RX Family

M3S-TFAT-Tiny Memory Driver Interface Module

Firmware Integration Technology

R20AN0335EJ0103
Rev.1.03
Oct 01, 2016

Introduction

This Application Note explains Memory Driver Interface combines RX Family Open source FAT filesystem M3S-TFAT-Tiny V.3.03 Release 00 (hereafter TFAT Library) with each memory drivers. TFAT Library package is provided as Firmware Integration Technology. (hereafter FIT).

Please refer to the following URL to know the details about FIT Modules.

https://www.renesas.com/en-us/solutions/rx-applications/fit.html

This Application Note provides the driver interface corresponding to USB and SDHI and USB Mini. Please use with following FIT Modules.

Function	Product	Website
File system (*1)	M3S-TFAT-Tiny (R20AN0038)	http://www.renesas.com/mw/tfat
USB Drive (*2)	USB Basic Host and Peripheral Driver USB Host Mass Storage Class Driver	http://www.renesas.com/driver/usb
SDHI Driver (*2)	SDHI Driver	This module is not published. Please contact our customer center.
USB Mini Drive (*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

*1 This is required.
*2 One module is required.
—

Target Device

RX Family

Contents

1. Overview

1.1 This Application Note

This Application explains memory driver interface combines TFAT Library and each memory drivers. This Module can change the target of memory driver using config file.

The APIs provided by this module are called by TFAT Library. It is no need to call by user.

The drive number controlled in TFAT Library and the drive number controlled in USB/SDHI driver are not equal. Therefore this module has the conversion table for drive. Initial value can be configured, please refer to the section 3.7 if you change this as dynamic.

1.2 Structure of application note

This application note includes files below.

Table 1.2.1 Structure of application note

file/folder name	description
r20an0335ej0103-rx-tfat.pdf	Application note
reference_document	
r01an1723eu0111_rx.pdf	Adding Firmware Integration Technology Modules to e^2 studio
r01an1826ej0102_rx.pdf	Adding Firmware Integration Technology Modules to CS+
FITModules	
r_tfat_driver_rx_v1.03.xml	FIT plug-in XML
r_tfat_driver_rx_v1.03.zip	FIT plug-in ZIP
configuration (r_config)	
r_tfat_driver_rx_config.h	configuration file(default)
FIT Module (r_tfat_driver_rx)	
document(doc)	
English(en)	
r20an0335ej0103-rx-tfat.pdf	Application note (English)
Japanese(ja)	
r20an0335jj0103-rx-tfat.pdf	Application note (Japanese)
configuration refer reference (ref)	
r_tfat_driver_rx_config_reference.h	configuration file(template)
source code(src)	
readme (readme.txt)	readme
r_tfat_driver_rx_if.h	Header file

1.3 Structure of Software

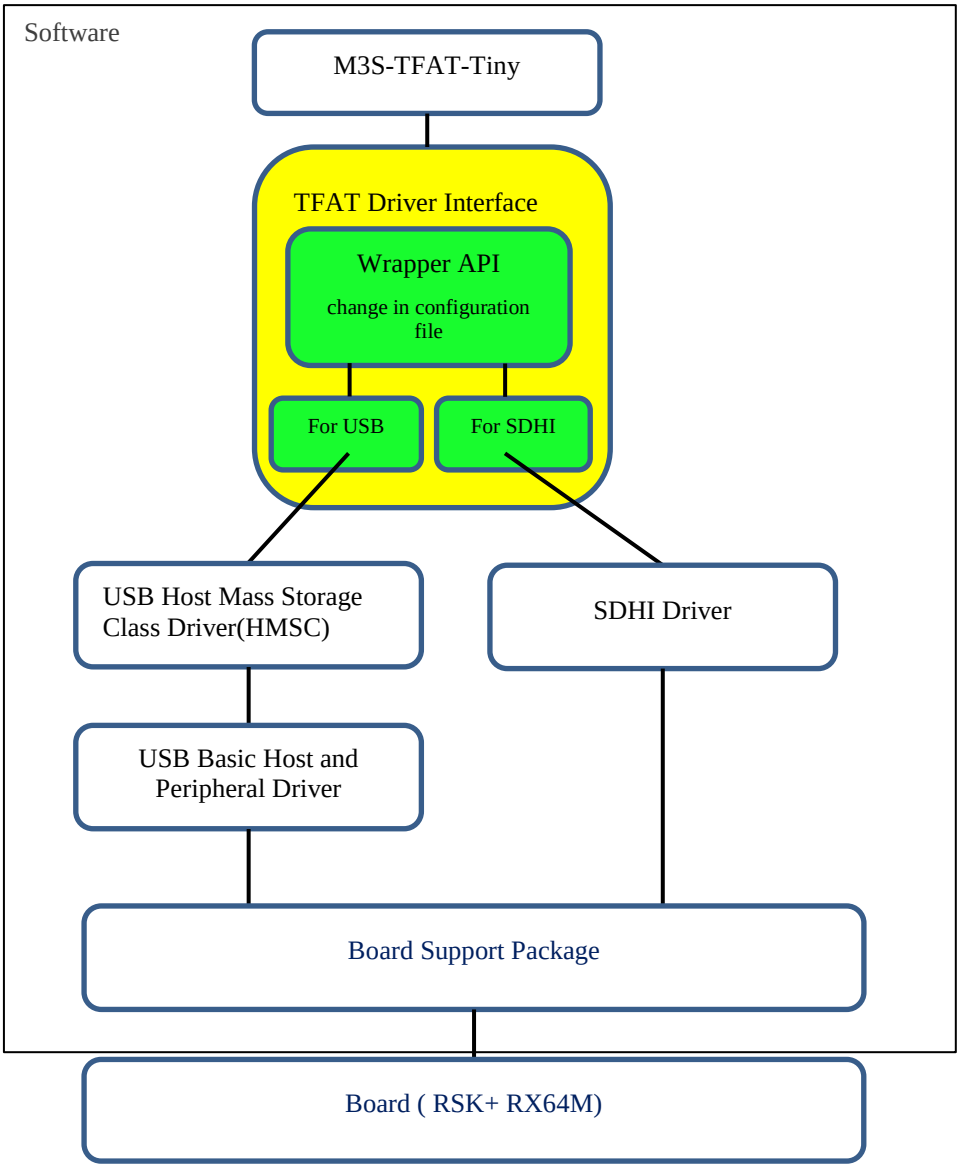

Table 1.3.1 Using FIT Modules version

Product	version
Board Support Package (BSP)	3.30
M3S-TFAT-Tiny	3.03
USB Driver USB Basic Host and Peripheral Driver USB Host Mass Storage Class Driver	 1.11 1.11
SDHI Driver	1.00
USB Basic Host and Peripheral Driver (USB Mini Firmware) USB Host Mass Storage Class Driver for USB Mini Firmware.	1.02 1.02

—

2. API Information

2.1 Hardware Requirements

The microcontroller used must support the following functionality.

- USB
- SDHI

2.2 Software Requirements

This FIT Module is dependent on the following packages:

- r_sdhi_rx
- r_usb_hmsc
- r_usb_hmsc_mini

The kind of memory driver can be set in r_tfat_driver_rx_config.h.

Table 1.3.2 Usable Memory Driver

	RX 110	RX 111	RX 113	RX 210	RX 231	RX 23T	RX 63N	RX 64M	RX 71M	RX 65N
r_usb_hmsc	–	–	–	–	–	–	✔	✔	✔	✔
r_usb_hmsc_mini	–	✔	✔	–	✔	–	–	–	–	–
r_sdhi_rx	–	–	–	–	✔	–	–	✔	✔	✔

SDHI and USB can be used in same time.

2.3 Supported Toolchain

This driver is tested and working with the following toolchains:

Renesas RXC Toolchain v.2.04.01

2.4 Header Files

All API calls are accessed by including a single file "r_tfat_driver_rx_if.h" which is supplied with this software's project code. Build-time configuration options are selected or defined in the file "r_tfat_driver_rx_config.h".

2.5 Integer Types

This project uses ANSI C99 “Exact width integer types” in order to make the code clearer and more portable. These types are defined in stdint.h.

2.6 Configuration Overview

The configuration options in this module are specified in r_tfat_driver_rx_config.h. The option names and setting values are listed in the table below.

Configuration options in r_tfat_driver_rx_config.h	
#define TFAT_USB_DRIVE_NUM - Default value = (0)	The number of drives for USB. Please set (0) if user does not use USB.
#define TFAT_SDHI_DRIVE_NUM - Default value = (0)	The number of drives for SDHI. Please set (0) if user does not use SDHI.
#define TFAT_USB_MINI_DRIVE_NUM - Default value = (0)	The number of drives for USB Mini. Please set (0) if user does not use USB Mini.
#define TFAT_DRIVE_ALLOC_NUM_i i = 0～9 - Default value = (NULL)	This config allocates the device for each drive number. The drive for USB = (TFAT_CTRL_USB) The driver for SDHI=(TFAT_CTRL_SDHI) The driver for USB Mini =(TFAT_CTRL_USB_MINI) The driver for “not using” = (NULL) This module uses these parameters for relating the drive number (TFAT Library) with the drive number of memory driver. The drive number is allocated ascending order. Please refer to the section 3.7 if user change this in dynamic.

2.7 Arguments

Please use definition of drive number when calling TFAT Library.

```
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;
```

2.8 Return Values

Return values are defined in “r_tfat_lib.h” in TFAT module.

```
/* Disk Status Bits (DSTATUS) */
typedef uint8_t DSTATUS;
```

- #define TFAT_STA_NOINIT 0x01 /* Drive not initialized */
- #define TFAT_STA_NODISK 0x02 /* No medium in the drive */
- #define TFAT_STA_PROTECT 0x04 /* Write protected */

```
/* Results of Disk Functions */
typedef enum
{
        TFAT_RES_OK = 0,        /* 0: Successful */
        TFAT_RES_ERROR,         /* 1: R/W Error */
        TFAT_RES_WRPRT,         /* 2: Write Protected */
        TFAT_RES_NOTRDY,        /* 3: Not Ready */
        TFAT_RES_PARERR         /* 4: Invalid Parameter */
} DRESULT;
```

2.9 Adding The FIT Module to Your Project

Please refer to the Adding Firmware Integration Technology Modules to Projects (r01an1723eu0111_rx.pdf, for e^2 studio) or the Adding Firmware Integration Technology Modules to CS+ Projects (r01an1826ej0102_rx.pdf).

3. API Functions

These functions are called TFAT module, these functions calls lower layer functions according to the configuration. Section 4 explains lower functions. The (*1) functions do not call lower functions.

Table 3.1 Functions List

Function Name	Function Overview
R_tfat_disk_initialize	Initialize disk drive
R_tfat_disk_read	Read sectors
R_tfat_disk_write	Write sectors
R_tfat_disk_ioctl	Control device dependent features
R_tfat_disk_status	Get disk status
R_tfat_get_fattime (*1)	Get current time
R_tfat_drv_change_alloc (*1)	Change the relating of TFAT module drive number with memory driver drive number.

*1 These functions do not call lower functions.

—

3.1 R_tfat_disk_initialize

Description

Initialize disk drive.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_disk_initialize (uint8_t  drive);
```

Parameters

drive	input	Specifies the initialize drive number.

Return Value

DSTATUS	Status of the disk after function execution as explained in section 2.8.

Remark

This function calls lower functions according to configuration. The lower function is R_tfat_(memory name)_disk_initialize.

3.2 R_tfat_disk_read

Description

This function read the data from disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
Buffer	output	Pointer to the read buffer to store the read data. A buffer of the size equal to the number of bytes to be read is required.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

This function calls lower function according to configuration. The lower function is R_tfat_(memory name)_disk_read.

3.3 R_tfat_disk_write

Description

This function writes the data to disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
buffer	input	Pointer to the data to be written.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

This function calls lower function according to configuration. The lower function is R_tfat_(memory name)_disk_write.

3.4 R_tfat_disk_ioctl

Description

This function controls the drive.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
command	input	Specifies the command code. The command code will always be 0.
buffer	input	Pointer should always be a NULL pointer.

Return Value

DRESULT Result of the function execution as explained in section 2.8.

Remark

R_tfat_disk_ioctl function is called from R_tfat_f_sync.

This function calls lower function according to configuration. The lower function is R_tfat_(memory name)_disk_ioctl.

3.5 R_tfat_disk_status

Description

This function gets the information about disk drive status.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_disk_status (uint8_t  drive );
```

Parameters

drive	input	Specifies the physical drive number.

Return Value

DSTATUS Status of the disk after function execution as explained in section 2.8.

Remark

This function calls lower function according to configuration. The lower function is R_tfat_(memory name)_disk_status.

3.6 R_tfat_get_fattime

Description

This function gets the time information.

Usage

```
#include "r_tfat_lib.h"
uint32_t R_tfat_get_fattime (void );
```

Parameters

None

Return Value

uint32_t Please refer the following table for explanation of return value.

Bit Range	Value Range	Significance
31 o 25	0 to 127	Year from 1980
24 o 21	1 to 12	Month
20 o 16	1 to 31	Day
15 o 11	0 to 23	Hour
10 o 5	0 to 59	Minute
4 to 0	0 to 29	Second / 2

Remark

This function returns the current date and time.

This function is used by the library functions for retrieving date during file operations.

3.7 R_tfat_drv_change_alloc

Description

This function change the relating of TFAT module drive number and memory driver drive number.

Usage

```
#include "r_tfat_lib.h"
#include "r_tfat_driver_rx_if.h”
DRESULT R_tfat_drv_change_alloc (TFAT_DRV_NUM  tfat_drv,
                                 uint8_t dev_type,
                                 uint8_t dev_drv_num );
```

Parameters

tfat_drv	input	The drive number for TFAT Library
dev_type	input	Device type (TFAT_CTRL_USB/TFAT_CTRL_SDHI)
dev_drv_num	input	The drive number for memory driver

Return Value

DRESULT Result of the function execution as explained in section 2.8.

Remark

This function updates the table information about drive number relation.

Relation table initial configuration are set in config file. If user would like to change this configuration as dynamic, please use this function.

Please do not call this function during other APIs in this module are called.

4. Local API

For USB, for SDHI, for USB Mini functions are prepared. Each function calls own memory driver functions.

4.1 For USB

Table 4.1.1. Functions are called when Section 2.6 TFAT_USB_DRIVE_NUM and TFAT_DRIVE_ALLOC_NUM_i(i=0-9) have the settings “TFAT_CTRL_USB”.

Table 4.1.1 Functions List

Function name	Function Overview
R_tfat_usb_disk_initialize	Initialize disk drive
R_tfat_usb_disk_read	Read sectors
R_tfat_usb_disk_write	Write sectors
R_tfat_usb_disk_ioctl	Control device dependent features
R_tfat_usb_disk_status	Get disk status

Table 4.1.2 Other Functions List

Function name	Function Overview
R_usb_hmsc_WaitLoop	Wait for read and write

4.1.1 R_tfat_usb_disk_initialize

Description

This function initialize the disk drive.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_usb_disk_initialize (uint8_t drive);
```

Parameters

drive input Specifies the initialize drive number.

Return Value

DSTATUS Status of the disk after function execution as explained in section 2.8.

Remark

This API does not call USB driver initialize function because of USB driver limitation (1 time call is only accepted). Please call USB driver initialize function in user program.

4.1.2 R_tfat_usb_disk_read

Description

This function reads the data from disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
Buffer	output	Pointer to the read buffer to store the read data. A buffer of the size equal to the number of bytes to be read is required.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

This function reads the data from disk drive. The position of read data is specified using this function argument.

4.1.3 R_tfat_usb_disk_write

Description

This function writes the data to the disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
buffer	input	Pointer to the data to be written.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

This function writes the data to the disk drive. The position of write data is specified using this function argument.

4.1.4 R_tfat_usb_disk_ioctl

Description

This function controls the drive.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
command	input	Specifies the command code. The command code will always be 0.
buffer	input	Pointer should always be a NULL pointer.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

The R_tfat_disk_ioctl function is used only by the R_tfat_f_sync function amongst all the TFAT library functions. Users who do not plan to use R_tfat_f_sync function in their applications can skip the implementation for this particular driver interface function.

For users who wish to use R_tfat_f_sync function in their applications, this particular driver interface function will have to be implemented. This driver function should consist of the code to finish off any pending write process. If the disk i/o module has a write back cache, the dirty sector must be flushed immediately. The R_tfat_f_sync function will perform a save operation to the unsaved data related to the fileobject passed as argument.

4.1.5 R_tfat_usb_disk_status

Description

This function gets the information about disk drive.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_usb_disk_status (uint8_t  drive );
```

Parameters

drive	input	Specifies the physical drive number.

Return Value

DSTATUS	Status of the disk after function execution as explained in section 2.8.

Remark

This function should consist of the code that checks the disk and returns the current disk status. The disk status can have any of the three values as explained in section 2.8. The disk status can be returned by updating the return value with the macros related to disk status.

4.1.6 R_usb_hmsc_WaitLoop

Description

This function waits the data read/write.

Usage

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

Parameters

None

Return Value

None

Remark

Please refer to the USB driver document for details.

4.2 For SDHI

Table 4.2.1. Functions are called when Section 2.6 TFAT_SDHI_DRIVE_NUM and TFAT_DRIVE_ALLOC_NUM_i (i=0-9) have the settings “TFAT_CTRL_SDHI”.

Table 4.2.1 List of Functions

Function Name	Outline
R_tfat_sdhi_disk_initialize	Initialize disk drive
R_tfat_sdhi_disk_read	Read sectors
R_tfat_sdhi_disk_write	Write sectors
R_tfat_sdhi_disk_ioctl	Control device dependent features
R_tfat_sdhi_disk_status	Get disk status

[Notice about SDHI

This module does not execute mount process and VDD power supply process. Please refer to the SDHI module document and please implement.

4.2.1 R_tfat_sdhi_disk_initialize

Description

This function initializes the disk drive.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_sdhi_disk_initialize (uint8_t drive);
```

Parameters

drive	input	Specifies the initialize drive number.

Return Value

DSTATUS	Status of the disk after function execution as explained in section 2.8.

Remark

This function does not execute the SDHI driver initialize. Please implement SDHI initialize code in user code.

4.2.2 R_tfat_sdhi_disk_read

Description

This function reads the data from disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
Buffer	output	Pointer to the read buffer to store the read data. A buffer of the size equal to the number of bytes to be read is required.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

Read data from SD memory.

4.2.3 R_tfat_sdhi_disk_write

Description

This function writes the data to the disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
buffer	input	Pointer to the data to be written.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

Writes the data to the SD memory.

4.2.4 R_tfat_sdhi_disk_ioctl

Description

This function controls the drive.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
command	input	Specifies the command code. The command code will always be 0.
buffer	input	Pointer should always be a NULL pointer.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

The R_tfat_sdhi_disk_ioctl function is used only by the R_tfat_f_sync function amongst all the TFAT library functions. Users who do not plan to use R_tfat_f_sync function in their applications can skip the implementation for this particular driver interface function.

This module does not implement.

4.2.5 R_tfat_sdhi_disk_status

Description

This function gets the disk drive status.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_sdhi_disk_status (uint8_t  drive );
```

Parameters

drive	input	Specifies the physical drive number.

Return Value

DSTATUS	Status of the disk after function execution as explained in section 2.8.

Remark

This function should consist of the code that checks the disk and returns the current disk status. The disk status can have any of the three values as explained in section 2.8. The disk status can be returned by updating the return value with the macros related to disk status.

This module does not implement.

4.3 For USB Mini

Table 4.1.1. Functions are called when Section 2.6 TFAT_USB_MINI_DRIVE_NUM and TFAT_DRIVE_ALLOC_NUM_i(i=0-9) have the settings “TFAT_CTRL_USB_MINI”.

Table 4.3.1 Functions List

Function name	Function Overview
R_tfat_usb_mini_disk_initialize	Initialize disk drive
R_tfat_usb_mini_disk_read	Read sectors
R_tfat_usb_mini_disk_write	Write sectors
R_tfat_usb_mini_disk_ioctl	Control device dependent features
R_tfat_usb_mini_disk_status	Get disk status

Table 4.3.2 Other Functions List

Function name	Function Overview
R_usb_mini_hmsc_WaitLoop	Wait for read and write

4.3.1 R_tfat_usb_mini_disk_initialize

Description

This function initialize the disk drive.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_usb_mini_disk_initialize (uint8_t drive);
```

Parameters

drive	input	Specifies the initialize drive number.

Return Value

DSTATUS Status of the disk after function execution as explained in section 2.8.

Remark

This API does not call USB driver initialize function because of USB driver limitation (1 time call is only accepted). Please call USB driver initialize function in user program.

4.3.2 R_tfat_usb_mini_disk_read

Description

This function reads the data from disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
Buffer	output	Pointer to the read buffer to store the read data. A buffer of the size equal to the number of bytes to be read is required.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

This function reads the data from disk drive. The position of read data is specified using this function argument.

4.3.3 R_tfat_usb_mini_disk_write

Description

This function writes the data to the disk.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
buffer	input	Pointer to the data to be written.
sector_number	input	Specifies the start sector number in logical block address (LBA).
sector_count	input	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

This function writes the data to the disk drive. The position of write data is specified using this function argument.

4.3.4 R_tfat_usb_mini_disk_ioctl

Description

This function controls the drive.

Usage

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

Parameters

drive	input	Specifies the physical drive number.
command	input	Specifies the command code. The command code will always be 0.
buffer	input	Pointer should always be a NULL pointer.

Return Value

DRESULT	Result of the function execution as explained in section 2.8.

Remark

The R_tfat_usb_mini_disk_ioctl function is used only by the R_tfat_f_sync function amongst all the TFAT library functions. Users who do not plan to use R_tfat_f_sync function in their applications can skip the implementation for this particular driver interface function.

For users who wish to use R_tfat_f_sync function in their applications, this particular driver interface function will have to be implemented. This driver function should consist of the code to finish off any pending write process. If the disk i/o module has a write back cache, the dirty sector must be flushed immediately. The R_tfat_f_sync function will perform a save operation to the unsaved data related to the fileobject passed as argument.

4.3.5 R_tfat_usb_mini_disk_status

Description

This function gets the information about disk drive.

Usage

```
#include "r_tfat_lib.h"
DSTATUS R_tfat_usb_mini_disk_status (uint8_t drive);
```

Parameters

drive	input	Specifies the physical drive number.

Return Value

DSTATUS	Status of the disk after function execution as explained in section 2.8.

Remark

This function should consist of the code that checks the disk and returns the current disk status. The disk status can have any of the three values as explained in section 2.8. The disk status can be returned by updating the return value with the macros related to disk status.

4.3.6 R_usb_mini_hmsc_WaitLoop

Description

This function waits the data read/write.

Usage

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

Parameters

None

Return Value

None

Remark

Please refer to the USB driver document for details.

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/contact/

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.03	Oct 01, 2016	-	Added support RX family.
1.02	Jun 30, 2015	-	Added support MCU RX231.
1.01	Jan 05, 2015	-	Added support MCUs.
1.00	Dec 01, 2014	-	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0