 のデフォルト値は以下のとおりです。 NAND_SPI_DRVR_CH0	使用するデバイスドライバのチャネル番号を定義します。 以下から 1 つ選択してください（CH0 - CH15）。 NAND_SPI_DRVR_CH0 NAND_SPI_DRVR_CH1 ・・・ NAND_SPI_DRVR_CH15
#define NAND_SPI_CFG_DEVx_BR ※DEV0 のデフォルト値は“0x10”	コマンド発行する際のビットレート設定です。 シリアル通信機能（RSPI or QSPI）のビットレートレジスタに書き込む値を設定してください。
#define NAND_SPI_CFG_DEVx_BR_WRITE_DATA ※DEV0 のデフォルト値は“0x10”	データ出力する際のビットレート設定です。 シリアル通信機能（RSPI or QSPI）のビットレートレジスタに書き込む値を設定してください。
#define NAND_SPI_CFG_DEVx_BR_READ_DATA ※DEV0 のデフォルト値は“0x10”	データ入力する際のビットレート設定です。 シリアル通信機能（RSPI or QSPI）のビットレートレジスタに書き込む値を設定してください。
#define NAND_SPI_CFG_DEVx_DMAC_CH_NO_Tx ※DEV0 のデフォルト値は“0”	DAMC FIT モジュールを使用する場合、DMAC チャネル番号を設定してください。

	この DMAC チャネルは、MCU の内蔵 RAM からシリアル通信機能のデータバッファへデータを転送する時に使用します。
#define NAND_SPI_CFG_DEVx_DMAC_CH_NO_Rx ※DEV0 のデフォルト値は“1”	DAMC FIT モジュールを使用する場合、DMAC チャネル番号を設定してください。 この DMAC チャネルは、シリアル通信機能のデータバッファから MCU の内蔵 RAM へデータを転送する時に使用します。
#define NAND_SPI_CFG_DEVx_DMAC_INT_PRL_Tx ※DEV0 のデフォルト値は“10”	DAMC FIT モジュールを使用する場合、DMAC 割り込み優先レベルを設定してください。 この DMAC チャネルは、MCU の内蔵 RAM からシリアル通信機能のデータバッファへデータを転送する時に使用します。
#define NAND_SPI_CFG_DEVx_DMAC_INT_PRL_Rx ※DEV0 のデフォルト値は“10”	DAMC FIT モジュールを使用する場合、DMAC 割り込み優先レベルを設定してください。 この DMAC チャネルは、シリアル通信機能のデータバッファから MCU の内蔵 RAM へデータを転送する時に使用します。
#define NAND_SPI_CFG_PARAM_CHECKING_ENABLE ※デフォルトは “BSP_CFG_PARAM_CHECKING_ENABLE”	パラメータエラーチェックの定義です。（1：有効、0：無効）
#define NAND_SPI_CFG_LONGQ_ENABLE ※デフォルトは“0（無効）”	FIT モジュールの BSP 環境で使用する場合、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。（1：有効、0：無効） 無効にした場合、処理をコードから省略します。 有効にした場合、処理をコードに含めます。 使用するためには、別途 LONGQ FIT モジュールが必要です。 また、デバイスドライバの#define xxx_LONGQ_ENABLE を有効にしてください。

注１：別途、DMACA FIT モジュールが必要です。
注２：別途、DTC FIT モジュールが必要です。

2.8　コードサイズ

表 2-3 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。

表 2-3 コードサイズ

MCU	使用メモリ	サイズ（注１、注２、注３、注４）
RX65N	ROM	2,077 バイト
	RAM	20 バイト
	最大使用ユーザスタック	56 バイト
	最大使用割り込みスタック	-
RX71M	ROM	2,077 バイト
	RAM	20 バイト
	最大使用ユーザスタック	56 バイト
	最大使用割り込みスタック	-

注１：「2.7　コンパイル時の設定」のデフォルト設定を選択した場合の値です。選択する定義により、コードサイズは異なります。

注２：動作条件は以下のとおりです。

- r_nand_spi.c
- r_nand_spi_drvif_common.c
- r_nand_spi_drvif_dmac.c
- r_nand_spi_drvif_dtc.c
- r_nand_spi_drvif_qspi_smstr.c
- r_nand_spi_sub.c

注３：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４：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_nand_spi_if.h で記載されています。

```
/* NAND command information */
typedef struct
{
    uint8_t        cmd;       /* Command */
    uint8_t        rsv[3];    /* Reserved */
    uint32_t       addr;      /* Address to issue the command */
    uint32_t       addr_cnt;  /* Bytes of address count */
    uint32_t       dummy_cnt; /* Bytes of dummy count */
} st_nand_spi_cmd_info_t;         /* 16 bytes */

/* NAND data information */
typedef struct
{
    uint32_t       cnt;       /* Number of bytes */
    uint8_t      * p_data;    /* Data storage buffer pointer */
    uint8_t        io_mode;   /* Single/Dual/Quad */
    uint8_t        rsv[3];    /* Reserved */
} st_nand_spi_data_info_t;        /* 12 bytes */
```

2.10　戻り値

API 関数の戻り値を示します。この構造体は API 関数のプロトタイプ宣言とともに r_nand_spi_if.h で記載されています。

```
/* Enumeration for return values */
typedef enum e_nand_spi_status
{
    NAND_SPI_SUCCESS     = 0,  /* Successful operation */
    NAND_SPI_ERR_PARAM   = -1, /* Parameter error */
    NAND_SPI_ERR_HARD    = -2, /* Hardware error */
    NAND_SPI_ERR_WP      = -4, /* Write-protection error */
    NAND_SPI_ERR_TIMEOUT = -6, /* Time out error */
    NAND_SPI_ERR_OTHER   = -7  /* Other error */
} e_nand_spi_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、Smart Configurator を使用した(1)、(3)の追加方法を推奨しています。ただし、Smart Configurator 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 Smart Configurator を使用して FIT モジュールを追加する場合
e² studio の Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」を参照してください。

(2) e² studio 上で FIT Configurator を使用して FIT モジュールを追加する場合
e² studio の FIT Configurator 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 Smart Configurator 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 Smart Configurator 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「Renesas e² studio スマート・コンフィグレータ ユーザーガイド (R20AN0451)」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。

3. API 関数

3.1 R_NAND_SPI_Open()

NAND SPI FIT モジュールを初期化する関数です。他の API 関数を使用する前に実行してください。

Format

```
e_nand_spi_status_t R_NAND_SPI_Open(
        void
)
```

Parameters

なし

Return Values

NAND_SPI_SUCCESS	*正常終了*
NAND_SPI_ERR_OTHER	*エラー*

Properties

ファイル r_nand_spi_if.h にプロトタイプ宣言されています。

Description

チップセレクト端子を汎用出力ポートに設定し、High 出力状態にします。
また、シリアル通信、データ転送（DTC／DMAC 選択時のみ）で使用する FIT モジュールを初期化します。2 個のスレーブデバイスを制御する場合、両デバイスで使用する FIT モジュールを初期化します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
   e_nand_spi_status_t  ret = NAND_SPI_SUCCESS;

   ret = R_NAND_SPI_Open();
   if (NAND_SPI_SUCCESS > ret)
     {
       r_nand_spi_demo_trap();
   }
```

Special Notes:

なし

3.2　R_NAND_SPI_Close()

NAND SPI FIT モジュール制御を終了する際に使用する関数です。

Format

```
e_nand_spi_status_t R_NAND_SPI_Close(
    void
)
```

Parameters

なし

Return Values

NAND_SPI_SUCCESS	*正常終了*
NAND_SPI_ERR_OTHER	*エラー*

Properties

r_nand_spi_if.h にプロトタイプ宣言されています。

Description

チップセレクト端子を汎用出力ポートに設定します。
また、シリアル通信、データ転送（DTC／DMAC 選択時のみ）で使用する FIT モジュールのリソースを解放します。2 個のスレーブデバイスを制御する場合、両デバイスで使用する FIT モジュールのリソースを解放します。
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されません。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
e_nand_spi_status_t  ret = NAND_SPI_SUCCESS;

ret = R_NAND_SPI_Close();
if (NAND_SPI_SUCCESS > ret)
{
    r_nand_spi_demo_trap();
}
```

Special Notes:

本 API をコールしても、チップセレクト端子の状態は Hi-Z になりません。端子状態を変更したい場合は、別途ユーザで行ってください。

3.3　R_NAND_SPI_SendCmd()

コマンドシーケンスを実行する関数です。

API のコール手順は「1.4.4(1) API コール手順」を参照してください。

Format

```
e_nand_spi_status_t R_NAND_SPI_SendCmd(
    uint8_t devno,
    st_nand_spi_cmd_info_t * p_nand_spi_cmd_info
)
```

Parameters

devno

デバイス番号（0, 1）

** p_nand_spi_cmd_info*

コマンド情報構造体

コマンドにより設定値が異なります。表 3-1 を参照して設定してください。

cmd

コマンド。設定可能範囲は 0～255 です。

addr

アドレス。設定可能範囲は 0～4,294,967,295 です。addr_cnt = 0 の場合、本設定は無視されます。

addr_cnt

アドレスバイト数。設定可能範囲は 0～4,294,967,295 です。アドレス送信が不要な場合は 0 にしてください。

dummy_cnt

ダミーデータバイト数。設定可能範囲は 0～4,294,967,295 です。ダミーデータ送信が不要な場合は 0 にしてください。

turn Values

NAND_SPI_SUCCESS	*正常終了*
NAND_SPI_ERR_PARAM	*パラメータエラー*
NAND_SPI_ERR_HARD	*ハードエラー*
NAND_SPI_ERR_OTHER	*その他エラー*

Properties

r_nand_spi_if.h にプロトタイプ宣言されています。

Description

Serial NAND Flash memory に対してコマンド発行、アドレス送信、ダミーデータ送信を実行します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
e_nand_spi_status_t  ret = NAND_SPI_SUCCESS;
st_nand_spi_cmd_info_t nand_cmd_info;

nand_cmd_info.cmd      = NAND_SPI_CMD_TC58_READ_ID;
nand_cmd_info.addr     = 0;
nand_cmd_info.addr_cnt  = 0;
nand_cmd_info.dummy_cnt = 1;
ret = R_NAND_SPI_SendCmd(gs_nand_spi_demo_devno, &nand_cmd_info);
if (NAND_SPI_SUCCESS > ret)
{
    r_nand_spi_demo_trap();
}
```

Special Notes:

表 3-1 に東芝メモリ社製 Serial Interface NAND を使用する場合の* p_nand_spi_cmd_info 設定情報を示します。

なお、p_nand_spi_cmd_info の addr／addr_cnr／dummy_cnt に関しては、設定上限値のパラメータチェックを行っていません。そのため、制御対象の Serial NAND Flash memory の上限値を上回る値を設定した場合でも、そのまま処理を実行します。

表 3-1　東芝メモリ社製 Serial Interface NAND * p_nand_spi_cmd_info 設定情報

コマンド		* p_nand_spi_cmd_info			
名称	値	cmd （r_nand_spi_rx_if.h のマクロ定義）	addr 注 1 （設定可能範囲： 0 - 4,294,967,295）	addr_cnt 注 1 （設定可能範囲： 0 - 4,294,967,295）	dummy_cnt （設定可能範囲： 0 - 4,294,967,295）
Read Cell Array	13h	NAND_SPI_CMD_TC58_READ_CELL_ARRAY	Row Address	3	0
Read Buffer	03h	NAND_SPI_CMD_TC58_READ_BUFF	Column Address	2	1
Read Buffer x2	3Bh	NAND_SPI_CMD_TC58_READ_BUFF_X2	Column Address	2	1
Read Buffer x4	6Bh	NAND_SPI_CMD_TC58_READ_BUFF_X4	Column Address	2	1
Program Load	02h	NAND_SPI_CMD_TC58_PROGRAM_LOAD	Column Address	2	0
Program Execute	10h	NAND_SPI_CMD_TC58_PROGRAM_EXECUTE	Row Address	3	0
Protect Execute	2Ah	NAND_SPI_CMD_TC58_PROTECT_EXECUTE	Row Address	3	0
Program Load Random Data	84h	NAND_SPI_CMD_TC58_PROGRAM_LOAD_RANDOM_DATA	Column Address	2	0
Block Erase	D8h	NAND_SPI_CMD_TC58_BLOCK_ERASE	Row Address	3	0
Reset	FEh	NAND_SPI_CMD_TC58_RESET	0	0	0
Write Enable	06h	NAND_SPI_CMD_TC58_WRITE_ENABLE	0	0	0
Write Disable	04h	NAND_SPI_CMD_TC58_WRITE_DISABLE	0	0	0
Get Feature	0Fh	NAND_SPI_CMD_TC58_GET_FEATURE	1 byte address	1	0
Set Feature	1Fh	NAND_SPI_CMD_TC58_SET_FEATURE	1 byte address	1	0
Read ID	9Fh	NAND_SPI_CMD_TC58_READ_ID	0	0	0

注 1：設定可能範囲はソフトウェアとしての定義です。実際に設定可能な最大サイズについては、Serial NAND Flash memory のデータシートを参照してください。

3.4　R_NAND_SPI_DataIn()

Serial NAND Flash memory からデータを受信する関数です。

API のコール手順は「1.4.4(1) API コール手順」を参照してください。

Format

```
e_nand_spi_status_t R_NAND_SPI_DataIn(
    uint8_t devno,
    st_nand_spi_data_info_t * p_nand_spi_data_info
)
```

Parameters

devno

デバイス番号（0, 1）

* p_nand_spi_data_info

データ情報構造体

cnt

データバイト数。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、以下の設定にしてください。以下の設定以外の場合、強制的に CPU 転送を実行します。

RSPI と DMAC／DTC の組み合わせ：設定値は 4 の倍数としてください。

QSPI と DMAC／DTC の組み合わせ：設定値は 16 の倍数としてください。

* p_data

受信データ格納バッファのポインタアドレス。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。4 バイト境界以外のアドレスを指定した場合は、強制的に CPU 転送を実行します。

io_mode

バス幅設定。以下から 1 つ設定してください。

NAND_SPI_MODE_SINGLE　：1bit バス
NAND_SPI_MODE_DUAL　：2bit バス
NAND_SPI_MODE_QUAD　：4bit バス

Return Values

NAND_SPI_SUCCESS	*正常終了*
NAND_SPI_ERR_PARAM	*パラメータエラー*
NAND_SPI_ERR_HARD	*ハードエラー*
NAND_SPI_ERR_OTHER	*その他エラー*

Properties

r_nand_spi_if.h にプロトタイプ宣言されています。

Description

Serial NAND Flash memory からデータを受信します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
e_nand_spi_status_t  ret = NAND_SPI_SUCCESS;
st_nand_spi_data_info_t nand_data_info;

nand_data_info.cnt    = 128;
nand_data_info.p_data  = &buff[0];
nand_data_info.io_mode = NAND_SPI_MODE_QUAD;
ret = R_NAND_SPI_DataIn(gs_nand_spi_demo_devno, &nand_data_info);
if (NAND_SPI_SUCCESS > ret)
{
   r_nand_spi_demo_trap();
}
```

Special Notes:

なし。

3.5　R_NAND_SPI_DataOut()

Serial NAND Flash memory へデータを送信する関数です。

API のコール手順は「1.4.4(1) API コール手順」を参照してください。

Format

```
e_nand_spi_status_t R_NAND_SPI_DataOut(
    uint8_t devno,
    st_nand_spi_data_info_t * p_nand_spi_data_info
)
```

Parameters

devno

デバイス番号（0, 1）

* p_nand_spi_data_info

データ情報構造体

cnt

データバイト数。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、以下の設定にしてください。以下の設定以外の場合、強制的に CPU 転送を実行します。

RSPI と DMAC／DTC の組み合わせ：設定値は 4 の倍数としてください。

QSPI と DMAC／DTC の組み合わせ：設定値は 16 の倍数としてください。

* p_data

送信データ格納バッファのポインタアドレス。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。4 バイト境界以外のアドレスを指定した場合は、強制的に CPU 転送を実行します。

io_mode

バス幅設定。以下から 1 つ設定してください。

NAND_SPI_MODE_SINGLE　：1bit バス
NAND_SPI_MODE_DUAL　：2bit バス
NAND_SPI_MODE_QUAD　：4bit バス

Return Values

NAND_SPI_SUCCESS	*正常終了*
NAND_SPI_ERR_PARAM	*パラメータエラー*
NAND_SPI_ERR_HARD	*ハードエラー*
NAND_SPI_ERR_OTHER	*その他エラー*

Properties

r_nand_spi_if.h にプロトタイプ宣言されています。

Description

Serial NAND Flash memory へデータを送信します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
e_nand_spi_status_t  ret = NAND_SPI_SUCCESS;
st_nand_spi_data_info_t nand_data_info;

nand_data_info.cnt    = 128;
nand_data_info.p_data  = &buff[0];
nand_data_info.io_mode = NAND_SPI_MODE_QUAD;
ret = R_NAND_SPI_DataOut(gs_nand_spi_demo_devno, &nand_data_info);
if (NAND_SPI_SUCCESS > ret)
{
   r_nand_spi_demo_trap();
}
```

Special Notes:

なし。

3.6　R_NAND_SPI_SetCS()

チップセレクト端子を制御する関数です。

API のコール手順は「1.4.4(1) API コール手順」を参照してください。

Format

```
e_nand_spi_status_t R_NAND_SPI_SetCS(
    uint8_t devno,
    uint8_t cs
)
```

Parameters

devno

デバイス番号（0, 1）

cs

端子状態。以下から 1 つ選択してください。

NAND_SPI_HI　：チップセレクト端子を High にする。

NAND_SPI_LOW　：チップセレクト端子を Low にする。

Return Values

NAND_SPI_SUCCESS	*正常終了*
NAND_SPI_ERR_PARAM	*パラメータエラー*
NAND_SPI_ERR_HARD	*ハードエラー*
NAND_SPI_ERR_OTHER	*その他エラー*

Properties

r_nand_spi_if.h にプロトタイプ宣言されています。

Description

チップセレクト端子を制御します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
e_nand_spi_status_t  ret = NAND_SPI_SUCCESS;

ret = R_NAND_SPI_SetCS(gs_nand_spi_demo_devno, NAND_SPI_LOW);
if (NAND_SPI_SUCCESS > ret)
{
    r_nand_spi_demo_trap();
}
```

Special Notes:

チップセレクトセットアップ時間やチップセレクトホールド時間のウェイト処理は行いません。別途ユーザで行ってください。

3.7　R_NAND_SPI_GetVersion()

NAND SPI FIT モジュールのバージョン情報を取得する関数です。

Format

```
uint32_t R_NAND_SPI_GetVersion(
   void
)
```

Parameters

なし

Return Values

上位 2 バイト	*メジャーバージョン（10 進表示）*
開 2 バイト	*マイナーバージョン（10 進表示）*

Properties

ファイル r_nand_spi_if.h にプロトタイプ宣言されています。

Description

ドライバのバージョン情報を返します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
uint32_t    version = 0;

version = R_NAND_SPI_GetVersion();
```

Special Notes:

なし

3.8　R_NAND_SPI_SetLogHdlAddress()

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。

Format

```
sdsi_status_t  R_NAND_SPI_SetLogHdlAddress(
   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

Return Values

NAND_SPI_SUCCESS	*正常終了*

Properties

ファイル r_nand_spi_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 NAND SPI FIT モジュールに設定します。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
#define NAND_SPI_DEMO_ERR_LOG_SIZE (32)
#define NAND_SPI_DEMO_LONGQ_IGN_OVERFLOW    (1)

e_nand_spi_status_t     ret = NAND_SPI_SUCCESS;
static uint32_t         gs_nand_spi_demo_errlog[NAND_SPI_DEMO_ERR_LOG_SIZE];
longq_err_t             ret_longq;
static longq_hdl_t      ps_nand_spi_long_que;
uint32_t                long_que_hndl_address;


/* Open LONGQ module. */
ret_longq = R_LONGQ_Open(&gs_nand_spi_demo_errlog[0],
                    NAND_SPI_DEMO_ERR_LOG_SIZE,
                    NAND_SPI_DEMO_LONGQ_IGN_OVERFLOW,
                    &ps_nand_spi_long_que
);

long_que_hndl_address = (uint32_t)ps_nand_spi_long_que;
ret = R_NAND_SPI_SetLogHdlAddress(long_que_hndl_address);
```

Special Notes:

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_NAND_SPI_Open()関数をコールする前に処理を実行してください。
別途 LONG FIT モジュールを組み込んでください。

3.9　R_NAND_SPI_Log()

エラーログを取得する関数です。

Format

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

Parameters

flg

0x00000001（固定値）

fid

0x0000003f（固定値）

line

0x00001fff（固定値）

Return Values

0	*正常終了*

Properties

ファイル r_nand_spi_if.h にプロトタイプ宣言されています。

Description

エラーログを取得します。
エラーログ取得を終了する場合、コールしてください。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
#define NAND_SPI_DEMO_DRIVER_ID     (0x00000001)
#define NAND_SPI_DEMO_LOG_MAX       (0x0000003f)
#define NAND_SPI_DEMO_LOG_ADR_MAX   (0x00001fff)

   e_nand_spi_status_t  ret = NAND_SPI_SUCCESS;

   ret = R_NAND_SPI_Open();
   if (NAND_SPI_SUCCESS > ret)
   {
      /* Set last error log to buffer. */
      R_NAND_SPI_Log(
         NAND_SPI_DEMO_DRIVER_ID,
         NAND_SPI_DEMO_LOG_MAX,
         NAND_SPI_DEMO_LOG_ADR_MAX
      );
      r_nand_spi_demo_trap();
   }
```

Special Notes:

別途 LONG FIT モジュールを組み込んでください。

3.10　R_NAND_SPI_1msInterval()

クロック同期式シングルマスタ制御ソフトウェアのインターバルタイマカウンタ関数をコールする関数です。DMAC もしくは DTC を使用する場合、タイマを使用して 1ms 毎に本関数をコールしてください。

Format

```
void R_NAND_SPI_1msInterval(
   void
)
```

Parameters

なし

Return Values

なし

Properties

r_nand_spi_if.h にプロトタイプ宣言されています。

Description

DMAC もしくは DTC 転送完了待ち時にクロック同期式シングルマスタ制御ソフトウェアの内部タイマカウンタをインクリメントします。

Reentrant

異なるチャネルからリエントラントは可能です。

Example

```
void r_nand_spi_demo_cmt_callback(void * pdata)
{
   uint32_t channel = (uint32_t)pdata;

   if ((*channel) == gs_nand_spi_demo_cmt_channel)
   {
      R_NAND_SPI_1msInterval();
   }
}
```

Special Notes:

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

4. 端子設定

NAND SPI FIT モジュールを使用するためには、r_nand_spi_pin_config.h にてチップセレクト制御端子の設定が必要です。設定方法は「表 2-1」を参照してください。

なお、シリアル通信端子の設定については、使用するデバイスドライバの端子設定章をご確認ください。

5. デモプロジェクト

5.1 概要

デモプロジェクトは、Renesas Starter Kit+ for RX65N-2MB(以下 RSK RX65N-2MB)上で、Little Endian で動作確認をしています。東芝メモリ社製 Serial Interface NAND に対して、QSPI で接続し、消去／書き込み／読み出しを実行するデモプロジェクトです。デモプロジェクトでは、
NAND_SPI_DEMO_START_ROW_ADR で指定した物理ブロックアドレスに対して Bad ブロックチェックを行い、Bad ブロックでなければ、消去／書き込み／読み出しを実行します。

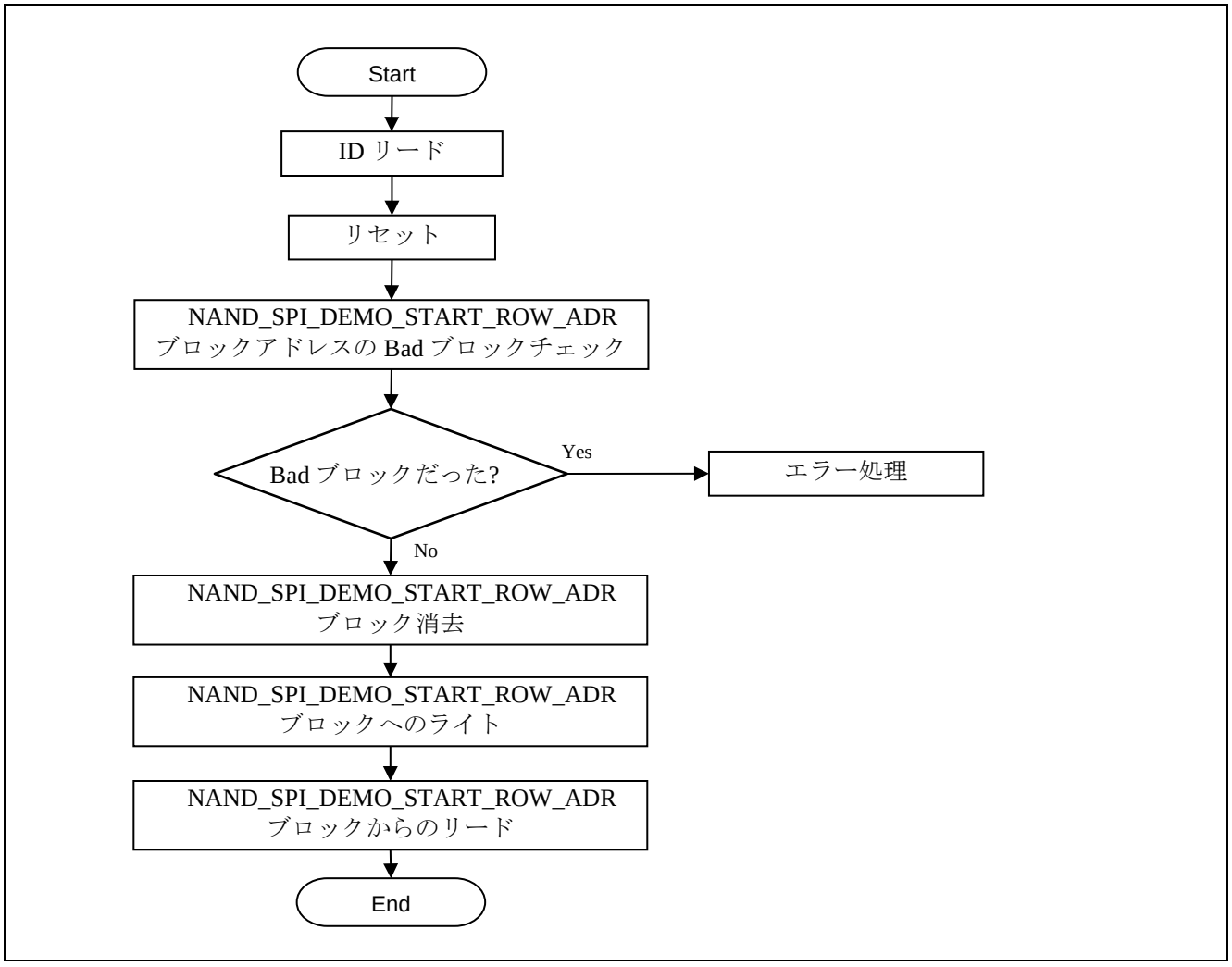


図 5-1　デモプロジェクト　Serial Interface NAND へのアクセスフロー

Serial NAND Flash と QSPI 接続時の RSK RX65N-2MB の接続時は SW4 を
表 5-1 にして、表 5-2 に従い JA3 に配線をしてください。

表 5-1　SW4 の設定

1	2	3	4
-	-	off	off

- : Don’t care

表 5-2　JA3 接続ピン番号

Serial NAND Flash 端子		RSK RX65N-2MB	
		コネクタ	ピン番号
CS	-	JA3	21
SI/SO0	-		23
SO/SO1	-		24
WP/SO2	-		19
HOLD/SO3	-		20
SCK	-		22

・r_nand_spi_pin_config.h の設定

チップセレクト制御端子として Port D4 を割り当てています。

#define NAND_SPI_DEV0_GPIO_CS　(GPIO_PORT_D_PIN_4)

5.2　API 関数

サンプルプログラム内の Serial NAND Flash を制御する API 関数を以下に示します。必要に応じて、関数の追加／修正してください。

表 5-3　API 関数一覧

関数名	機能概要
r_nand_spi_demo_init()	初期化処理
r_nand_spi_demo_init_user()	NAND SPI ドライバで使用する FIT の初期化
r_nand_spi_demo_set_block_lock()	Serial NAND Flash のブロックロックを設定
r_nand_spi_demo_erase()	Serial NAND Flash のブロックイレース
r_nand_spi_demo_read()	Serial NAND Flash リード(ECC ステータスチェック付)
r_nand_spi_demo_read_nocheck()	Serial NAND Flash リード(ECC ステータスチェックなし)
r_nand_spi_demo_write()	Serial NAND Flash ライト
r_nand_spi_demo_close()	終了処理
r_nand_spi_demo_cmt_callback()	

(1)　r_nand_spi_demo_init()

NAND SPI ドライバの Open 処理と Serial NAND Flash のリセット、ID リードを行う関数です。

Format

```
void r_nand_spi_demo_init(
    uint8_t *p_buff
)
```

Parameters

p_buff

Serial NAND ID　　Serial NAND の ID 格納領域ポインタ

Return Values

なし

Description

NAND SPI ドライバの Open 処理をおこない、Serial NAND Flash の ID をリード後、Serial NAND Flash をリセットします。

Special Notes

必要に応じて修正をして下さい。

(2)　r_nand_spi_demo_init_user()

NAND SPI ドライバで使用する他の FIT の初期化処理を行う関数です。

Format

```
void r_nand_spi_demo_init_user(
    void
)
```

Parameters

なし

Return Values

なし

Description

NAND SPI ドライバで使用する、他の FIT の初期化処理をおこないます。サンプルでは CMT の初期化を行います。

Special Notes

必要に応じて修正をして下さい。

(3)　r_nand_spi_demo_set_block_lock()

ロック指定する関数です。

Format

```
void r_nand_spi_demo_set_block_lock(
    uint8_t block_lock
)
```

Parameters

block_lock

Block lock コマンド設定	Block lock コマンド時にデバイス設定するデータ

Return Values

なし

Description

接続している Serial NAND Flash に Block Lock コマンドを送信します。
引数 block_lock に設定する値は Serial NAND Flash のデータシートを参照してください。

Special Notes

必要に応じて修正してください。
本関数実行前に r_nand_spi_demo_init()関数による初期化処理が必要です。

(4)　r_nand_spi_demo_erase()

イレースを行う関数です。

Format

```
void r_nand_spi_demo_erase(
    uint32_t row_addr
)
```

Parameters

row_addr

ROW アドレス	イレースする Serial NAND Flash の ROW アドレス

Return Values

なし

Description

接続した Serial NAND Flash の指定 ROW アドレスブロックをイレースします

Special Notes

必要に応じて修正してください。
本関数実行前に r_nand_spi_demo_init()関数による初期化処理が必要です。

(5)　r_nand_spi_demo_read()

指定アドレスのリードを行う関数です。リード後に ECC ステータスチェックを行います。

Format

```
void r_nand_spi_demo_read(
    uint32_t cnt,
    uint32_t row_addr,
    uint32_t column_addr,
    uint8_t *p_data
)
```

Parameters

cnt

リードバイト数

row_addr

リードを開始する ROW アドレス

column_addr

リードを開始する COLUMN アドレス

p_data

リードデータ格納バッファポインタ

Return Values

なし

Description

接続した Serial NAND Flash の指定 ROW,COLUMN アドレスから、指定バイト数分のリードを行います。
リード後に Get Feature コマンドを実行し、ECC ステータス値のチェックを行います。
ECC チェック後の処理は未実装です。必要に応じて、追加をしてください。

Special Notes

必要に応じて修正してください。
本関数実行前に r_nand_spi_demo_init()関数による初期化処理が必要です。

(6)　r_nand_spi_demo_read_nocheck()

指定アドレスのリードを行う関数です。リード後の ECC チェックを行いません。

Format

```
void r_nand_spi_demo_read_nocheck(
    uint32_t cnt,
    uint32_t row_addr,
    uint32_t column_addr,
    uint8_t *p_data
)
```

Parameters

cnt

リードバイト数

row_addr

リードを開始する ROW アドレス

column_addr

リードを開始する COLUMN アドレス

p_data

リードデータ格納バッファポインタ

Return Values

なし

Description

接続した Serial NAND Flash の指定 ROW,COLUMN アドレスから、指定バイト数分のリードを行います。
リード後に Get Feature コマンドを実行しますが、ECC ステータス値のチェックは行いません。

Special Notes

必要に応じて修正してください。
本関数実行前に r_nand_spi_demo_init()関数による初期化処理が必要です。

(7)　r_nand_spi_demo_write()

指定アドレスのライトを行う関数です。

Format

```
void r_nand_spi_demo_read_nocheck(
    uint32_t cnt,
    uint32_t row_addr,
    uint32_t column_addr,
    uint8_t *p_data
)
```

Parameters

cnt
　ライトバイト数
row_addr
　ライトを開始する ROW アドレス
column_addr
　ライトを開始する COLUMN アドレス
p_data
　ライトデータ格納バッファポインタ

Return Values

なし

Description

接続した Serial NAND Flash の指定 ROW,COLUMN アドレスから、指定バイト数分のライトを行います。

Special Notes

必要に応じて修正してください。
本関数実行前に r_nand_spi_demo_init()関数による初期化処理が必要です。

(8)　r_nand_spi_demo_close()

NAND SPI ドライバの Close 処理を行う関数です。

Format
void r_nand_spi_demo_close(
　　void
)

Parameters
なし

Return Values
なし

Description
NAND SPI ドライバの Close 処理を行います。

Special Notes
必要に応じて修正してください。
本関数実行前に r_nand_spi_demo_init()関数による初期化処理が必要です。

(9)　r_nand_spi_demo_cmt_callback()

CMT 割り込みで呼ばれる callback 関数です。

Format

```
void r_nand_spi_demo_cmt_callback(
    void
)
```

Parameters

なし

Return Values

なし

Description

CMT 割り込みで呼ばれる callback 関数です。R_NAND_SPI_1msInterval()を呼び出します。

Special Notes

必要に応じて修正してください。

5.3　デモプログラム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 動作確認環境詳細

表 6-1 に動作確認環境を示します。

表 6-1　動作確認環境

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 V7.1.0
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.2.08.00
	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.1.00
使用ボード	Renesas Starter Kit+ for RX71M（型名：R0K50571MSxxxxx） Renesas Starter Kit+ for RX65N-2MB (型名：R0K50565N2Sxxxxxxx)
Serial NAND Flash memory	東芝メモリ社製 Serial Interface NAND

6.2 トラブルシューティング

(1) 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。

(2) 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_nand_spi 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。

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
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）
テクニカルアップデート／テクニカルニュース
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）
ユーザーズマニュアル：開発環境
（最新の情報をルネサス エレクトロニクスホームページから入手してください。）

テクニカルアップデートの対応について

- テクニカルアップデートはありません。

ホームページとサポート窓口

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

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2018.10.31	-	新規作成

製品ご使用上の注意事項

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

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子
   （または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（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、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、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。
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。
6. 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。
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、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。
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。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 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。
10. 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。
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。
12. 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.4.0-1　2017.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/