RXファミリ

R01AN2026JJ0142
Rev.1.42
Sep 30, 2023

USB Host Mass Storage Class Driver (HMSC) Firmware Integration Technology

要旨

本アプリケーションノートでは、Firmware Integration Technology(FIT)を使用した、USB Host マスストレージクラスドライバ(HMSC)について説明します。本モジュールは USB Basic Host and Peripheral Driver(USB-BASIC-FW FIT モジュール)と組み合わせることで動作します。以降、本モジュールを USB HMSC FIT モジュールと称します。

対象デバイス

RX65N/RX651 グループ
RX64M グループ
RX71M グループ
RX66T グループ
RX72T グループ
RX72M グループ
RX66N グループ
RX72N グループ
RX671 グループ

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あわせて変更し、十分評価してください。

関連ドキュメント

1. Universal Serial Bus Revision 2.0 specification
2. USB Mass Storage Class Specification Overview Revision 1.1
3. USB Mass Storage Class Bulk-Only Transport Revision 1.0
   【http://www.usb.org/developers/docs/】
4. RX64M グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0377)
5. RX71M グループユーザーズマニュアル ハードウェア編 (ドキュメント No.R01UH0493)
6. RX65N/RX651 グループユーザーズマニュアル ハードウェア編 (ドキュメント No. R01UH0590)
7. RX65N/RX651-2M グループユーザーズマニュアル ハードウェア編 (ドキュメント No. R01UH0659)
8. RX66T グループユーザーズマニュアル ハードウェア編（ドキュメント No. R01UH0749)
9. RX72T グループユーザーズマニュアル ハードウェア編（ドキュメント No. R01UH0803)
10. RX72M グループユーザーズマニュアル ハードウェア編（ドキュメント No. R01UH0804)
11. RX66N グループユーザーズマニュアル ハードウェア編 (ドキュメント No. R01UH0825)
12. RX72N グループユーザーズマニュアル ハードウェア編 (ドキュメント No. R01UH0824)
13. RX671 グループユーザーズマニュアル ハードウェア編 (ドキュメント No. R01UH0899)
14. M3S-TFAT-Tiny FAT ファイルシステムソフトウェア (ドキュメント No. R20AN0038)
15. M3S-TFAT-Tiny メモリドライバインタフェースモジュール(ドキュメント No. R20AN0335)
16. USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート
    (ドキュメント No. R01AN2025)

— ルネサス エレクトロニクスホームページ
【http://japan.renesas.com/】
— USB デバイスページ
【http://japan.renesas.com/prod/usb/】

目次

1. 概要

USB HMSC FIT モジュールは、USB-BASIC-FW FIT モジュールと組み合わせることで、USB Host マスストレージクラスドライバ（以降 HMSC と記述）として動作します。

HMSC は､USB マスストレージクラスの Bulk-Only Transport (BOT) プロトコルで構築されています｡ファイルシステム、ストレージデバイスドライバと組み合わせることで BOT 対応の USB ストレージ機器と通信を行うことが可能です。

なお、本ドライバを使用する場合、M3S-TFAT-Tiny FAT ファイルシステムソフトウェア(Document No. R20AN0038)および M3S-TFAT-Tiny メモリドライバインタフェースモジュール (Document No. R20AN0335)を組み合わせてご使用ください。

以下に、本モジュールがサポートしている機能を示します。

1. 接続された USB ストレージ機器のデバイス照合（動作可否判定）を行う。
2. BOT プロトコルによるストレージコマンド通信を行う。
3. USB マスストレージサブクラスの SFF-8070i(ATAPI)に対応。
4. データ転送に使用するパイプを、IN/OUT 転送で共有。複数のデバイスを接続した場合も 1 本のパイプを共有。
5. Non-OS, FreeRTOS 版では、最大 4 つのストレージ機器を接続することができます。

1.1 必ずお読みください

このドライバを使ってアプリケーションプログラムを作成する場合は、USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート(ドキュメント No.R01AN2025)を参照いただきますようお願いします。このアプリケーションノートは、パッケージ内の**"reference_documents"**フォルダにあります。

1.2 注意事項

1. 本ドライバは、USB 通信動作を保証するものではありません。システムに適用される場合は、お客様における動作検証はもとより、多種多様なデバイスに対する接続確認を実施してください。
2. Azure RTOS を使用する場合、USBX ドライバが使用されるため、本モジュールを使用する必要はありません。
3. 本ドライバは以下の FAT と組み合わせて動作確認を行っています。
   (1). RX Family Open Source FAT File System [M3S-TFAT-Tiny] Module Firmware Integration Technology Rev.3.03 (Non-OS,FreeRTOS,uITRON 使用時)
   (2). Azure RTOS FileX 6.1.12
4. Azure RTOS API、USBX API や FileX API については、Azure RTOS、USBX および FileX のドキュメントを参照ください。

1.3 制限事項

1. ストレージ機器として認識できない MSC デバイスがあります。
2. 本ドライバは、GetMaxLun コマンド(Mass Storage Class コマンド)に対する応答が"1"以上の MSC デバイスをサポートしておりません。

3. セクタサイズが 512 バイトの USB ストレージ機器と接続可能です。
4. READ_CAPACITY コマンドに対し STALL 応答する MSC デバイスはセクタサイズを 512 バイトとして動作します。
5. 本ドライバは MFi 認証済の USB メモリはサポートしておりません。
6. Azure RTOS 版の接続可能な MSC デバイス数は 1 デバイスです。

1.4 用語一覧

APL	:	Application program
BOT	:	Mass Storage class Bulk Only Transport
FSL	:	FAT File System Library
HCD	:	Host Control Driver for USB-BASIC-FW
HDCD	:	Host Device Class Driver (Device driver and USB class driver)
MGR	:	Peripheral Device State Manager for HCD
MSC	:	Mass Storage Class
Non-OS	:	USB Driver for OS-less
RSK	:	Renesas Starter Kits
RTOS	:	USB Driver for FreeRTOS and uITRON.
TFAT	:	Tiny FAT file System Software for microcontrollers (M3S-TFAT-Tiny-RX)
USB-BASIC-FW	:	USB Basic Host and Peripheral Driver

1.5 USB HMSC FIT モジュール

本モジュールは、r_usb_basic を使用したプロジェクトに組み込む必要があります。プロジェクトに組み込み後、API を使用することで USB の H/W 制御を行います。

2. ソフトウェア構成 (FreeRTOS、Non-OS の場合)

HDCD(ホストデバイスクラスドライバ)は HMSDD（ホストマスストレージデバイスドライバ）と HMSCD（USB ホストマスストレージクラスドライバ）の総称です。
Figure 2-1に HMSC のモジュール構成、Table 2-1にモジュール機能概要を示します。

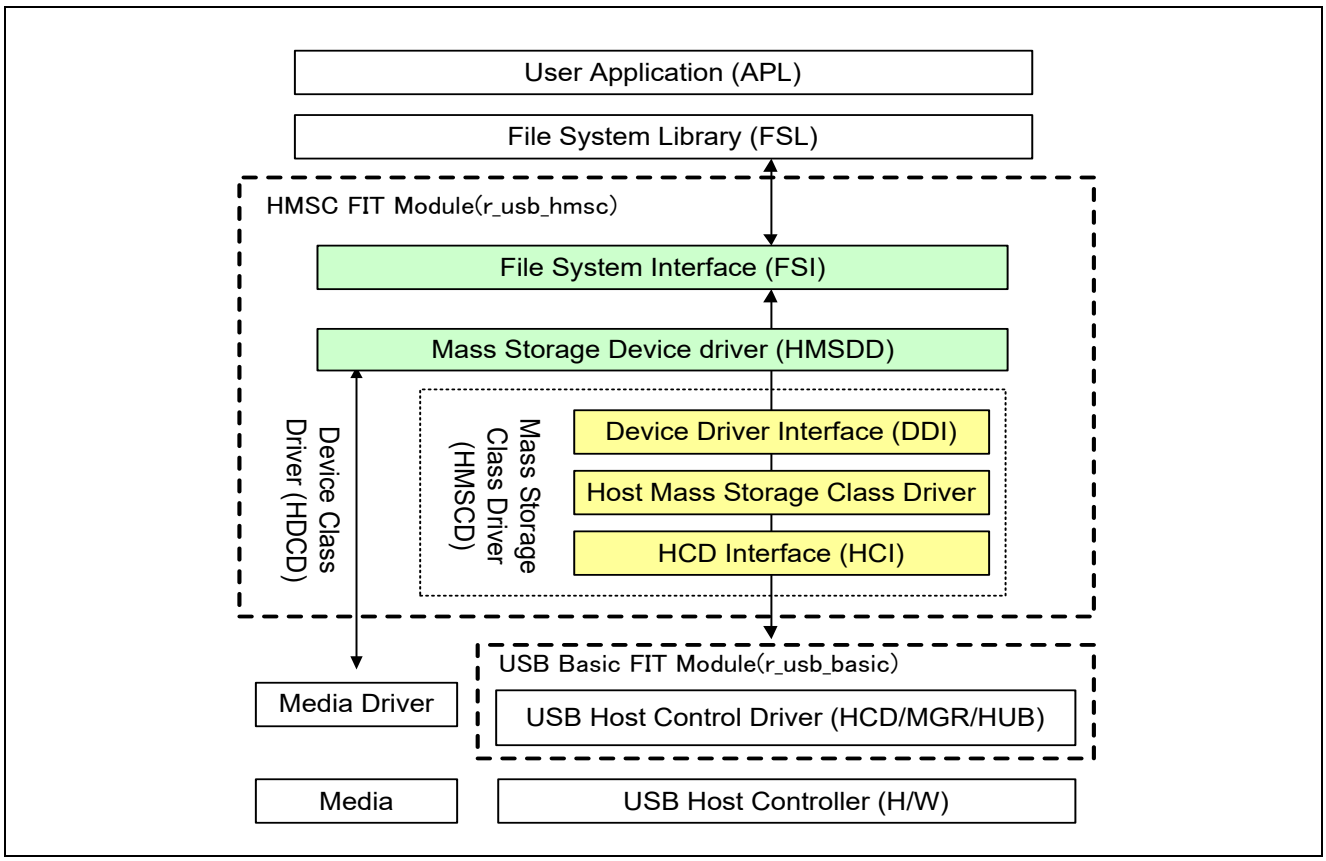

Figure 2-1 ソフトウェア構成図

Table 2-1 モジュール機能概要

モジュール名	機能概要
FSI	FSL－HMSDD 間のインタフェース関数 FSL にあわせて改変してください。
HMSDD	マスストレージデバイスドライバ
DDI	HMSDD－HMSCD 間のインタフェース関数 HMSDD に合わせて改変してください。
HMSCD	マスストレージクラスドライバ ストレージコマンドに BOT プロトコルを付加して HCD へ要求します。また、BOT のシーケンスを管理します。 システム仕様にあわせてストレージコマンドを追加（改変）してください。
HCI	HMSCD－HCD 間のインタフェース関数です。
MGR / HUB	接続されたデバイスとエニュメレーションをして HMSCD を起動します。またデバイスの状態管理も行います。
HCD	USB Host H/W 制御ドライバ

3. API 情報

本ドライバの API はルネサスの API の命名基準に従っています。

3.1 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。

- USB

3.2 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_usb_basic

3.3 動作確認環境

このドライバの動作確認環境を以下に示します。

Table 3-1 動作確認環境

項目	内容
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.3.03.00 (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 version 4.12.01
リアルタイム OS	FreeRTOS V.10.0.0 RI600V4 Azure RTOS (USBX) 6.1.12
エンディアン	リトルエンディアン / ビッグエンディアン
モジュールのリビジョン	Rev.1.42
使用ボード	Renesas Starter Kits for RX64M Renesas Starter Kits for RX71M Renesas Starter Kits for RX65N, Renesas Starter Kits for RX65N-2MB Renesas Starter Kits for RX72T Renesas Starter Kits for RX72M Renesas Starter Kits for RX72N Renesas Starter Kits for RX671

3.4 使用する割り込みベクタ

このドライバが使用する割り込みベクタを以下に示します。

Table 3-2 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 RX71M	USBI0 割り込み(ベクタ番号: 189, 割り込み要因番号：62, 選択型割り込み B) USB D0FIFO0 割り込み(ベクタ番号: 34) / USB D1FIFO0 割り込み(ベクタ番号: 35) USBR0 割り込み(ベクタ番号:90)

	USBAR 割り込み(ベクタ番号: 94) USB D0FIFO2 割り込み(ベクタ番号: 32) / USB D1FIFO2 割り込み(ベクタ番号: 33)
RX65N RX651 RX72M RX72N RX66N	USBI0 割り込み(ベクタ番号: 185, 割り込み要因番号 : 62, 選択型割り込み B) USB D0FIFO0 割り込み(ベクタ番号: 34) / USB D1FIFO0 割り込み(ベクタ番号: 35) USBR0 割り込み(ベクタ番号:90)
RX66T RX72T	USBI0 割り込み(ベクタ番号: 174) / USBR0 割り込み(ベクタ番号: 90) USB D0FIFO0 割り込み(ベクタ番号: 34) / USB D1FIFO0 割り込み(ベクタ番号: 35)
RX671	USBI0 割り込み(ベクタ番号: 185, 割り込み要因番号 : 62, 選択型割り込み B) USB D0FIFO0 割り込み(ベクタ番号: 34) / USB D1FIFO0 割り込み(ベクタ番号: 35) USBR0 割り込み(ベクタ番号:90)
	USBI1 割り込み(ベクタ番号: 182, 割り込み要因番号 : 63, 選択型割り込み B) USB D0FIFO1 割り込み(ベクタ番号: 36) / USB D1FIFO1 割り込み(ベクタ番号: 37)

3.5 ヘッダファイル

すべての API 呼び出しとそれをサポートするインタフェース定義は r_usb_basic_if.h と r_usb_hmsc_if.h に記載されています。

3.6 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。

3.7 コンパイル時の設定

コンパイル時の設定については、USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート(ドキュメント No. R01AN2025)の「コンフィグレーション」章を参照してください。

3.8 ROM / RAM サイズ

本ドライバの ROM/RAM サイズを以下に示します。

1. CC-RX (最適化レベル: Default)

(1). Non-OS

	引数チェック実施時	引数チェック非実施時
ROM サイズ	44.5K バイト (Note 4)	44.0K バイト (Note 5)
RAM サイズ	23.9K バイト	23.9K バイト

(2). RTOS

a. FreeRTOS

	引数チェック実施時	引数チェック非実施時
ROM サイズ	54.2K バイト (Note 4)	53.7K バイト (Note 5)
RAM サイズ	51.2K バイト	51.2K バイト

b. RI600V4

	引数チェック実施時	引数チェック非実施時
ROM サイズ	56.4K バイト (Note 4)	55.9K バイト (Note 5)

RAM サイズ	28.3K バイト	28.3K バイト

c. Azure RTOS (Note 7)

ROM サイズ	106.3K バイト
RAM サイズ	17.7K バイト

2. GCC (最適化レベル: -O2)

(1). Non-OS

	引数チェック実施時	引数チェック非実施時
ROM サイズ	51.2K バイト (Note 4)	50.6K バイト (Note 5)
RAM サイズ	23.7K バイト	23.7K バイト

(2). Azure RTOS (Note 7)

ROM サイズ	122.8K バイト
RAM サイズ	16.1K バイト

3. IAR (最適化レベル: Medium)

(1). Non-OS

	引数チェック実施時	引数チェック非実施時
ROM サイズ	45.5K バイト (Note 4)	44.9K バイト (Note 5)
RAM サイズ	22.4K バイト	22.4K バイト

(2). Azure RTOS (Note 7)

ROM サイズ	92.8K バイト
RAM サイズ	15.5K バイト

[Note]

1. 上記のサイズには、BSP および USB Basic Driver の ROM/RAM サイズが含まれています。
2. 上記のサイズには、TFAT の ROM/RAM サイズは含まれていません。
3. 上記は V2 コアオプション指定時のサイズです。
4. 「引数チェック実施時」の ROM サイズは、r_usb_basic_config.h ファイル内の USB_CFG_PARAM_CHECKING 定義に対し USB_CFG_ENABLE を指定した時の値です。
5. 「引数チェック非実施時」の ROM サイズは、r_usb_basic_config.h ファイル内の USB_CFG_PARAM_CHECKING 定義に対し USB_CFG_DISABLE を指定した時の値です。
6. RTOS には、リアルタイム OS の ROM/RAM サイズが含まれています。
7. Azure RTOS には Azure RTOS, USBX および FileX の ROM/RAM サイズが含まれています。

3.9 引数

API 関数の引数に使用される構造体については、USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート(ドキュメント No.R01AN2025)内の「構造体」の章を参照してください。

3.10 for 文、while 文、do while 文について

FIT 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。

3.11 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。

4. ターゲットペリフェラルリスト（TPL）

TPLについては、USB Basic Host and Peripheral Driver Firmware Integration Technologyアプリケーションノート(ドキュメントNo.R01AN2025)内の「ターゲットペリフェラルリスト(TPL)の設定方法」の章を参照してください。

5. クラスドライバ概要

5.1 クラスリクエスト

本ドライバがサポートしているクラスリクエストを以下に示します。

Table 5-1 クラスリクエスト

リクエスト	説明
GetMaxLun	サポートする最大ユニット数を取得する。
MassStrageReset	プロトコルエラーを解除する。

5.2 ストレージコマンド

本ドライバは、以下のストレージコマンドをサポートしています。

1. TEST_UNIT_READY
2. REQUEST_SENSE
3. MODE_SELECT10
4. MODE_SENSE10
5. PREVENT_ALLOW
6. READ_FORMAT_CAPACITY
7. READ10
8. WRITE10

6. API

Host Mass Storage Class 固有の API を以下に示します。Azure RTOS を使用する場合、以下の API を使用しないでください。

API	説明
R_USB_HmscStrgCmd()	Host Mass Storage コマンドを発行
R_USB_HmscGetDriveNo()	ドライブ番号を取得
R_USB_HmscGetSem()	セマフォを取得(RTOS のみ)
R_USB_HmscRelSem()	セマフォを解放(RTOS のみ)

[Note]

1. ストレージメディアに対するアクセスを行う場合は、FAT(File Allocation Table)の API を使用してください。
2. その他の API を使用する場合は USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート(ドキュメント No.R01AN2025)内の「API」の章を参照してください。

6.1 R_USB_HmscStrgCmd

Mass Storage コマンドを発行する。

形式

usb_err_t R_USB_HmscStrgCmd(usb_ctrl_t *p_ctrl, uint8_t *p_buf, uint16_t command)

引数

p_ctrl	usb_ctrl_t 構造体領域へのポインタ
p_buf	バッファ領域へのポインタ
command	Mass Storage コマンド

戻り値

USB_SUCCESS	正常終了
USB_ERR_PARA	パラメータエラー
USB_ERR_NG	その他のエラー

解説

1. Non-OS の場合

引数(p_ctrl)のメンバ module とメンバ address によって指定された MSC デバイスに対し、引数 command に指定された Mass Storage コマンドを発行します。アプリケーションプログラムは、R_USB_GetEvent 関数の戻り値(USB_STS_MSC_CMD_COMPLETE)により Mass Storage コマンドの完了を確認することができます。

応答データがある Mass Storage コマンドを発行した場合、アプリケーションプログラムでは、R_USB_GetEvent 関数の戻り値(USB_STS_MSC_CMD_COMPLETE)を確認後、第 2 引数(p_buf)が示す領域から応答データを取得することができます。受信した応答データのサイズは、第 1 引数(p_ctrl)のメンバ size を参照してください。

2. RTOS の場合

引数(p_ctrl)のメンバ module とメンバ address によって指定された MSC デバイスに対し、引数 command に指定された Mass Storage コマンドを発行します。アプリケーションプログラムは MassStorage コマンドの完了をコールバック関数の引数(usb_ctrl_t 構造体:メンバ event)を参照することにより確認できます。Mass Storage コマンド完了時、この引数(usb_ctrl_t 構造体:メンバ event)には USB_STS_MSC_CMD_COMPLETE が設定されます。

応答データがある Mass Storage コマンドを発行した場合、アプリケーションプログラムでは、コールバック関数の引数(usb_ctrl_t 構造体:メンバ event) に USB_STS_MSC_CMD_COMPLETE が設定されていることを確認した後、第 2 引数(p_buf)が示す領域から応答データを取得することができます。受信した応答データのサイズは、第 1 引数(p_ctrl)のメンバ size を参照してください。

Non-OS と RTOS ともに引数 command には、以下を指定してください。

Table 6-1 MassStorage コマンド

MassStorage コマンド
USB_ATAPI_TEST_UNIT_READY
USB_ATAPI_REQUEST_SENSE
USB_ATAPI_INQUIRY
USB_ATAPI_MODE_SELECT10
USB_ATAPI_PREVENT_ALLOW
USB_ATAPI_READ_FORMAT_CAPACITY
USB_ATAPI_READ_CAPACITY
USB_ATAPI_MODE_SENSE10

補足

1. 本 API をコールする前に usb_ctrl_t 構造体のメンバ module に USB モジュール番号を指定し、メンバ address に対しデバイスアドレスを指定してください。なお、メンバ module に対し USB_IP0/USB_IP1 以外を指定した場合、戻り値に USB_ERR_PARA が返されます。
2. ご使用の MCU が USB モジュールを 1 つしかサポートしていない場合、メンバ module に対し USB_IP1 を指定しないでください。USB_IP1 を指定した場合は、戻り値に USB_ERR_PARA が返されます。
3. 引数 p_ctrl に対し USB_NULL を指定した場合は、戻り値に USB_ERR_PARA が返されます。
4. 引数 p_buf には自動変数(スタック)領域へのポインタは指定しないでください。
5. MSC デバイスからの応答データがないコマンドを発行する場合、引数 p_buf に対し USB_NULL を指定してください。
6. 引数 command に対しTable 6-1の MassStorage コマンド以外のコマンドを指定した場合、戻り値に USB_ERR_PARA が返されます。
7. 本 API による Mass Storage コマンド発行後、FAT API や本 API をコールする場合、必ず、R_USB_GetEvent 関数の戻り値(USB_STS_CMD_COMPLETE)を確認した後でこれらの API をコールしてください。
8. CSW については、「7. Mass Storageコマンドの戻り値 (USB_STS_MSC_CMD_COMPLETE)」を参照してください。
9. CSW 情報は、usb_ctrl_t 構造体のメンバ status に設定されます。メンバ status の値が USB_CSW_FAIL の場合、本 API を使って"Request Sense"コマンドを MSC デバイスに発行してください。
10. Mode Sense10 コマンドのページコード(1 バイト)は、第 2 引数(p_buf)が示す領域の先頭アドレスに指定してください。
11. Mode Select10 コマンドのパラメータデータは USB マスストレージサブクラス(SFF-8070i など)の規格書をもとに第 2 引数(p_buf)が示す領域に指定してください。
12. USB デバイスが CONFIGURED 状態の場合に、本 API をコールすることができます。CONFIGURED 以外の状態で本 API をコールすると戻り値に USB_ERR_NG が返されます。

使用例

1. Non-OS

```
uint8_t g_buf[64];
void    usb_application( void )
{
    usb_ctrl_t ctrl;
    usb_err_t err;
                    :
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
        {
                    :
            case USB_STS_CONFIGURED:
                    :
                g_buf[0] = 0x3F             /* Page Code */
                ctrl.module = USB_IP1;
                ctrl.address = adr;
                R_USB_HmscStrgCmd(&ctrl, &g_buf, USB_ATAPI_MODE_SENSE10);
                    :
            break;
            case USB_STS_MSC_CMD_COMPLETE:
                if (ctrl.status == USB_CSW_FAIL)
                {
                    R_USB_HmscStrgCmd(&ctrl, &g_buf, USB_ATAPI_REQUEST_SENSE);
                }
                    :
            break;
                    :
        }
    }
}
```

2. RTOS

```
/* Callback function */
void usb_apl_callback (usb_ctrl_t *p_ctr, rtos_task_id_t task_id, uint8_t is_request)
{
      USB_APL_SND_MSG(USB_APL_MBX, (usb_msg_t *)p_ctrl);
}

void          usb_application_task( void )
{
    usb_ctrl_t        ctrl;
    usb_ctrl_t        *p_mess;
              :
    while(1)
    {
          USB_APL_RCV_MSG(USB_APL_MBX, (usb_msg_t **)&p_mess);
          ctrl = *p_mess;
          switch (ctrl.event)
          {
                        :
                case USB_STS_CONFIGURED:
                              :
                      g_buf[0] = 0x3F               /* Page Code */
                      ctrl.module = USB_IP1;
                      ctrl.address = adr;
                      R_USB_HmscStrgCmd(&ctrl, &g_buf, USB_ATAPI_MODE_SENSE10);
                              :
                break;
                case USB_STS_MSC_CMD_COMPLETE:
                      if (ctrl.status == USB_CSW_FAIL)
                      {
                            R_USB_HmscStrgCmd(&ctrl, &g_buf, USB_ATAPI_REQUEST_SENSE);
                      }
                              :
                break;
           }
      }
}
```

6.2 R_USB_HmscGetDriveNo

ドライブ番号を取得する

形式

usb_err_t R_USB_HmscGetDriveNo(usb_ctrl_t *p_ctrl, uint8_t *p_drive)

引数

p_ctrl	usb_ctrl_t 構造体領域へのポインタ
p_drive	ドライブ番号を格納する領域へのポインタ

戻り値

USB_SUCCESS	正常終了
USB_ERR_PARA	パラメータエラー
USB_ERR_NG	その他のエラー

解説

usb_ctrl_t 構造体に指定された情報(メンバ module およびメンバ address)をもとに関連するドライブ番号を取得します。ドライブ番号は、引数 p_drive が示す領域に格納されます。

補足

1. 本 API をコールする前に usb_ctrl_t 構造体のメンバ address およびメンバ module に対し、ドライブ番号を取得したい MSC デバイスのデバイスアドレスおよびその MSC デバイスが接続された USB モジュール番号(USB_IP0/USB_IP1)を指定してください。なお、これらのメンバに対する指定に問題がある場合は、戻り値に USB_ERR_PARA が返されます。
2. 引数 p_ctrl に対し USB_NULL を指定した場合は、戻り値に USB_ERR_PARA が返されます。
3. USB デバイスが CONFIGURED 状態の場合に、本 API をコールすることができます。CONFIGURED 以外の状態で本 API をコールすると戻り値に USB_ERR_NG が返されます。

使用例

```
void usb_application( void )
{
    usb_ctrl_t ctrl;
    uint8_t    drive;
                    :
    while (1)
    {
        switch (R_USB_GetEvent(&ctrl))
        {
                    :
            case USB_STS_CONFIGURED:
                    :
                ctrl.module = USB_IP0;
                ctrl.address = adr;
                R_USB_HmscGetDriveNo( &ctrl, &drive );
                    :
            break;
                    :
        }
    }
}
```

6.3 R_USB_HmscGetSem

セマフォを取得する **(RTOS** のみ**)**

形式

void R_USB_HmscGetSem(void)

引数

--

戻り値

--

解説

本ドライバ内で使用しているセマフォの取得処理を行います。

補足

1. FAT ファイルオープン関数(e.g R_tfat_f_open)をコールする前に必ず本 API をコールしてください。
2. セマフォカウント値が 0(ゼロ)の時に、本 API がコールされた場合、本 API をコールしたタスクはセマフォ待ち状態に移行します。
3. 本 API が使用するセマフォの生成処理は USB ドライバ内で行われます。

使用例

```
/* Callback function */
void usb_apl_callback (usb_ctrl_t *p_ctr, rtos_task_id_t task_id, uint8_t is_request)
{
     USB_APL_SND_MSG(USB_APL_MBX, (usb_msg_t *)p_ctrl);
}

void          usb_application_task( void )
{
    usb_ctrl_t        ctrl;
    usb_ctrl_t        *p_mess;
              :
    while(1)
    {
        USB_APL_RCV_MSG(USB_APL_MBX, (usb_msg_t **)&p_mess);
        ctrl = *p_mess;
        switch (ctrl.event)
        {
                  :
           case USB_STS_CONFIGURED:
                  :
              R_USB_HmscGetSem();
              R_tfat_f_open(&file, (const char *) &g_msc_file[drvno][0],
                            (TFAT_FA_CREATE_ALWAYS | TFAT_FA_WRITE));
              R_tfat_f_write(&file, g_file_data, sizeof(g_file_data), &file_size);
              R_tfat_f_close(&file);
              R_USB_HmscRelSem();
                  :
           break;
                  :
        }
    }
}
```

6.4 R_USB_HmscRelSem

セマフォを解放する **(RTOS** のみ**)**

形式

void R_USB_HmscRelSem(void)

引数

--

戻り値

--

解説

本ドライバ内で使用しているセマフォの解放処理を行います。

補足

1. FAT ファイルクローズ関数(e.g R_tfat_f_close)をコールした後に必ず本 API をコールしてください。
2. R_USB_HmscGetSem 関数によるセマフォ待ち状態中のアプリケーションタスクは、本 API によりそのセマフォ待ち状態が解除されます。
3. 本 API が使用するセマフォの生成処理は USB ドライバ内で行われます。

使用例

```
/* Callback function */
void usb_apl_callback (usb_ctrl_t *p_ctr, rtos_task_id_t task_id, uint8_t is_request)
{
      USB_APL_SND_MSG(USB_APL_MBX, (usb_msg_t *)p_ctrl);
}

void          usb_application_task( void )
{
    usb_ctrl_t        ctrl;
    usb_ctrl_t        *p_mess;
                :
    while(1)
    {
                        :
            case USB_STS_CONFIGURED:
                        :
                R_USB_HmscGetSem();
                R_tfat_f_open(&file, (const char *) &g_msc_file[drvno][0],
                                (TFAT_FA_CREATE_ALWAYS | TFAT_FA_WRITE));
                R_tfat_f_write(&file, g_file_data, sizeof(g_file_data), &file_size);
                R_tfat_f_close(&file);
                R_USB_HmscRelSem();
                        :
            break;
                        :
        }
    }
}
```

7. Mass Storage コマンドの戻り値 (USB_STS_MSC_CMD_COMPLETE)

(1). Non-OS

R_USB_HmscStrgCmd 関数による Mass Storage コマンドの完了を認識した後で R_USB_GetEvent 関数をコールすると戻り値 USB_STS_MSC_CMD_COMPLETE が返されます。

(2). RTOS (FreeRTOS, uITRON のみ)

R_USB_HmscStrgCmd 関数による Mass Storage コマンドが完了すると USB ドライバから R_USB_Callback 関数により登録されたコールバック関数がコールされます。この時、コールバック関数の引数(usb_ctrl_t 構造体へのポインタ)のメンバ event に USB_STS_MSC_CMD_COMPLETE が設定されています。

Mass Storage コマンド完了時、usb_ctrl_t 構造体のメンバに設定される情報を以下に示します。

module	:	Mass Storage コマンドが完了した USB モジュール番号
address	:	Mass Storage コマンドが完了した USB デバイスのデバイスアドレス
size	:	応答データサイズ
status	:	CSW 情報

[Note]

1. usb_ctrl_t 構造体のメンバ module にはその USB デバイスが接続されている USB モジュール番号(USB_IP0 / USB_IP1)が設定され、メンバ address には Mass Storage コマンドが完了した USB デバイスのデバイスアドレスが設定されます。
2. メンバ size には MSC デバイスから送信される応答データのデータサイズが設定されます。
3. メンバ status には CSW(Command Status Wrapper)の bCSWStatus が設定されます。

USB_CSW_SUCCESS	(値:00H)	:	成功
USB_CSW_FAIL	(値:01H)	:	失敗
USB_CSW_PHASE	(値:02H)	:	フェーズエラー

8. コンフィグレーションファイル (RI600V4 使用時のみ)

RI600V4 を使用する場合、HMSC ドライバで使用する OS 資源を RI600V4 に登録する必要があります。以下の HMSC 関連の定義をコンフィグレーションファイルに追加してください。コンフィグレーションファイルの作成方法については、USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート(ドキュメント No.R01AN2025)内の「**コンフィグレーションファイル(RI600V4 使用時のみ)**」の章を参照してください。

8.1 メールボックス定義

1. メールボックス 1

name	:	ID_USB_RTOS_HMSC_MBX
wait_queue	:	TA_FIFO
message_queue	:	TA_MFIFO

2. メールボックス 2

name	:	ID_USB_RTOS_HMSC_REQ_MBX
wait_queue	:	TA_FIFO
message_queue	:	TA_MFIFO

8.2 セマフォ定義

name	:	ID_USB_RTOS_HMSC_SEM
max_count	:	1
initial_count	:	1
wait_queue	:	TA_FIFO

9. アプリケーションの作成方法

USB Basic Host and Peripheral Driver Firmware Integration Technology アプリケーションノート(ドキュメント No.R01AN2025)内の「**アプリケーションプログラムの作成方法**」の章を参照してください。

ホームページとサポート窓口

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

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Aug 1, 2014	—	初版発行
1.10	Dec 26, 2014	—	1. 対象デバイスに RX71M を追加。 2. 以下の API を追加。 R_usb_hmsc_alloc_drvno, R_usb_hmsc_free_drvno, R_usb_hmsc_ref_drvno 3. 以下の API に対し引数 drvno を追加。 R_usb_hmsc_SetDevSts, R_usb_hmsc_GetDevSts 4. 以下の API に対し引数 ipno を追加。 R_usb_hmsc_Information 5. MSC デバイスの複数接続をサポート。
1.11	Sep 30, 2015	—	対象デバイスに RX63N と RX631 を追加
1.20	Sep 30, 2016	—	1. 対象デバイスに RX65N/RX651 を追加 2. DMA 転送をサポート 3. USB Host and Peripheral Interface Driver アプリケーションノート (ドキュメント No.R01AN3293JJ)に対応
1.21	Mar 30, 2017	—	1. Technical Update(発行番号: TN-RX*-A172A/J)に対応しました。 2. 「6. API」以外の API については USB Basic Host and Peripheral Driver Firmware Integration Technology (ドキュメント番号 :R01AN2025)へ移しました。
1.22	Sep 30, 2017	—	RX65N/RX651-2M をサポート
1.23	Mar 31, 2018	—	Smart Configurator に対応しました。
1.24	Dec 28, 2018	—	1. RTOS をサポート 2. R_USB_HmscGetSem/R_USB_HmscRelSem 関数をサポート
1.25	Apr 16, 2019	—	対象デバイスに RX66T/RX72T を追加
1.26	May 31, 2019	—	1. GCC/IAR コンパイラをサポートしました。 2. 対象デバイスから RX63N を削除しました。
1.27	Jul 31, 2019	—	対象デバイスに RX72M を追加
1.30	Mar 1, 2020	—	1. リアルタイム OS(uITRON:RI600V4)をサポートしました。 2. 対象デバイスに RX72N/RX66N を追加
1.31	Mar 1, 2021	—	対象デバイスに RX671 を追加
1.41	Oct 30, 2022	—	Azure RTOS(USBX)をサポートしました。
1.42	Sep 30, 2023		Azure RTOS(USBX)で DMA 転送をサポートしました。

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。