RX ファミリ

M3S-TFAT-Tiny メモリドライバインタフェースモジュール

Firmware Integration Technology

要旨

本アプリケーションノートでは、Firmware Integration Technology (以降、FIT)を使用した M3S-TFAT-Tiny メモリドライバインタフェースについて説明します。

本モジュールは、FIT を使用した RX ファミリ オープンソース FAT ファイルシステム M3S-TFAT-Tiny と各メモリドライバとをつなぐメモリドライバインタフェースです。

FIT モジュールについては、

https://www.renesas.com/ja-jp/solutions/rx-applications/fit.html

をご覧ください。

本書では以下のとおり用語を使い分けます。

・TFAT FIT：

RX ファミリ用 オープンソース FAT ファイルシステム M3S-TFAT-Tiny モジュール FIT (R20AN0038)

・TFAT driver FIT：

RX ファミリ用 M3S-TFAT-Tiny メモリドライバインタフェースモジュール FIT (R20AN0335)

・TFAT：

M3S-TFAT-Tiny、または、TFAT FIT と TFAT driver FIT の総称

本アプリケーションノートで提供されるドライバインタフェースモジュールで対応しているメモリドライバは、SD メモリカード（SD モード）、USB メモリ、USB Mini、eMMC、Serial Flash memory の 5 つです。使用するには、下記 FIT モジュールが必要です。

機能	ミドルウェア製品	ウェブページ
ファイルシステム (※1)	オープンソース FAT ファイルシステム M3S-TFAT-Tiny モジュール	http://www.renesas.com/mw/tfat-rx
SD メモリカードドライバ（SD モード）(※2)	SD モード SD メモリカードドライバ	https://www.renesas.com/driver/rtm0rx0000dsdd
USB ドライバ (※2)	USB Basic Host and Peripheral Driver USB Host Mass Storage Class Driver (HMSC)	http://www.renesas.com/driver/usb
USB Mini ドライバ (※2)	USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) USB Host Mass Storage Class Driver for USB Mini Firmware	http://www.renesas.com/driver/usb
eMMC ドライバ(※2)	MMC モード MMCIF ドライバ	https://www.renesas.com/jp/ja/products/software-tools/software-os-middleware-driver/mmc/multimediacard-emmc-driver-for-rx-family.html
Serial Flash memory ドライバ(※2)	Serial Flash memory アクセスクロック同期式制御モジュール	https://www.renesas.com/in/ja/products/software-tools/software-os-middleware-driver/serial-memory/spi-qspi-serial-flash-driver.html

※1 必須です。
※2 どちらか一方の入手で構いません。お使いの評価環境に応じて入手してください。

対象デバイス

・RX ファミリ

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。

対象コンパイラ

・ Renesas Electronics C/C++ Compiler Package for RX Family
・ GCC for Renesas RX (RX23W はサポートしていません。)
・ 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

- RX ファミリ ボードサポートパッケージモジュール Firmware Integration Technology (R01AN1685)
- RX ファミリ オープンソース FAT ファイルシステム M3S-TFAT-Tiny モジュール Firmware Integration Technology (R20AN0038)
- RX ファミリ SD モード SD メモリカードドライバ Firmware Integration Technology (R01AN4233)
- RX ファミリ USB Basic Host and Peripheral Driver Firmware Integration Technology (R01AN2025)
- RX ファミリ USB Host Mass Storage Class Driver (HMSC) Firmware Integration Technology (R01AN2026)
- RX ファミリ USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) Firmware Integration Technology (R01AN2166)
- RX ファミリ USB Host Mass Storage Class Driver for USB Mini Firmware Firmware Integration Technology (R01AN2169)
- RX ファミリ MMC モード MMCIF ドライバ Firmware Integration Technology (R01AN4234)
- RX ファミリ Serial Flash memory アクセスクロック同期式制御モジュール Firmware Integration Technology (R01AN2662)
- RX ファミリ システムタイマモジュール Firmware Integration Technology (R20AN0431)

目次

1. 概要

1.1 本アプリケーションノートについて

本アプリケーションノートでは、TFAT FIT と各記憶デバイス用のメモリドライバとをつなぐメモリドライバインタフェースについて説明します。

メモリドライバインタフェースは、コンフィグファイルの変更によって制御対象を切り替えることができます。

本モジュールで提供する API は、TFAT FIT から呼び出されます。ユーザ側で新たに呼び出す必要はありません。

TFAT FIT で管理するドライブ番号と SD メモリカードドライバなどのデバイス・ドライバで管理するドライブ番号は等しくありません。その為、本モジュール内で変換テーブルを持っています。初期値は、コンフィグレーション設定で決まります。動的に変更したい場合は、3.7 drv_change_alloc()を参照してください。

1.2 アプリケーションの概要

1.2.1 アプリケーション構成

本アプリケーションノートは、以下のものから構成されています。

表 1.1 アプリケーションノート構成

ファイル/ディレクトリ名	内容
FITModules	
r_tfat_driver_rx_v2.20.xml	FIT プラグイン XML
r_tfat_driver_rx_v2.20_extend.mdf	スマート・コンフィグレータ 設定ファイル
r_tfat_driver_rx_v2.20.zip	FIT プラグイン ZIP
コンフィグレーション (r_config)	
r_tfat_driver_rx_config.h	コンフィグレーションファイル(デフォルト設定)
FIT Module 本体 (r_tfat_driver_rx)	
ドキュメント(doc)	
英語版(en)	
r20an0335ej0220-rx-tfat.pdf	アプリケーションノート(英語版)
日本語版(ja)	
r20an0335jj0220-rx-tfat.pdf	アプリケーションノート(日本語版)
コンフィグレーションリファレンス(ref)	
r_tfat_driver_rx_config_reference.h	コンフィグレーションファイル(テンプレート)
ソースコード(src)	
readme (readme.txt)	readme
r_tfat_driver_rx_if.h	ヘッダファイル

1.2.2 ソフトウェア構成

TFAT driver FIT は、TFAT FIT、システムタイマモジュール FIT、および、各種デバイス・ドライバ FIT を併用して動作します。

TFAT FIT はオープンソース FatFs を内部に含む、ファイルシステムの主モジュールです。TFAT driver FIT は内部に Wrapper 関数をもち、記憶デバイスごとにファイルシステム用の I/O 処理を切り替えます。ユーザは TFAT driver FIT のコンフィギュレーション設定で使用する記憶デバイスを設定し、TFAT FIT の API を介してファイルシステムを操作します。

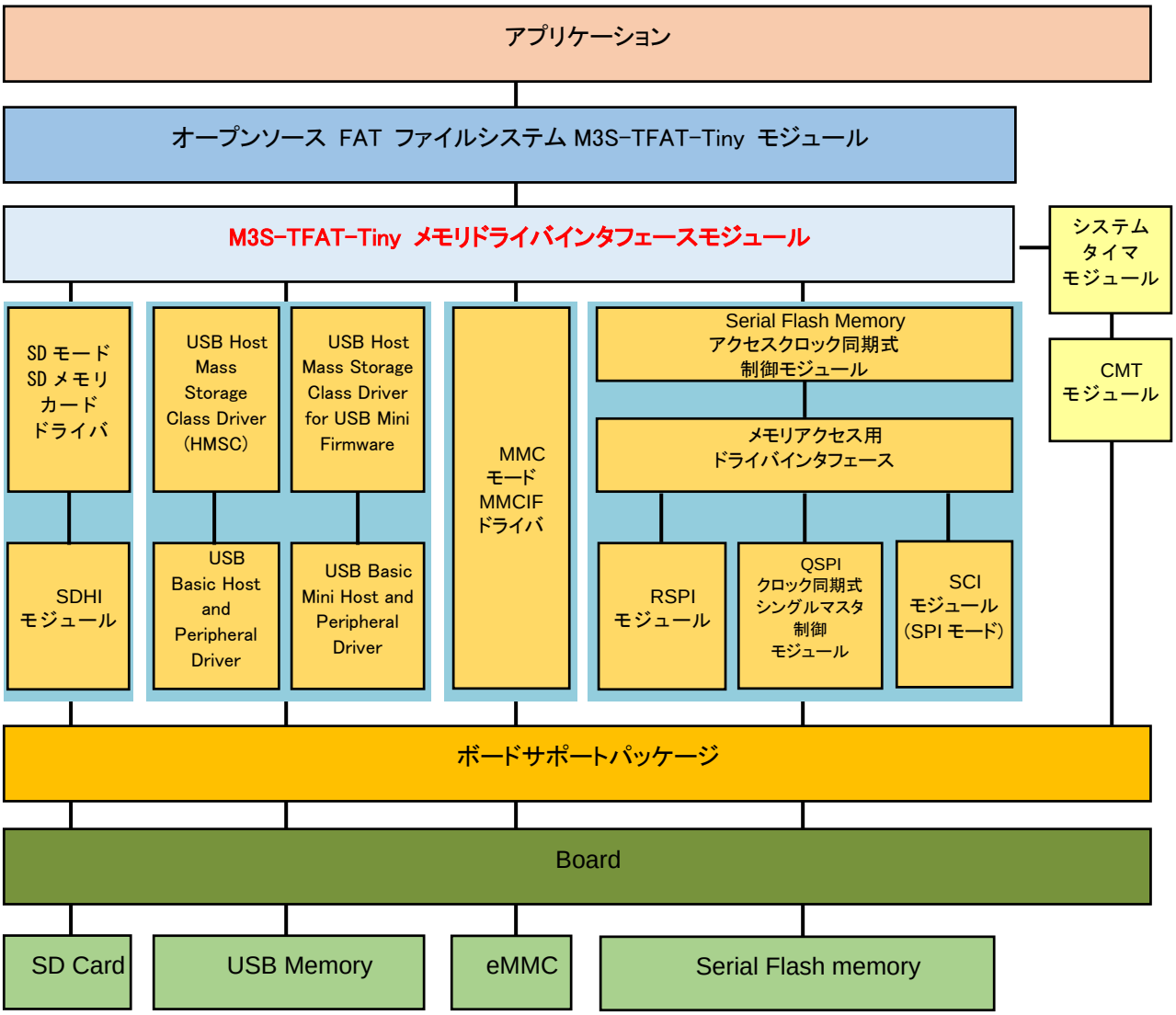

図 1-1 ソフトウェア構成図

表 1.2 使用した FIT モジュールバージョン

記憶デバイス	FIT モジュール	Rev.
共通	ボードサポートパッケージ(BSP)	5.52
	TFAT	4.02
	システムタイマモジュール CMT モジュール	1.01 4.40
SD カード （SD モード）	SD モード SD メモリカードドライバ SDHI モジュール	3.00 2.06
USB メモリ	USB Basic Host and Peripheral Driver USB Host Mass Storage Class Driver	1.30 1.30
	USB Basic Host and Peripheral Driver (USB Mini Firmware) USB Host Mass Storage Class Driver for USB Mini Firmware	1.20 1.20
eMMC	MMC モード MMCIF ドライバ	1.07
Serial Flash memory	Serial Flash memory アクセスクロック同期式制御モジュール メモリアクセス用ドライバインタフェース	3.01 1.02
	RSPI モジュール QSPI クロック同期式シングルマスタ制御モジュール SCI モジュール (SPI モード)	2.05 1.14 3.40

1.3 API の概要

表 1.3 に本 FIT モジュールに含まれる API 関数を示します。

表 1.3　API 関数一覧

関数	関数説明
disk_initialize()	ディスク・ドライブの初期化
disk_status()	ディスク・ドライブ状態取得
disk_read()	ディスクからの読み出し
disk_write()	ディスクへの書き込み
disk_ioctl()	その他のドライブ制御
get_fattime()	日付・時刻の取得
drv_change_alloc()	TFAT モジュールのドライブ番号とメモリドライバのドライブ番号との関連付け変更

1.4 制限事項

(1) TFAT の対象デバイスは、TFAT より下位層でユーザが使用する全ての FIT でサポートされているデバイスです。各 FIT の対象デバイスはそれぞれのアプリケーションノートをご覧ください。

(2) USB Basic Host and Peripheral Driver (USB Mini Firmware) FIT 及び USB Host Mass Storage Class Driver for USB Mini Firmware FIT はそれぞれ Rev.1.20 以上を使用してください。
これらの FIT の Rev.1.20 は RTOS に対応しているためです。

2. API 情報

2.1 ハードウェア要件

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。

- USB
- SDHI
- CMT

2.2 ソフトウェア要件

本 FIT モジュールは、以下のパッケージに依存しています。

- r_bsp (Rev.5.52 以上)
- r_tfat_rx (Rev.4.02 以上)
- r_sys_time_rx (Rev.1.01 以上)
- r_cmt_rx (Rev.4.40 以上)

使用するメモリドライバの種類は r_tfat_driver_rx_config.h で設定することができます。

2.3 サポートされているツールチェーン

本 FIT モジュールは、各メモリドライバがサポートしているツールチェーンに依存します。

2.4 使用する割り込みベクタ

TFAT driver FIT では割り込みベクタを使用しません。

2.5 ヘッダファイル

すべての API 呼び出しはこのソフトウェアのプロジェクトコードとして提供されている 1 個のファイル「r_tfat_driver_rx_if.h」をインクルードすることによって行われます。

ビルドタイムのコンフィグレーションオプションは、ファイル「r_tfat_driver_rx_config.h」で選択または定義されます。

2.6 整数型

このプロジェクトでは、コードをわかりやすく、移植性をより大きくするために、ANSI C99 の固定長整数型（exact width integer type）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 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_tfat_driver_rx_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。

Configuration options in r_tfat_driver_rx_config.h	
#define TFAT_USB_DRIVE_NUM - Default value = (0)	USB メモリとして割り当てるドライブ数 未使用時は(0)としてください。
#define TFAT_SDMEM_DRIVE_NUM - Default value = (0)	SD メモリカードとして割り当てるドライブ数 未使用時は(0)としてください。
#define TFAT_USB_MINI_DRIVE_NUM - Default value = (0)	USB メモリ（Mini）として割り当てるドライブ数 未使用時は(0)としてください。
#define TFAT_MMC_DRIVE_NUM - Default value = (0)	eMMC として割り当てるドライブ数 未使用時は(0)としてください。
#define TFAT_SERIAL_FLASH_DRIVE_NUM - Default value = (0)	Serial Flash memory として割り当てるドライブ数 未使用時は(0)としてください。
#define TFAT_DRIVE_ALLOC_NUM_i i = 0～9 - Default value = (TFAT_CTRL_NONE)	各ドライブ番号に対して使用するデバイスを割り当てます。 USB メモリで使用するドライブ = (TFAT_CTRL_USB) SD メモリカードで使用するドライブ= (TFAT_CTRL_SDMEM) USB メモリ(Mini)で使用するドライブ= (TFAT_CTRL_USB_MINI) eMMC で使用するドライブ = (TFAT_CTRL_MMC) Serial Flash memory で使用するドライブ = (TFAT_CTRL_SERIAL_FLASH) 使用しないドライブ = (TFAT_CTRL_NONE) としてください。 このデータを元に、TFAT FIT のドライブ番号とメモリドライバのドライブ番号との関連付けを行います。メモリドライバのドライブ番号は、昇順に割り当てられます。動的に変更したい場合は、3.7 drv_change_alloc() を参照してください。

#define RI600V4_MUTEX_ID_FOR_TFAT_ DRIVE_ALLOC_NUM_i i = 0～9 - Default value = (0)	RI600V4 を使用する場合、RI600V4 コンフィギュレーションで設定するミューテックス ID を指定してください。 このミューテックスは、TFAT の API がリエントラント性を獲得する（ドライブ上のファイル/ディレクトリの排他的アクセスを行う）ために使用されます。 RI600V4 使用時、かつ、当該ドライブ使用時は(1～255)の値としてください。各ドライブで異なる値（ミューテックス ID）を使用してください。 RI600V4、または、当該ドライブ未使用時は(0)としてください。各ドライブは(0)を重複できます。

2.8 コードサイズ

本 FIT モジュールの ROM サイズ、RAM サイズ、最大使用スタックサイズを下表に示します。RX100 シリーズ、RX200 シリーズ、RX600 シリーズから代表して 1 デバイスずつ掲載しています。

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_tfat_driver_rx Rev.2.20

コンパイラバージョン: Renesas Electronics C/C++ Compiler Package for RX Family V3.02.00
(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 8.3.0.202002
(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 4.14.1
(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

ROM、RAM およびスタックのコードサイズ

デバイス	分類	使用メモリ		
		Renesas Compiler	GCC	IAR Compiler
RX113	ROM(注)	6,487 バイト	13,168 バイト	8,443 バイト
	RAM(注)	26 バイト	28 バイト	58 バイト
	スタック(注)	368 バイト	-	380 バイト
RX231	ROM(注)	6,487 バイト	13,272 バイト	8,449 バイト
	RAM(注)	26 バイト	28 バイト	58 バイト
	スタック(注)	368 バイト	-	380 バイト
RX65N	ROM(注)	6,535 バイト	13,272 バイト	8,449 バイト
	RAM(注)	26 バイト	28 バイト	58 バイト
	スタック(注)	380 バイト	-	384 バイト

注：TFAT FIT モジュールの ROM/RAM/スタックサイズが含まれています。

2.9 引数

TFAT FIT の API を呼び出す際のドライブ番号の定義としてご使用ください。

```
typedef enum
{
    TFAT_DRIVE_NUM_0 = 0x00,
    TFAT_DRIVE_NUM_1,
    TFAT_DRIVE_NUM_2,
    TFAT_DRIVE_NUM_3,
    TFAT_DRIVE_NUM_4,
    TFAT_DRIVE_NUM_5,
    TFAT_DRIVE_NUM_6,
    TFAT_DRIVE_NUM_7,
    TFAT_DRIVE_NUM_8,
    TFAT_DRIVE_NUM_9,
}TFAT_DRV_NUM;
```

2.10 戻り値

それぞれ TFAT FIT モジュールの"diskio.h"で定義されています。

```
/* Disk Status Bits (DSTATUS) */
typedef uint8_t DSTATUS;
#define STA_NOINIT   0x01  /* Drive not initialized */
#define STA_NODISK   0x02  /* No medium in the drive */
#define STA_PROTECT  0x04  /* Write protected */

/* Results of Disk Functions */
typedef enum
{
  RES_OK = 0,   /* 0: Successful */
  RES_ERROR,    /* 1: R/W Error */
  RES_WRPRT,    /* 2: Write Protected */
  RES_NOTRDY,   /* 3: Not Ready */
  RES_PARERR    /* 4: Invalid Parameter */
} DRESULT;
```

2.11 FIT モジュールの追加方法

本 FIT 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) e^2 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
e^2 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: e^2 studio 編 (R20AN0451)」を参照してください。

(2) e^2 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
e^2 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 e^2 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。

3. API 関数

TFAT FIT モジュールから本関数群は呼び出されます。ここに記載された一部例外を除く関数は、コンフィグレーション設定に応じて、記憶デバイスの種類ごとに用意された下位層の関数を呼び分けます。下位層の関数は、「4 内部関数」に記載しています。
以下の関数をアプリケーション・プログラム側から呼び出さないでください。

表 3.1 関数一覧

関数名	機能概要
disk_initialize()	ディスク・ドライブの初期化
disk_status()	ディスク・ドライブの状態取得
disk_read()	ディスクからの読み込み
disk_write()	ディスクへの書き込み
disk_ioctl()	その他のドライブ制御
get_fattime()	日付・時刻の取得
drv_change_alloc()	TFAT モジュールのドライブ番号とメモリドライバのドライブ番号(もしくはチャンネル番号)との関連付けの変更

3.1 disk_initialize()

disk_initialize 関数はストレージデバイスを初期化するために呼び出されます。

Format

```
DSTATUS disk_initialize (
  BYTE pdrv         /* [IN] Physical drive number */
);
```

Parameters

pdrv

ターゲット・デバイスを識別するための物理ドライブ番号。 シングル・ドライブシステムでは常にゼロです。

Return Values

この関数は結果として現在のドライブステータスフラグを返します。 ドライブの状態の詳細については、disk_status 関数を参照してください。

Properties

ファイル diskio.h にプロトタイプ宣言されています。

Description

この機能は記録メディアを初期化し、それを通常のリード/ライト準備状態にします。関数が成功すると、戻り値の STA_NOINIT フラグがクリアされます。

備考：この関数は FatFs モジュールの管理下にある必要があります。アプリケーション・プログラムはこの関数を呼んではいけません。さもないと、ボリューム上の FAT 構造体が壊れる可能性があります。 ファイルシステムを再初期化するには、代わりに f_mount 関数を使用してください。

Example

なし

Special Notes:

なし

3.2 disk_status()

この関数は、現在のドライブステータスを問い合わせるために呼び出されます。

Format

```
DSTATUS disk_status (
  BYTE pdrv    /* [IN] Physical drive number */
);
```

Parameters

pdrv

ターゲット・デバイスを識別するための物理ドライブ番号。 シングル・ドライブシステムでは常にゼロです。

Return Values

現在のドライブステータスは、下記のステータスフラグの組み合わせで返されます。FatFs は STA_NOINIT と STA_PROTECT のみを参照します。

STA_NOINITI

デバイスが初期化されておらず、動作の準備ができていないことを示します。このフラグは、システムリセット、メディアの取り外し、または disk_initialize 関数の失敗時に設定されます。disk_initialize 関数が成功するとクリアされます。非同期に発生したメディアの変更はすべてキャプチャしてステータスフラグに反映させる必要があります。さもないと、自動マウント機能が正しく機能しません。システムがメディア変更検出をサポートしていない場合、アプリケーション・プログラムはメディア変更のたびに f_mount 関数でボリュームを明示的に再マウントする必要があります。

STA_NODISK

ドライブにメディアがないことを示します。これは固定ディスク・ドライブでは常にクリアされます。FatFs はこのフラグを参照しないことに注意してください。

STA_PROTECT

メディアが書き込み保護されていることを示します。これは書き込み保護機能のないドライブで常にクリアされます。 STA_NODISK が設定されている場合は無効です。

Properties

ファイル diskio.h にプロトタイプ宣言されています。

Description

なし

Example
なし

Special Notes:
なし

3.3 disk_read()

この関数は、記録メディアのセクタからデータを読み込むために呼び出されます。

Format

```
DRESULT disk_read (
  BYTE pdrv,     /* [IN] Physical drive number */
  BYTE* buff,    /* [OUT] Pointer to the read data buffer */
  DWORD sector,  /* [IN] Start sector number */
  UINT count     /* [IN] Number of sectors to read */
);
```

Parameters

pdrv

ターゲット・デバイスを識別するための物理ドライブ番号。

buff

読み出しデータを格納するためのバイト配列の最初の項目へのポインタ。読み出しデータ・サイズはセクタ・サイズ×count バイトとなります。

sector

開始セクタ番号。32 ビット LBA で指定します。

count

読み出すセクタ数。

Return Values

RES_OK（0）

正常終了。

RES_ERRORA

読み出し操作中に回復不能なハードエラーが発生しました。

RES_PARERR

パラメータが不正です。

RES_NOTRDY

デバイスは初期化されていません。

Properties

ファイル diskio.h にプロトタイプ宣言されています。

Description

メモリカード、ハードディスク、光ディスクなどの一般的な記録メディアに対する読み書き操作は、セクタと呼ばれるデータ・バイトのブロック単位で行われます。FatFs は 512 から 4096 バイトの範囲のセクタ・サイズをサポートします。 FatFs が固定セクタ・サイズ（FF_MIN_SS = FF_MAX_SS、これは最もよくあるケースです）に設定されている場合、読み書き関数はそのセクタ・サイズで機能しなければなりません。FatFs が可変セクタ・サイズに設定されている場合（FF_MIN_SS < FF_MAX_SS）、disk_initialize 関数が成功した直後に disk_ioctl 関数を使用してメディアのセクタ・サイズを問い合わせます。

buff を介して渡されるメモリアドレスについては、いくつか考慮事項があります。引数は BYTE *として定義されているため、必ずしもワード境界にアライメントされているわけではありません。アライメントされていない読み書き操作は直接転送で発生する可能性があります（FatFs アプリケーションノートを参照）。バスアーキテクチャ、特に DMA コントローラが、アライメントの合っていないメモリアクセスを許可しない場合は、disk_read 関数の中で解決する必要があります。その場合は、この問題を回避するために以下に説明するいくつかの回避策があります。

·必要に応じて、この関数でワード転送をバイト転送に変換します。 - おすすめです。

·f_read（）呼び出しで、セクタ全体を含む長い読み出し要求を避けます。 - 直接転送は発生しません。

·f_read（fp、dat、btw、bw）呼び出しで、(((UINT) dat & 3) == (f_tell(fp) & 3)) が true であることを確認してください。 - buff の Word アライメントは保証されています。

また、メモリ領域が DMA で到達できない可能性があります。これは、通常スタックに使用される密結合メモリ内にある場合です。ダブルバッファ転送を使用するか、スタック上のローカル変数として FATFS および FIL 構造体を含むファイル I / O バッファを定義しないでください。一般に、複数セクタの読み出し要求（count > 1）を記録メディアへの単一セクタ転送に分割しないでください。複数の単一セクタ転送に分解した場合、読み出しスループットが低下します。

Example

なし

Special Notes:

なし

3.4 disk_write()

この関数は、記録メディアのセクタにデータを書き込むために呼び出されます。

Format

```
DRESULT disk_write (
  BYTE pdrv,       /* [IN] Physical drive number */
  const BYTE* buff, /* [IN] Pointer to the data to be written */
  DWORD sector,    /* [IN] Sector number to write from */
  UINT count       /* [IN] Number of sectors to write */
);
```

Parameters

pdrv

ターゲット・デバイスを識別するための物理ドライブ番号。

buff

書き込まれるバイト配列の最初の項目へのポインタ。書き込まれるデータのサイズはセクタ・サイズ× count バイトです。

sector

開始セクタ番号。32 ビット LBA で指定します。

count

書き込むセクタ数。

Return Values

RES_OK (0)

正常終了。

RES_ERROR

書き込み操作中に回復不能なハードエラーが発生しました。

RES_WRPRT

メディアが書き込み禁止状態です。

RES_PARERR

パラメータが不正です。

RES_NOTRDY

デバイスは初期化されていません。

Properties

ファイル diskio.h にプロトタイプ宣言されています。

Description

引数は BYTE *として定義されているため、指定されたメモリアドレスは必ずしもワード境界にアライメントされているわけではありません。詳細は、disk_read 関数の Description を参照してください。

一般に、複数のセクタ書き込み要求（count > 1）を記録メディアへの単一セクタ転送に分割しないでください。複数の単一セクタ転送に分解した場合、ファイル書き込みスループットが大幅に低下します。

FatFs は、ディスク制御層の遅延書き込み機能を想定しています。進行中の書き込み操作、または、データがライトバックキャッシュへ格納されるのみによって、この関数から戻ってきますが、メディアへの書き込み操作が実際に完了している必要はありません。ただし、この関数から戻った後の buff への書き込みデータは無効です。書き込み完了の確認要求は、disk_ioctl 関数の CTRL_SYNC コマンドによって実行されます。 したがって、遅延書き込み機能を実装する場合、ファイルシステムの書き込みスループットは向上します。

備考：アプリケーション・プログラムはこの関数を呼び出してはいけません。呼び出すとボリューム上の FAT 構造体が壊れる可能性があります。

Example

なし

Special Notes:

FF_FS_READONLY = 1 の場合は使用できません。

3.5 disk_ioctl()

この関数は、一般的なリード/ライト以外のデバイス固有の機能やその他の機能を制御するために呼び出されます。

Format

```
DRESULT disk_ioctl (
  BYTE pdrv,     /* [IN] Drive number */
  BYTE cmd,      /* [IN] Control command code */
  void* buff     /* [I/O] Parameter and data buffer */
);
```

Parameters

pdrv

ターゲット・デバイスを識別するための物理ドライブ番号。

cmd

制御コマンド・コードを設定します。

buff

コマンドコードに依存するパラメータへのポインタ。コマンドがパラメータを持っているかどうかは確認しません。

Return Values

RES_OK (0)

正常終了。

RES_ERROR

何らかのエラーが発生しました。

RES_PARERR

コマンドコードまたはパラメータが不正。

RES_NOTRDY

デバイスは初期化されていません。

Properties

ファイル diskio.h にプロトタイプ宣言されています。

Description

物理ドライブの種類によりサポートされるコマンドは異なりますが、FatFs モジュールでは、次の汎用コマンドのみ使用し、特定のハードウェアやユーザ定義に依存した制御は行いません。

表 3.1 汎用コマンド

コマンド	解説
CTRL_SYNC	デバイスが書き込み処理を完了するのを待ちます。ディスク I / O モジュールまたは記録メディアにライトバックキャッシュがある場合は、ダーティとしてマークされたキャッシュデータをすぐにメディアに書き戻します。メディアへの各書き込み操作が disk_write 関数内で完了した場合、このコマンドには何もしません。
GET_SECTOR_COUNT	ドライブ上の使用可能なセクタ数を、buff が指す DWORD 変数に返します。このコマンドは、作成するボリューム/パーティションのサイズを決定するために f_mkfs および f_fdisk 関数によって使用されます。 FF_USE_MKFS = 1 の場合は必須です。
GET_SECTOR_SIZE	デバイスのセクタ・サイズを buff が指す WORD 変数に返します。このコマンドの有効な戻り値は 512、1024、2048、および 4096 です。このコマンドは、FF_MAX_SS > FF_MIN_SS の場合にのみ必要です。FF_MAX_SS = FF_MIN_SS の場合、このコマンドは使用されず、デバイスはそのセクタ・サイズで動作する必要があります。
GET_BLOCK_SIZE	フラッシュメモリメディアの消去ブロックサイズをセクタ単位で、buff が指す DWORD 変数に返します。許容値は 2 の累乗で 1 から 32768 です。消去ブロックサイズが不明またはフラッシュメモリメディアでない場合は 1 を返します。このコマンドは f_mkfs 関数でのみ使用され、データ領域を消去ブロック境界に揃えようとします。FF_USE_MKFS = 1 の場合は必須です。
CTRL_TRIM	セクタブロックのデータが不要になり、データを消去できることをデバイスに通知します。 セクタブロックは、buff が指す DWORD 配列 {<開始セクタ>, <終了セクタ>} で設定されます。これは ATA デバイスの Trim と同じコマンドです。この機能がサポートされていないかフラッシュメモリデバイスではない場合、このコマンドは何もしません。FatFs は結果コードをチェックせず、セクタブロックがうまく消去されなかったとしてもファイル機能は影響を受けません。このコマンドは、クラスタチェーンの削除時および f_mkfs 関数内で呼び出されます。 FF_USE_TRIM = 1 の場合は必須です。

Example

なし

Special Notes:

FF_FS_READONLY = 1 かつ FF_MAX_SS = FF_MIN_SS の場合、disk_ioctl 関数を使用する必要はありません。

3.6 get_fattime()

この関数は現在時刻を取得します。

Format

```
DWORD get_fattime (void);
```

Return Values

現在のローカル・タイムが DWORD 値にパックされて返されます。ビット・フィールドは以下のとおりです。

bit31:25

1980 年を起点とした年が 0 - 127 で入ります（37 の場合、2017 年）。

bit24:21

月が 1 - 12 の値で入ります。

bit20:16

日が 1 - 31 の値で入ります。

Bit15:11

時が 0 - 23 の値で入ります。

bit10:5

分が 0 - 59 の値で入ります。

bit4:0

秒/2 が 0 - 29 の値で入ります（25 の場合、50 秒）。

Properties

ファイル ff.h にプロトタイプ宣言されています。

Description

システムがリアルタイムクロックをサポートしていなくても、get_fattime 関数は有効な時間を返します。 ゼロが返された場合、ファイルには有効なタイムスタンプがありません。

Example

なし

Special Notes:

FF_FS_READONLY = 1 または FF_FS_NORTC = 1 の場合、この関数は不要です。

3.7 drv_change_alloc()

この関数はドライブ割り当てを変更します。この関数は FatFs とは無関係であり、Renesas 独自の API です。

Format

```
DRESULT drv_change_alloc(TFAT_DRV_NUM tfat_drv, uint8_t dev_type,
                         uint8_t dev_drv_num );
```

Parameters

tfat_drv

TFAT FIT 用の物理ドライブ番号。

dev_type

デバイスのタイプ（TFAT_USB_DRIVE_NUM、TFAT_SDMEM_DRIVE_NUM、または TFAT_USB_MINI_DRIVE_NUM）。

dev_drv_num

デバイス・ドライバ用のドライブ番号／デバイスチャネル。

Return Values

RES_OK (0)

正常終了。

RES_ERROR

tfat_drv で指定した値が無効です。

Properties

ファイル r_tfat_driver_rx_if.h にプロトタイプ宣言されています。

Description

TFAT で使用するドライブは r_tfat_driver_rx_config.h の TFAT_DRIVE_ALLOC_NUM_i 定義によって指定し、ドライブ番号(TFAT)とメモリドライバのドライブ番号が関連付けられます。メモリドライバのドライブ番号は、昇順に自動で割り当てられます。

関連付けを動的に変更したい場合、この関数を使用します。

Example

なし

Special Notes:

なし

4. 内部関数

USB メモリ、SD メモリカード（SD モード）、USB Mini、eMMC、および、Serial Flash memory 向けの関数が用意されています。それぞれの関数内でメモリドライバを呼び出します。

4.1 USB メモリ用

「2.7 コンパイル時の設定」の TFAT_USB_DRIVE_NUM と TFAT_DRIVE_ALLOC_NUM_i(i=0-9) で TFAT_CTRL_USB が設定されている場合に表 4.1.1 の関数が呼び出されます。

表 4.1.1 関数一覧

関数名	機能概要
usb_disk_initialize	ディスク・ドライブの初期化
usb_disk_read	ディスクからの読み込み
usb_disk_write	ディスクへの書き込み
usb_disk_ioctl	その他のドライブ制御
usb_disk_status	ディスク・ドライブの状態取得

表 4.1.2 その他の関数一覧

関数名	機能概要
R_usb_hmsc_WaitLoop	データのリード/ライド完了待ち

4.1.1 usb_disk_initialize

本関数は、ディスク・ドライブの初期化を行います。

Format

```
#include "r_tfat_drv_if_dev.h"
DSTATUS usb_disk_initialize (uint8_t pdrv);
```

Parameters

pdrv	入力	初期化するドライブ番号を指定します。

Return Values

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

USB ドライバの呼び出し制限(起動後、1 度のみ)により、本 API では USB ドライバの初期設定は行っていません。ユーザ側での対応が必要です。

4.1.2 usb_disk_read

本関数は、ディスクからの読み込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT usb_disk_read (
                            uint8_t   pdrv ,
                            uint8_t   * buff ,
                            uint32_t  sector ,
                            uint8_t   count
);
```

Parameters

pdrv	入力	物理的なドライブ番号を指定します。
buff	出力	読み取りデータを格納するバッファを指すポインタ。
sector	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
count	入力	読み取るセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

この関数は、ディスク・ドライブからデータを読み取ります。読み取るデータ位置に関する詳細は引数で指定します。

4.1.3 usb_disk_write

本関数は、ディスクへの書き込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT usb_disk_write (
                          uint8_t   pdrv ,
                          uint8_t   *buff ,
                          uint32_t  sector ,
                          uint8_t   count
);
```

Parameters

pdrv	入力	物理的なドライブ番号を指定します。
buff	入力	書き込むデータを格納するバッファを指すポインタ。
sector	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
count	入力	書き込むセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

この関数は、ディスク・ドライブにデータを書き込みます。書き込むデータに関する詳細は引数で指定します。

4.1.4 usb_disk_ioctl

本関数は、その他のドライブ制御を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT usb_disk_ioctl (
                          uint8_t  pdrv ,
                          uint8_t  cmd ,
                          void    * buff
);
```

Parameters

pdrv	入力	物理的なドライブ番号を指定します。
cmd	入力	コマンド値を指定します。コマンド値は常に 0 になります。
buff	入力	読み取りデータを格納するバッファを指すポインタ。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

usb_disk_ioctl 関数は、すべての TFAT 関数の中で f_sync 関数によってのみ使用されます。 f_sync 関数をアプリケーションで使用しないユーザは、この特定のドライバインタフェース関数の実装をスキップすることができます。

アプリケーションで f_sync 関数を使用する場合はコマンド CTRL_SYNC を実装してください。

f_sync 関数をアプリケーションで使用するユーザは、この特定のドライバインタフェース関数を実装しなければなりません。 このドライバ関数は、保留中の書き込みプロセスを終了するためのコードから構成する必要があります。 ディスク I/O モジュールが書き戻しキャッシュを持つ場合、ダーティセクタは直ちにフラッシュされます。 f_sync 関数は、引数として渡すファイルオブジェクトと関連する保存されていないデータを保存します。

4.1.5 usb_disk_status

本関数は、ディスク・ドライブの状態取得を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS usb_disk_status (uint8_t  pdrv );
```

Parameters

pdrv	入力	物理的なドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

この関数は、ディスクをチェックするコードから構成し、現在のディスクステータスを返します。ディスクステータスは、「2.10 戻り値」に記載するように 3 つの値のいずれかになります。 ディスクステータスは、ディスクステータスと関連するマクロを使用して戻り値を更新することにより、返すことができます。

4.1.6 R_usb_hmsc_WaitLoop

本関数は、データリード/ライドの完了待ちを行います。

Format

```
void  R_usb_hmsc_WaitLoop (void );
```

Parameters

なし

Return Value

なし

Description

処理内容の詳細は、USB ドライバ側のドキュメントをご参照ください。

4.2 SD メモリカード用

「2.7 コンパイル時の設定」の TFAT_SDMEM_DRIVE_NUM と TFAT_DRIVE_ALLOC_NUM_i(i=0-9) で TFAT_CTRL_SDMEM が設定されている場合に表 4.2.1 の関数が呼び出されます。

表 4.2.1 関数一覧

関数名	機能概要
sdmem_disk_initialize	ディスク・ドライブの初期化
sdmem_disk_read	ディスクからの読み込み
sdmem_disk_write	ディスクへの書き込み
sdmem_disk_ioctl	その他のドライブ制御
sdmem_disk_status	ディスク・ドライブの状態取得

[SD メモリカード使用時の注意]

初期設定、マウント処理、VDD 電源電圧供給処理は、本モジュールでは行いません。SD メモリカードモジュールのドキュメントを参考にユーザ側で対応してください。それらの設定を行わないと本モジュールは正常動作しません。

4.2.1 sdmem_disk_initialize

本関数は、ディスク・ドライブの初期化を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS sdmem_disk_initialize (uint8_t drive);
```

Parameters

drive	入力	初期化するドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

本関数では、SD メモリカードドライバの初期設定は行っていません。ユーザ側での対応が必要です。

4.2.2 sdmem_disk_read

本関数は、ディスクからの読み込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT sdmem_disk_read (
                        uint8_t   drive ,
                        uint8_t   *buffer ,
                        uint32_t  sector_number ,
                        uint8_t   sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	出力	読み取りデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	読み取るセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

SD メモリカードからデータを読み出します。ブロック毎に実施します。

4.2.3 sdmem_disk_write

本関数は、ディスクへの書き込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT sdmem_disk_write (
                            uint8_t    drive ,
                            uint8_t    *buffer ,
                            uint32_t   sector_number ,
                            uint8_t    sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	入力	書き込みデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	書き込むセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

SD メモリカードにデータを書き込みます。ブロック毎に実施します。

4.2.4 sdmem_disk_ioctl

本関数は、その他のドライブ制御を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT sdmem_disk_ioctl (
                            uint8_t  drive ,
                            uint8_t  command ,
                            void   *buffer
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
command	入力	コマンド値を指定します。コマンド値は常に 0 になります。
buffer	入力	読み取りデータを格納するバッファを指すポインタ。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

sdmem_disk_ioctl 関数は、すべての TFAT 関数の中で f_sync 関数によってのみ使用されます。 f_sync 関数をアプリケーションで使用しないユーザは、この特定のドライバインタフェース関数の実装をスキップすることができます。

アプリケーションで f_sync 関数を使用する場合はコマンド CTRL_SYNC を実装してください。

f_sync 関数をアプリケーションで使用するユーザは、この特定のドライバインタフェース関数を実装しなければなりません。 このドライバ関数は、保留中の書き込みプロセスを終了するためのコードから構成する必要があります。 ディスク I/O モジュールが書き戻しキャッシュを持つ場合、ダーティセクタは直ちにフラッシュされます。 f_sync 関数は、引数として渡すファイルオブジェクトと関連する保存されていないデータを保存します。

4.2.5 sdmem_disk_status

本関数は、ディスク・ドライブの状態取得を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS sdmem_disk_status (uint8_t  drive );
```

Parameters

drive	入力	物理的なドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

この関数は、ディスクをチェックするコードから構成し、現在のディスクステータスを返します。ディスクステータスは、「2.10 戻り値」に記載するように 3 つの値のいずれかになります。 ディスクステータスは、ディスクステータスと関連するマクロを使用して戻り値を更新することにより、返すことができます。

4.3 USB Mini 用

「2.7 コンパイル時の設定」の TFAT_USB_MINI_DRIVE_NUM と TFAT_DRIVE_ALLOC_NUM_i(i=0-9) で TFAT_CTRL_USB_ MINI が設定されている場合に表 4.3.1 の関数が呼び出されます。

表 4.3.1　関数一覧

関数名	機能概要
usb_mini_disk_initialize	ディスク・ドライブの初期化
usb_mini_disk_read	ディスクからの読み込み
usb_mini_disk_write	ディスクへの書き込み
usb_mini_disk_ioctl	その他のドライブ制御
usb_mini_disk_status	ディスク・ドライブの状態取得

表 4.3.2　その他の関数一覧

関数名	機能概要
R_usb_mini_hmsc_WaitLoop	データのリード/ライド完了待ち

4.3.1 usb_mini_disk_initialize

本関数は、ディスク・ドライブの初期化を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS usb_mini_disk_initialize (uint8_t drive);
```

Parameters

drive	入力	初期化するドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

USB ドライバの呼び出し制限(起動後、1 度のみ)により、本 API では USB ドライバの初期設定は行っていません。ユーザ側での対応が必要です。

4.3.2 usb_mini_disk_read

本関数は、ディスクからの読み込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT usb_disk_read (
                        uint8_t   drive ,
                        uint8_t   *buffer ,
                        uint32_t  sector_number ,
                        uint8_t   sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	出力	読み取りデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	読み取るセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

この関数は、ディスク・ドライブからデータを読み取ります。読み取るデータ位置に関する詳細は引数で指定します。

4.3.3 usb_mini_disk_write

本関数は、ディスクへの書き込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT usb_mini_disk_write (
                                uint8_t    drive ,
                                uint8_t    *buffer ,
                                uint32_t   sector_number ,
                                uint8_t    sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	入力	読み取りデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	読み取るセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

この関数は、ディスク・ドライブにデータを書き込みます。書き込むデータに関する詳細は引数で指定します。

4.3.4 usb_mini_disk_ioctl

本関数は、その他のドライブ制御を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT usb_mini_disk_ioctl (
                                uint8_t  drive ,
                                uint8_t  command ,
                                void    *buffer
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
command	入力	コマンド値を指定します。コマンド値は常に 0 になります。
buffer	入力	読み取りデータを格納するバッファを指すポインタ。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

usb_mini_disk_ioctl 関数は、すべての TFAT 関数の中で f_sync 関数によってのみ使用されます。 f_sync 関数をアプリケーションで使用しないユーザは、この特定のドライバインタフェース関数の実装をスキップすることができます。

アプリケーションで f_sync 関数を使用する場合はコマンド CTRL_SYNC を実装してください。

f_sync 関数をアプリケーションで使用するユーザは、この特定のドライバインタフェース関数を実装しなければなりません。 このドライバ関数は、保留中の書き込みプロセスを終了するためのコードから構成する必要があります。 ディスク I/O モジュールが書き戻しキャッシュを持つ場合、ダーティセクタは直ちにフラッシュされます。 f_sync 関数は、引数として渡すファイルオブジェクトと関連する保存されていないデータを保存します。

4.3.5 usb_mini_disk_status

本関数は、ディスク・ドライブの状態取得を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS usb_mini_disk_status (uint8_t drive);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

この関数は、ディスクをチェックするコードから構成し、現在のディスクステータスを返します。ディスクステータスは、「2.10 戻り値」に記載するように 3 つの値のいずれかになります。 ディスクステータスは、ディスクステータスと関連するマクロを使用して戻り値を更新することにより、返すことができます。

4.3.6 R_usb_mini_hmsc_WaitLoop

本関数は、データリード/ライドの完了待ちを行います。

Format

```
void  R_usb_mini_hmsc_WaitLoop (void );
```

Parameters

なし

Return Value

なし

Description

処理内容の詳細は、USB ドライバ側のドキュメントをご参照ください。

4.4 eMMC 用

「2.7 コンパイル時の設定」の TFAT_MMC_DRIVE_NUM と TFAT_DRIVE_ALLOC_NUM_i(i=0-9) で TFAT_CTRL_MMC が設定されている場合に表 4.4.1 の関数が呼び出されます。

表 4.4.1 関数一覧

関数名	機能概要
mmcif_disk_initialize	ディスク・ドライブの初期化
mmcif_disk_read	ディスクからの読み込み
mmcif_disk_write	ディスクへの書き込み
mmcif_disk_ioctl	その他のドライブ制御
mmcif_disk_status	ディスク・ドライブの状態取得

4.4.1 mmcif_disk_initialize

本関数は、ディスク・ドライブの初期化を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS mmcif_disk_initialize (uint8_t drive);
```

Parameters

drive	入力	初期化するドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

本関数では、eMMC ドライバの初期設定は行っていません。ユーザ側での対応が必要です。

4.4.2 mmcif_disk_read

本関数は、ディスクからの読み込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT mmcif_disk_read (
                        uint8_t   drive ,
                        uint8_t   *buffer ,
                        uint32_t  sector_number ,
                        uint8_t   sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	出力	読み取りデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	読み取るセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

eMMC からデータを読み出します。ブロック毎に実施します。

4.4.3 mmcif_disk_write

本関数は、ディスクへの書き込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT mmcif_disk_write (
                            uint8_t   drive ,
                            uint8_t   *buffer ,
                            uint32_t  sector_number ,
                            uint8_t   sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	入力	書き込みデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	書き込むセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

eMMC にデータを書き込みます。ブロック毎に実施します。

4.4.4 mmcif_disk_ioctl

本関数は、その他のドライブ制御を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT mmcif_disk_ioctl (
                            uint8_t  drive ,
                            uint8_t  command ,
                            void   *buffer
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
command	入力	コマンド値を指定します。指定できるコマンドは以下です。 ・CTRL_SYNC ・GET_SECTOR_COUNT ・GET_SECTOR_SIZE ・GET_BLOCK_SIZE ・CTRL_TRIM
buffer	入力	読み取りデータを格納するバッファを指すポインタ。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

mmcif_disk_ioctl 関数は、すべての TFAT 関数の中で f_sync または f_mkfs 関数によってのみ使用されます。これらの関数をアプリケーションで使用しないユーザは、この特定のドライバインタフェース関数の実装をスキップすることができます。

アプリケーションで f_sync 関数を使用する場合はコマンド CTRL_SYNC を実装してください。

コマンド CTRL_SYNC は、保留中の書き込みプロセスを終了するためのコードから構成する必要があります。 ディスク I/O モジュールが書き戻しキャッシュを持つ場合、ダーティセクタは直ちにフラッシュされます。 f_sync 関数は、引数として渡すファイルオブジェクトと関連する保存されていないデータを保存します。

その他のコマンドについては、表 3.1　汎用コマンド をご覧ください。

4.4.5 mmcif_disk_status

本関数は、ディスク・ドライブの状態取得を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS mmcif_disk_status (uint8_t  drive );
```

Parameters

drive	入力	物理的なドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

この関数は、ディスクをチェックするコードから構成し、現在のディスクステータスを返します。ディスクステータスは、「2.10 戻り値」に記載するように 3 つの値のいずれかになります。 ディスクステータスは、ディスクステータスと関連するマクロを使用して戻り値を更新することにより、返すことができます。

4.5 Serial Flash Memory 用

「2.7 コンパイル時の設定」の TFAT_SERIAL_FLASH_DRIVE_NUM と TFAT_DRIVE_ALLOC_NUM_i(i=0-9) で TFAT_CTRL_SERIAL_FLASH が設定されている場合に表 4.5.1 の関数が呼び出されます。

表 4.5.1　関数一覧

関数名	機能概要
flash_spi_disk_initialize	ディスク・ドライブの初期化
flash_spi_disk_read	ディスクからの読み込み
flash_spi_disk_write	ディスクへの書き込み
flash_spi_disk_ioctl	その他のドライブ制御
flash_spi_disk_status	ディスク・ドライブの状態取得

表 4.5.2　その他の関数一覧

関数名	機能概要
flash_spi_1ms_interval	1 ms ごとに内部タイマ更新

4.5.1 flash_spi_disk_initialize

本関数は、ディスク・ドライブの初期化を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS flash_spi_disk_initialize (uint8_t drive);
```

Parameters

drive	入力	初期化するドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

本関数では、Serial Flash memory ドライバの初期設定は行っていません。ユーザ側での対応が必要です。

本関数が呼び出されたタイミングで Serial Flash memory がビジー状態またはエラー状態の場合、返り値はステータス STA_NOINIT です。

4.5.2 flash_spi_disk_read

本関数は、ディスクからの読み込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT flash_spi_disk_read (
                        uint8_t   drive ,
                        uint8_t   *buffer ,
                        uint32_t  sector_number ,
                        uint8_t   sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	出力	読み取りデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	読み取るセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

Serial Flash memory からデータを読み出します。ブロック毎に実施します。

4.5.3 flash_spi_disk_write

本関数は、ディスクへの書き込みを行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT flash_spi_disk_write (
                            uint8_t    drive ,
                            uint8_t    *buffer ,
                            uint32_t   sector_number ,
                            uint8_t    sector_count
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
buffer	入力	書き込みデータを格納するバッファを指すポインタ。
sector_number	入力	開始セクタ番号を論理ブロックアドレス（LBA）で指定します。
sector_count	入力	書き込むセクタ数を指定します。値は 1～255 の範囲です。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

Serial Flash memory にデータを書き込みます。ブロック毎に実施します。

4.5.4 flash_spi_disk_ioctl

本関数は、その他のドライブ制御を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DRESULT flash_spi_disk_ioctl (
                            uint8_t  drive ,
                            uint8_t  command ,
                            void   *buffer
);
```

Parameters

drive	入力	物理的なドライブ番号を指定します。
command	入力	コマンド値を指定します。指定できるコマンドは以下です。 ・CTRL_SYNC ・GET_SECTOR_COUNT ・GET_SECTOR_SIZE ・GET_BLOCK_SIZE ・CTRL_TRIM
buffer	入力	読み取りデータを格納するバッファを指すポインタ。

Return Value

DRESULT	「2.10 戻り値」に記載した関数実行の結果

Description

flash_spi_disk_ioctl 関数は、すべての TFAT 関数の中で f_sync または f_mkfs 関数によってのみ使用されます。これらの関数をアプリケーションで使用しないユーザは、この特定のドライバインタフェース関数の実装をスキップすることができます。

アプリケーションで f_sync 関数を使用する場合はコマンド CTRL_SYNC を実装してください。

コマンド CTRL_SYNC は、保留中の書き込みプロセスを終了するためのコードから構成する必要があります。ディスク I/O モジュールが書き戻しキャッシュを持つ場合、ダーティセクタは直ちにフラッシュされます。f_sync 関数は、引数として渡すファイルオブジェクトと関連する保存されていないデータを保存します。

その他のコマンドについては、表 3.1　汎用コマンド をご覧ください。

4.5.5 flash_spi_disk_status

本関数は、ディスク・ドライブの状態取得を行います。

Format

```
#include " r_tfat_drv_if_dev.h "
DSTATUS flash_spi_disk_status (uint8_t  drive );
```

Parameters

drive	入力	物理的なドライブ番号を指定します。

Return Value

RES_OK	正常終了
RES_OK 以外	「2.10 戻り値」に記載した関数実行後のディスクステータス

Description

この関数は、ディスクをチェックするコードから構成し、現在のディスクステータスを返します。ディスクステータスは、「2.10 戻り値」に記載するように 3 つの値のいずれかになります。 ディスクステータスは、ディスクステータスと関連するマクロを使用して戻り値を更新することにより、返すことができます。

4.5.6 flash_spi_1ms_interval

本関数は、1 ms ごとに内部タイマを更新します。

Format

```
#include " r_tfat_drv_if_dev.h "
void flash_spi_1ms_interval (void);
```

Parameters

なし

Return Value

なし

Description

この関数は、1 ms ごとに TFAT driver FIT 内部のタイマを更新します。このタイマは Serial Flash memory に対してビジー状態の確認のために使用されます。

5. 端子設定

TFAT driver FIT に端子設定はありません。

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.1.05。SD メモリカードドライバ用, USB Mini ドライバ用)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V6.3.0
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 V2.08.00
	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
モジュールのリビジョン	Rev.1.05
使用ボード	Renesas Starter Kit+ for RX64M (型名：R0K50564MSxxxxx) Renesas Starter Kit for RX231 (型名：R0K505231Sxxxxx)
RTOS	なし

表 6.2 動作確認環境 (Rev.1.05。USB ドライバ用)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 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 V3.01.00
	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
モジュールのリビジョン	Rev.1.05
使用ボード	Renesas Starter Kit+ for RX64M (型名：R0K50564MSxxxxx)
RTOS	なし

表 6.3　動作確認環境 (Rev.2.00)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.7.0 IAR Embedded Workbench for Renesas RX 4.13.1
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 V3.02.00 コンパイルオプション：統合開発環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 8.3.0.201904 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 4.13.1 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	Rev.2.00
使用ボード	Renesas Starter Kit+ for RX72M (型名：RTK5572Mxxxxxxxxxx)
RTOS	FreeRTOS V10.0.00 RI600V4 V1.06.00

表 6.4　動作確認環境 (Rev.2.10)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio V7.8.0 IAR Embedded Workbench for Renesas RX 4.14.1
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 V3.02.00 コンパイルオプション：統合開発環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 8.3.0.201904 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 4.14.1 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	Rev.2.10
使用ボード	Renesas Starter Kit for RX231 (型名：RTK55231xxxxxxxxxx) Renesas Starter Kit+ for RX64M (型名：RTK5564Mxxxxxxxxxx)
RTOS	FreeRTOS V10.0.00 RI600V4 V1.06.00

表 6.5　動作確認環境 (Rev.2.20)

項目	内容
統合開発環境	ルネサスエレクトロニクス製 e^2 studio 2020-07 IAR Embedded Workbench for Renesas RX 4.14.1
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 V3.02.00 コンパイルオプション：統合開発環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 8.3.0.202002 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 4.14.1 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	Rev.2.20
使用ボード	Renesas Starter Kit for RX231 (型名：RTK55231xxxxxxxxxx) Renesas Starter Kit+ for RX72N (型名：RTK5572Nxxxxxxxxxx) Renesas Starter Kit+ for RX72M (型名：RTK5572Mxxxxxxxxxx)
RTOS	FreeRTOS V10.0.03 RI600V4 V1.06.00

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)」
- e^2 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 e^2 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_tafat_rx_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。

(3) Q：本 FIT モジュールをプロジェクトに追加しましたが、ビルド実行するとコンフィグ設定が間違っている旨のエラーが発生します。

A：“r_tfat_driver_rx_config.h”ファイルの設定値が間違っている可能性があります。“r_tfat_driver_rx_config.h”ファイルを確認して正しい値を設定してください。詳細は「2.7 コンパイル時の設定」を参照してください。

(4) Q：端子設定していない場合発生する現象が発生します。

A：正しく端子設定が行われていない可能性があります。本 FIT モジュールを使用する場合は端子設定が必要です。詳細は「5 端子設定」を参照してください。

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
RX ファミリ CC-RX コンパイラ ユーザーズマニュアル（R20UT3248）
（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の対応について

- テクニカルアップデートはありません。

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2014.12.01	-	初版発行
1.01	2015.01.05	-	サポートマイコン追加
1.02	2015.06.30	-	サポートマイコン(RX231)追加
1.03	2016.10.01	-	サポート(RX ファミリ)追加
1.04	2018.06.29	-	1.2.2 図 1-1 呼び出しモジュールに System timer、CMT 追加 1.3 API 概要 追加 2.6 SD メモリカード定義名変更 2.7 コードサイズ 追加 3.6 get_fattime() 説明修正 4.2.1-4.2.5 API 名変更 5 付録 追加 6 参考ドキュメント 追加
1.05	2018.12.14	-	USB ドライバの RTOS サポートによるリビジョンアップ
2.00	2020.02.25	-	以下のコンパイラに対応 ・GCC for Renesas RX ・IAR C/C++ Compiler for Renesas RX 以下の RTOS に対応 ・FreeRTOS ・RI600V4 関数名から「R_TFAT_」を削除しました。
2.10	2020.07.27	-	・以下の記憶メディアに対応 ・eMMC ・Serial Flash memory ・以下の記憶メディアに対しフォーマット機能（f_mkfs）を追加 ・eMMC ・Serial Flash memory ・セクタ・サイズ 4096 バイトに対応 ・USB mini FIT を使用するデバイスと以下の RTOS の組み合わせに対応 ・FreeRTOS ・RI600V4
2.20	2020.09.10	-	・以下の記憶メディアに対しフォーマット機能（f_mkfs）を追加 ・SD メモリカード ・USB

Rev.	発行日	改訂内容	
		ページ	ポイント
2.20	2020.09.10	プログラム	ソフトウェア不具合のため、TFAT driver FIT モジュールを改修 ■内容 eMMC の GET_BLOCK_SIZE コマンドで取得した BLOCK SIZE が不正値である。 ■発生条件 次の 2 つの条件に該当したとき ・TFAT driver FIT モジュール Rev.2.10 のバージョンをご使用されている ・eMMC をフォーマットする ■対策 TFAT driver FIT モジュール Rev2.20 以降をご使用ください。
		プログラム	ソフトウェア不具合のため、TFAT driver FIT モジュールを改修 ■内容 Serial Flash memory の GET_BLOCK_SIZE コマンドで取得した BLOCK SIZE が不正値である。 Serial Flash memory の size が非常に大きい場合、フォーマット処理の時間が長くなり、使用可能なメモリも小さくなってしまう現象が発生する。 ■発生条件 次の 2 つの条件に該当したとき ・TFAT driver FIT モジュール Rev.2.10 のバージョンをご使用されている ・Serial Flash memory をフォーマットする ■対策 TFAT driver FIT モジュール Rev2.20 以降をご使用ください。

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。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)

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。