User's Manual

Open Source FAT File System [M3S-TFAT-Tiny]

User's Manual

Renesas Micro Computer Middleware

Renesas Electronics
www.renesas.com

Rev.3.01 2014.04

Introduction

This document explains the usage of the Open Source FAT File System [M3S-TFAT-Tiny] (hereafter referred to as “TFAT Library”) for Renesas Microcomputer.

The TFAT library was made based on FatFs. I show below the relations of each source version.

$$\begin{tikzcd}
\text{FatFs } 0.06 \arrow[r] \arrow[d] & 0.07 \arrow[r] & 0.08 \arrow[r] & 0.09 \arrow[r] & 0.09a \arrow[r] & 0.09b \arrow[r] \arrow[d] & 0.10 \\
\text{TFAT } 1.00 \arrow[rr] & & 2.00 \arrow[rrr] & & & 3.00 & \\
\begin{array}{l}\cdot\text{R8C}\\ \cdot\text{H8}\\ \cdot\text{M16C}\end{array} & & \begin{array}{l}\cdot\text{RL78}\\ \cdot\text{SH2A}\\ \cdot\text{V850}\end{array} & & & \cdot\text{RX} &
\end{tikzcd}$$

Figure 1. Relations of the version of TFAT library and FatFs

What is FatFs?

FatFs is the File system module for the small embedded system.

Fat Fs is developed by ChaN Software.

FatFs is provided as non-payment for embedded system.

Please refer to the Website below for more details.

http://elm-chan.org/fsw/ff/00index_e.html

Limitation about Long File Name (hereafter LFN):

Microsoft has some Patents about Filesystem.

Each Patents are about LFN implementation, Microsoft requires license payment about it.

TFAT Library can select LFN function disable/enebale. If you do not need LFN, please select disable (default).

Table of Contents

List of Figures

List of Tables

1. Library specifications

1.1. specification of TFAT Library

Following are some of the main specifications of the TFAT Library.

Table 1.1. Specifications of the TFAT Library

Item	Specifications
Base software	FatFs R0.09b
Supported FAT Type	FAT16, FAT32
FileName Support	8.3 format (8 lettered filename & lettered extension)
File system format function support	None
Number of drives supported	MAX 10
Logical Sector size	512byte
ROM size(*1)	About 10Kbyte
Work area for File system mounting per drive(*1)	About 500byte
Work area for File access(*1)	About 50byte
Stack size(*1)	About 300byte

(*1)The detailed value refers to an FIT Module Application Note.

1.2. Structure of software stack

Following are structure of software stack of the TFAT Library.

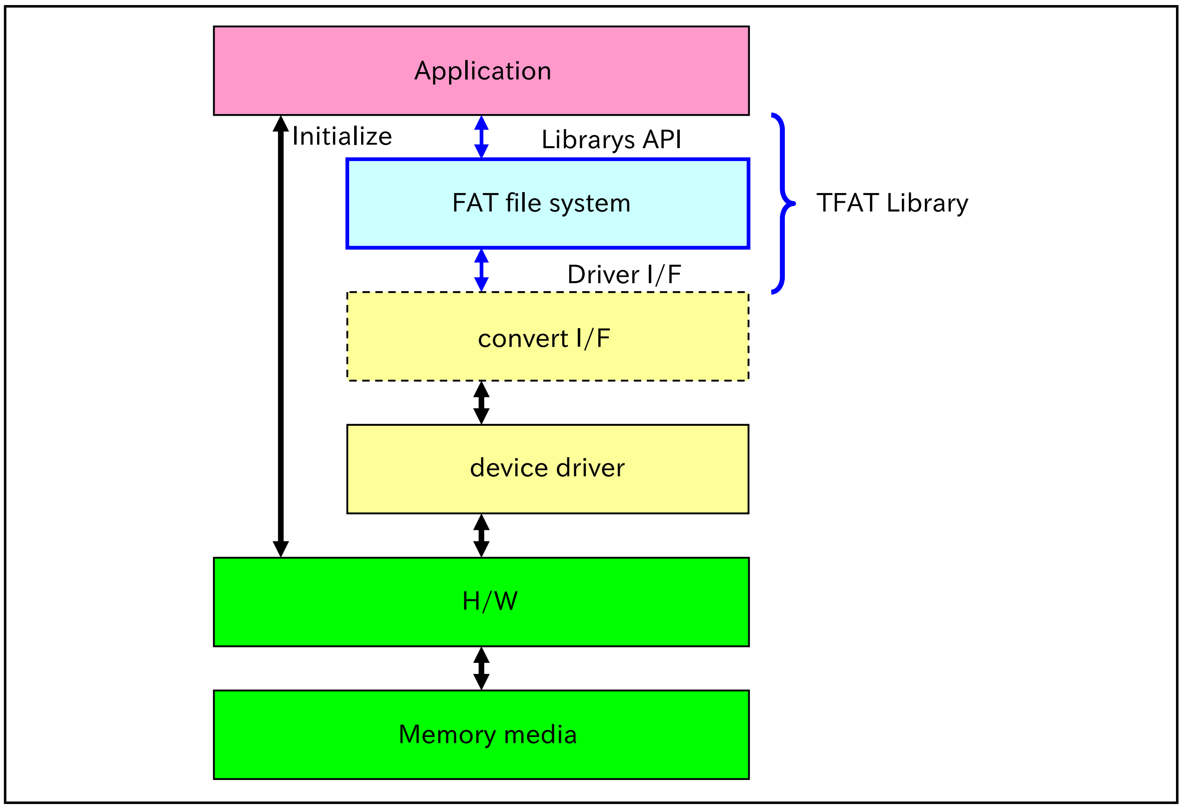


Figure 1.1. Structure of software stack of the TFAT Library

Sample program works in various Renesas Starter Kit.

An expansion boards such as memory card slots may be necessary for various Renesas Starter Kit.

Please refer to an FIT Module Application Note for the details.

1.3. Function of the TFAT library

I show below a list of functions of the TFAT Library.

Table 1.2. List of functions

Functions	TFAT	FatFs	Windows
Support file system	FAT16 FAT32	FAT16 FAT32	FAT12 FAT16 FAT32 NTFS
Support 2byte code	Japanese(S-JIS)	Japanese(S-JIS) Simplified Chinese(GB18030) Korean(EUC) Traditional Chinese(Big5) etc.	Japanese(S-JIS) etc.

1.4. Config setting of the TFAT library

I show below a list of config setting of the TFAT Library. This version cannot change the settings excluding _USE_LFN feature.

Table 1.3. List of config setting

Optional flag	Explanation	Configurable	Default value
_FS_TINY	Choice of standard constitution and the smallest constitution	×	1
_FS_READONLY	build it as a library for exclusive use of the reading	×	0
_FS_MINIMIZE	set a limit to a function	×	0
_USE_STRFUNC	Disable/Enable the function input/ output strings to file (fputs,fprintf)	×	0
_USE_MKFS	To enable f_mkfs function, set _USE_MKFS to 1 and set _FS_READONLY to 0	×	0
_USE_FASTSEEK	To enable fast seek feature, set _USE_FASTSEEK to 1.	×	0
_USE_LABEL	To enable volume label functions, set _USE_LAVEL to 1	×	0
_USE_FORWARD	To enable f_forward function, set _USE_FORWARD to 1 and set _FS_TINY to 1.	×	1
_CODE_PAGE	The _CODE_PAGE specifies the OEM code page to be used on the target system.	×	932
_USE_LFN	To enable long file name functions, set _USE_LFN to 1	○	0(*1)
_MAX_LFN	Maximum LFN length to handle (12 to 255)	×	255
_LFN_UNICODE	To switch the character code set on FatFs API to Unicode, enable LFN feature and set _LFN_UNICODE to 1.	×	0
_FS_RPATH	The _FS_RPATH option configures relative path feature.	×	0
_VOLUMES	Number of volumes (logical drives) to be used.	×	10
_MAX_SS	Maximum sector size to be handled. (512, 1024, 2048 or 4096)	×	512
_MULTI_PARTITION	To enable multi partition function, set _MULTI_PARTITION to 1.	×	0
_USE_ERASE	To enable sector erase feature, set _USE_ERASE to 1.	×	0
_WORD_ACCESS	Set 0 first and it is always compatible with all platforms.	×	0

Optional flag	Explanation	Configurable	Default value
_FS_REENTRANT	This option switches the reentrancy (thread safe) of the FatFs module.	×	0
_FS_TIMEOUT	Timeout period in unit of time ticks	×	1000
_SYNC_t	O/S dependent type of sync object.	×	HANDLE
_FS_LOCK	To enable file lock control feature, set _FS_LOCK to 1 or greater.	×	0

(*1)0 or 1 value are available.

1.5. The front for API of TFAT and FatFs

Table 1.4. The front for API of TFAT and FatFs

TFAT	FatFs
R_tfat_f_mount	f_mount
R_tfat_f_open	f_open
R_tfat_f_read	f_read
R_tfat_f_write	f_write
R_tfat_f_lseek	f_lseek
R_tfat_f_close	f_close
R_tfat_f_opendir	f_opendir
R_tfat_f_readdir	f_readdir
R_tfat_f_stat	f_stat
R_tfat_f_getfree	f_getfree
R_tfat_f_truncate	f_truncate
R_tfat_f_sync	f_sync
R_tfat_f_unlink	f_unlink
R_tfat_f_mkdir	f_mkdir
R_tfat_f_chmod	f_chmod
R_tfat_f_utime	f_utime
R_tfat_f_rename	f_rename
R_tfat_f_forward	f_forward

2. Library type definitions

This section gives the details about the type definitions used in the library.

Table 2.1. Library type definitions

Datatype	Typedef
signed char	int8_t
unsigned char	uint8_t
signed short	int16_t
unsigned short	uint16_t
signed long	int32_t
unsigned long	uint32_t
unsigned char	DSTATUS

3. Library structures

This section gives the details of the structures used in the library.

3.1. FATFS - File system object structure

FATFS structure has a work area for logical drive. It is allocated by the application program and registered/unregistered with R_tfat_f_mount function. The following table gives the details of the members of the FATFS structure. No member of this structure can be changed from the application program.

Table 3.1. Structure members of FATFS

Datatype	Structure element	Explanation
uint8_t	fs_type	FAT sub type
uint8_t	drv	Physical drive number
uint8_t	csize	Sectors per cluster
uint8_t	n_fats	Number of FAT copies
uint8_t	wflag	win[] dirty flag (1:must be written back)
uint8_t	fsi_flag	fsinfo dirty flag (1:must be written back)
uint16_t	id	File system mount ID
uint16_t	n_rootdir	Number of root directory entries
uint16_t	ssize	Bytes per sector
uint32_t	last_clust	Last allocated cluster
uint32_t	free_clust	Number of free clusters
uint32_t	fsi_sector	fsinfo sector
uint32_t	cdir	Current directory start cluster (0:root)
uint32_t	n_fatent	Number of FAT entries (= number of clusters + 2)
uint32_t	fsize	Sectors per FAT
uint32_t	volbase	Volume start sector
uint32_t	fatbase	FAT start sector
uint32_t	dirbase	Root directory start sector
uint32_t	database	Data start sector
uint32_t	winsect	Current sector appearing in the win[]
uint8_t	win[512]	Disk access window for Directory, FAT (and Data on tiny cfg)

3.2. DIR - Directory object structure

DIR structure (Directory Object) has related data from directory info.

The related data from directory info is stored to DIR structure used R_tfat_f_opendir() or R_tfat_f_readdir() functions.

Table 3.2. Structure members of DIR

Datatype	Structure element	Explanation
FATFS*	fs	Pointer to the owner file system object

Datatype	Structure element	Explanation
uint16_t	id	Owner file system mount ID
uint16_t	index	Current index
uint32_t	sclust	Start cluster
uint32_t	clust	Current cluster
uint32_t	sect	Current sector
uint8_t*	dir	Pointer to the current SFN entry in the win[]
uint8_t*	fn	Pointer to the SFN (in/out)
uint16_t*	lfn	Pointer to the LFN working buffer
uint16_t	lfn_idx	Last matched LFN index number

3.3. FIL - File object structure

The FIL structure (file object) holds state of a file. It is created by R_tfat_f_open function and discarded by R_tfat_f_close function. No member of this structure can be changed by the application program.

Table 3.3. Structure members of FIL

Datatype	Structure element	Explanation
FATFS*	fs	Pointer to the related file system object
uint16_t	id	Owner file system mount ID
uint8_t	flag	File status flags
uint8_t	pad1	padding
uint32_t	fptr	File R/W pointer
uint32_t	fsize	File size
uint32_t	sclust	File start cluster
uint32_t	clust	Current cluster
uint32_t	dsect	Current sector
uint32_t	dir_sect	Sector containing the directory entry
uint8_t*	dir_ptr	Pointer to the directory entry in the window

3.4. FILINFO - File status structure

The FILINFO structure holds the file information returned by R_tfat_f_stat() and R_tfat_f_readdir() functions.

Table 3.4. Structure members of FILINFO

Datatype	Structure element	Explanation
uint32_t	fsize	Stores the size of file in bytes. This is always zero in case of a directory.
uint16_t	fdate	Stores the date when the file was modified or the directory was created. bit15:9 - Year from 1980 (Value in the range of 0 to 127) bit8:5 - Month (Value in the range 1 to 12)

Datatype	Structure element	Explanation
		bit4:0 - Day (Value in the range 1 to 31)
uint16_t	ftime	Stores the time when the file was modified or the directory was created. bit15:11 - Hour (Value in the range 0 to 23) bit10:5 - Minute (Value in the range 0 to 59) bit4:0 - Second / 2 (Value in the range 0 to 29)
uint8_t	fattrib	Stores the file/directory attributes.
uint8_t	fname[8+1+3+1]	Stores the file/directory name in 8.3 format null-terminated string.
uint8_t*	lfname	Pointer to the LFN buffer
uint16_t	lfsize	Size of LFN buffer in TCHAR

4. Library constants

This section gives the details of the constants used in the library. These constans are defined by r_tfat_lib.h

4.1. FRESULT - File function return code

Return value of Library function is defined typeof enum.

Table 4.1. FRESULT Value

Enum Name	Value	Significance
TFAT_FR_OK	0	Succeeded
TFAT_FR_DISK_ERR	1	A hard error occurred in the low level disk I/O layer
TFAT_FR_INT_ERR	2	Assertion failed
TFAT_FR_NOT_READY	3	The physical drive cannot work
TFAT_FR_NO_FILE	4	Could not find the file
TFAT_FR_NO_PATH	5	Could not find the path
TFAT_FR_INVALID_NAME	6	The path name format is invalid
TFAT_FR_DENIED	7	Access denied due to prohibited access or directory full
TFAT_FR_EXIST	8	Access denied due to prohibited access
TFAT_FR_INVALID_OBJECT	9	The file/directory object is invalid
TFAT_FR_WRITE_PROTECTED	10	The physical drive is write protected
TFAT_FR_INVALID_DRIVE	11	The logical drive number is invalid
TFAT_FR_NOT_ENABLED	12	The volume has no work area
TFAT_FR_NO_FILESYSTEM	13	There is no valid FAT volume
TFAT_FR_MKFS_ABORTED	14	The f_mkfs() aborted due to any parameter error
TFAT_FR_TIMEOUT	15	Could not get a grant to access the volume within defined period
TFAT_FR_LOCKED	16	The operation is rejected according to the file sharing policy
TFAT_FR_NOT_ENOUGH_CORE	17	LFN working buffer could not be allocated
TFAT_FR_TOO_MANY_OPEN_FILES	18	Number of open files > _FS_SHARE
TFAT_FR_INVALID_PARAMETER	19	Given parameter is invalid

4.2. File Attribute information

These macros is values to be set the fattrib member of FILINFO structure. The following list show the contents of each bit.

Table 4.2. File Attribute information macros

Name	Value	Explanation
TFAT_AM_RDO	0x01	When this flag is set,the applicable file(or directory) is read only.

Name	Value	Explanation
TFAT_AM_HID	0x02	When this flag is set,the applicable file(or directory) is hidden.
TFAT_AM_SYS	0x04	When this flag is set,the applicable file(or directory) is system file.
TFAT_AM_DIR	0x10	When this flag is set,the applicable file is directory.
TFAT_AM_ARC	0x20	When this flag is set,the applicable file(or directory) is Archive.

4.3. Macros for Disk Status

These macros shows status of disk to set in DSTATUS type. User sets applicable macro by Memory driver interface function and passes a result to a library.

Table 4.3. Macros for Disk Status

Macros	Value	Explanation
TFAT_STA_NOINIT	0x01	This flag indicates that the disk drive has not been initialized. This flag is set on: system reset, disk removal and failure of R_tfat_disk_initialize function, and cleared on: success of R_tfat_disk_initialize function.
TFAT_STA_NODISK	0x02	If this flag is set, it indicates that there is no media in the drive. This is flag is cleared when media is present in the drive.
TFAT_STA_PROTECT	0x04	This flag is used to indicate that the media is write protected. This is always cleared on the drive that does not support write protect notch. This flag is not valid when TFAT_STA_NODISK is set.

4.4. Enum - DRESULT

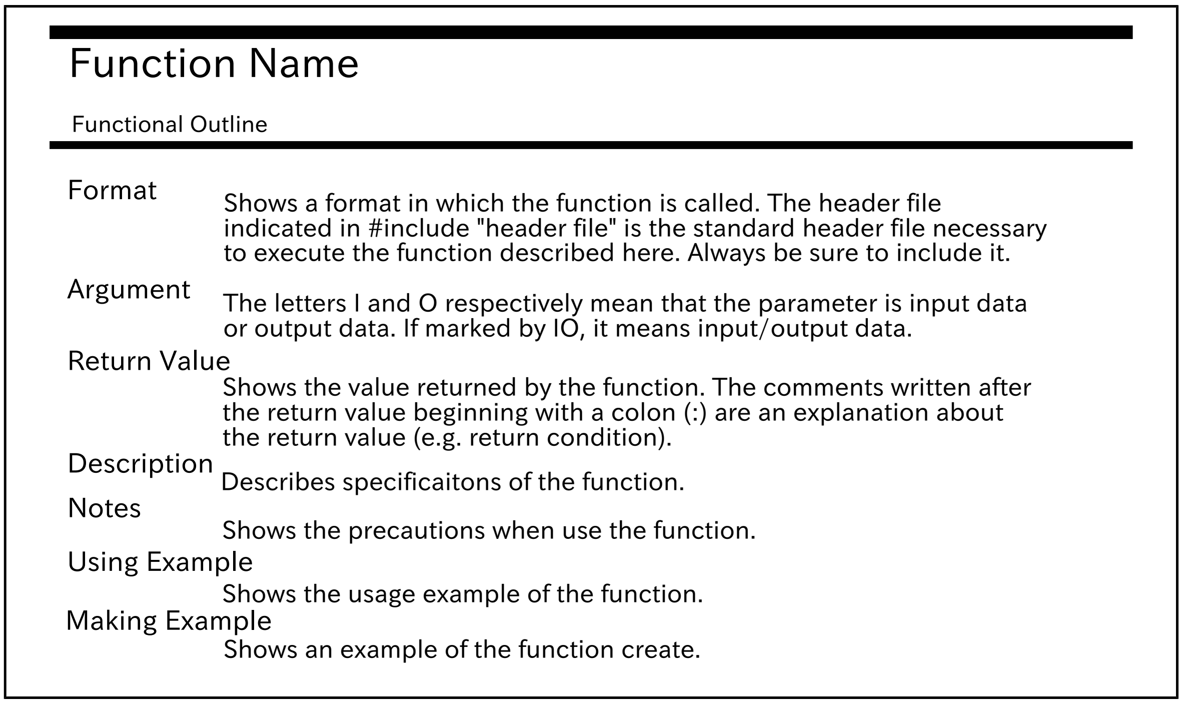
This enum is used to indicate the result of the disk operations performed by the driver functions.

Table 4.4. DRESULT Value

DRESULT	Value	Explanation
TFAT_RES_OK	0	Function execution is successful
TFAT_RES_ERROR	1	Error occurred during function execution
TFAT_RES_WRPRT	2	Disk is write protected
TFAT_RES_NOTRDY	3	Disk drive is not initialized
TFAT_RES_PARERR	4	Invalid argument passed to the function

5. Library functions

This section shows the details of each function of the TFAT Library. The way to description of each function is as follows.



Function Name

Functional Outline

Format	Shows a format in which the function is called. The header file indicated in #include "header file" is the standard header file necessary to execute the function described here. Always be sure to include it.
Argument	The letters I and O respectively mean that the parameter is input data or output data. If marked by IO, it means input/output data.
Return Value	Shows the value returned by the function. The comments written after the return value beginning with a colon (:) are an explanation about the return value (e.g. return condition).
Description	Describes specificaitons of the function.
Notes	Shows the precautions when use the function.
Using Example	Shows the usage example of the function.
Making Example	Shows an example of the function create.

Figure 5.1. Description of Library Function Details

R_tfat_f_mount

— Register/Unregister a work area

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_mount (
                uint8_t Drive ,
                FATFS *FileSystemObject );
```

Argument

Argument	I/O	Explanation
Drive	I	Logical drive number to register/unregister the work area. The value should always be 0.
FileSystemObject	I	Pointer to the work area (file system object) to be registered.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

Description

This function registers/unregisters the work area (i.e. a file system object) on the memory media. This work area must be registered before using any other file operation functions. To discard the registered work area, specify a NULL to the FileSystemObject.

This function only initializes the given work area and registers its address to the internal table, any access to the disk I/O layer does not occur. The volume mount process is performed on first file access after R_tfat_f_mount or media change.

Before using any file function, a work area must be given to the logical drive with R_tfat_f_mount function. All file functions can work only after this procedure.

Using Example

```
FATFS fatfs;
FRESULT res;
res = R_tfat_f_mount(0, &fatfs);
```

R_tfat_f_open

— Open/Create a file

Format

#include "r_tfat_lib.h"

```
FRESULT R_tfat_f_open (
                FIL *FileObject ,
                const uint8_t *FileName ,
                uint8_t ModeFlags );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to the file object structure to be created. After the R_tfat_f_open function succeeds, the file can be accessed with this file object structure until it is closed.
FileName	I	Pointer to a null-terminated string that specifies the name of the file to be created or opened. Specification about this is below. The details is shown in after. "<the physical drive number>:<file name>"
ModeFlags	I	Specifies the type of access and open method for the file. Please refer the following table for details.

ModeFlags are specified as a combination of the following macros.

Value	Explanation
TFAT_FA_READ	Specifies read access to the object. Data can be read from the file. Combine with TFAT_FA_WRITE for read-write access.
TFAT_FA_WRITE	Specifies write access to the object. Data can be written to the file. Combine with TFAT_FA_READ for read-write access.
TFAT_FA_OPEN_EXISTING	Opens the file. The function fails if the file does not exist. (Default)
TFAT_FA_OPEN_ALWAYS	Opens the file, if it is exists. If not, the function creates a new file.
TFAT_FA_CREATE_NEW	Creates a new file. The function fails if the file already exists.
TFAT_FA_CREATE_ALWAYS	Creates a new file. If the file exists, it is truncated and overwritten.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

If return value is TFAT_FR_DENIED, The required access was denied due to one of the following reasons.

- Write mode open of a read-only file.
- File (or directory) could not be created as a read-only file (or a directory) with the same name is already existing.
- File cannot be created as the directory table or disk is full.

Description

This function creates a file object to be used for accessing the file.

This file object is then used for the subsequent file operations through the library functions.

Using Example

```
FIL file;                       // Open a new file file.txt in the write mode
FRESULT res;
res = R_tfat_f_open(&file, "0:file.txt", TFAT_FA_CREATE_ALWAYS | TFAT_FA_WRITE);
```

R_tfat_f_close

— Close a file

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_close (
                FIL *FileObject );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to the open file object structure to be closed.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

Description

This function closes an open file object. If any data has been written to the file, the cached information of the file is written back to the disk. After the function succeeds, the file object is no longer valid and can be discarded. If the file object has been opened in read-only mode, it may be discarded without calling this function.

Using Example

```
FIL file;                                         // File structure variable
FRESULT res;                                      // Result
res = R_tfat_f_open(&file, ”0:file.txt”, TFAT_FA_WRITE); // Open file
// File operations
res = R_tfat_f_close(&file);                      // Close file
```

R_tfat_f_read

— Read file

Format

#include "r_tfat_lib.h"

```
FRESULT R_tfat_f_read (
                FIL *FileObject ,
                void *Buffer ,
                uint16_t BytesToRead ,
                uint16_t *BytesRead );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to open file object structure.
Buffer	O	Pointer to the buffer to store read data.
BytesToRead	I	Number of bytes to read in range of uint16_t.
BytesRead	O	Pointer to the uint16_t variable to return number of bytes read.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

If return value is TFAT_FR_DENIED, The required access was denied as the file has been opened in non-read mode.

Description

The R_tfat_f_read function reads data from a file.

The file pointer of the file object increments with the number of bytes read. After the function succeeds, *BytesRead should be checked to detect the end of file. In case of *BytesRead is less than BytesToRead, it means the R/W pointer reached end of file during read operation.

Using Example

```
FIL file;                                         // File structure
uint8_t Buff[128];                                // Data Buffer
FRESULT res;                                      // Result
uint16_t file_read_cnt;                           // Data read count
R_tfat_f_open(&file, ”0:file.txt”, TFAT_FA_READ);  // Open file in read mode
//Read 128 bytes
res = R_tfat_f_read(&file, Buff, 128, &file_read_cnt);
R_tfat_f_close(&file);                            // Close file
```

R_tfat_f_write

— Write file

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_write (
                FIL *FileObject ,
                void *Buffer ,
                uint16_t BytesToWrite ,
                uint16_t *BytesWritten );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to open file object structure.
Buffer	I	Pointer to the buffer to be written.
BytesToWrite	I	Specifies number of bytes to written in range of uint16_t.
BytesWritten	O	Pointer to the uint16_t variable to return number of bytes written.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

If return value is TFAT_FR_DENIED, The required access was denied as the file has been opened in non-write mode.

Description

The R_tfat_f_write function writes data to a file.

The read/write pointer in the file object is incremented with the number of bytes written. After the function succeeds, *ByteWritten should be checked to detect if the disk full. In case if *BytesWritten is less than BytesToWrite, it means the disk got full during write operation.

Using Example

```
FIL file;                                   // File structure
uint8_t Buff[] = "Hello";                   // Data Buffer
FRESULT res;                                // Result
uint16_t file_write_cnt;                    // Data write count
R_tfat_f_open(&file, "0:file.txt", TFAT_FA_WRITE);  // Open file in write mode
//Write data
res = R_tfat_f_write(&file,Buff,(sizeof(Buff)-1),&file_write_cnt);
R_tfat_f_close(&file);                      // Close file
```

R_tfat_f_lseek

— Move read/write pointer, Expand file size

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_lseek (
                FIL *FileObject ,
                uint32_t Offset );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to open file object structure
Offset	I	Offset of number of bytes from the start of the file

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

Description

This function moves the file read/write pointer of an open file object. It can also be used to extend the file size.

The R_tfat_f_lseek function moves the file R/W pointer of an open file. The offset can be specified only from top of the file. When an offset above the file size is specified in write mode, the file size is extended to the offset and the data in the extended area is undefined. This is suitable to create a large file quickly, for fast write operation. After the R_tfat_f_lseek function succeeds, member fptr in the file object should be checked in order to make sure the R/W pointer has been moved correctly. In case if the fptr is less than expected value, any of the followings may have occurred.

- In read-only mode, the Offset was truncated to the file size.
- The drive got full during the file extension process.
- There is an error in the FAT structure.

Using Example

```
FRESULT res;                      // Result
// Move to offset of 5000 from top of the file.
res = R_tfat_f_lseek(&file, 5000);

// Forward 3000 bytes
res = R_tfat_f_lseek(&file, file.fptr + 3000);
```

R_tfat_f_truncate

— Truncate file size

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_truncate (
                FIL *FileObject );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to the open file object to be truncated.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

If return value is TFAT_FR_DENIED, The required access was denied as the file has been opened in read-only mode.

Description

The R_tfat_f_truncate function truncates the file size.

The R_tfat_f_truncate function truncates the file size until the current file R/W point. When the file R/W pointer is already at the end of the file, there is no effect. The current R/W point can be manipulated to the desired position of truncation with the help of R_tfat_f_lseek function.

Using Example

```
FRESULT res;                                 // Result
// To truncate a file at 5000 bytes from the start.
R_tfat_f_lseek(&file, 5000);                 // Offset of 5000 from top of the file.
res = R_tfat_f_truncate(&file);              // Truncate the file
```

R_tfat_f_sync
— Flush cached data

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_sync (
                FIL *FileObject );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to the open file object to be flushed.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

Description

This function performs the same process as R_tfat_f_close function but the file is left opened and can continue read/write/seek operations to the file. This is suitable for applications that open files for a long time in writing mode, such as data logger. Performing R_tfat_f_sync periodically or immediately after R_tfat_f_write can minimize risk of data loss due to a sudden power shutdown or unintentional disk removal.

Using Example

```
FRESULT res;                      // Result
// To synchronize the file at any point.
res = R_tfat_f_sync(&file);
```

R_tfat_f_opendir

— Open a directory

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_opendir (
                DIR *DirObject ,
                const uint8_t *DirName );
```

Argument

Argument	I/O	Explanation
DirObject	I/O	Pointer to the blank directory object to be created.
DirName	I	Pointer to the null-terminated string that specifies the directory name to be opened.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

Description

The R_tfat_f_opendir function opens a directory.

This function opens an existing directory and creates a directory object. This directory object can now be used with R_tfat_f_readdir. The directory object structure can be discarded at any time without any procedure.

Using Example

```
DIR dir;
FRESULT res;                          // Result
res = R_tfat_f_opendir(&dir, "0:abc");   // Open a directory named "abc"
```

R_tfat_f_readdir
— Read a directory item

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_readdir (
                DIR *DirObject ,
                FILINFO *FileInfo );
```

Argument

Argument	I/O	Explanation
DirObject	I/O	Pointer to the open directory structure.
FileInfo	O	Pointer to the file information structure to store the read items.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

Description

The R_tfat_f_readdir function reads directory entries.

This function reads the entries in a directory. All items in the directory can be read by calling R_tfat_f_readdir function repeatedly. When all directory items have been read and there are no further items to read, the function returns a null string into fname[] member without any error. For details of the file information, refer to the FILINFO structure.

Please note that this function cannot be used without assigning a valid directory to the DIR* pointer variable through the R_tfat_f_opendir function.

Using Example

```
DIR dir;
FILINFO finfo;
FRESULT res;                        // Result
R_tfat_f_opendir(&dir, "0:abc");      // Open a directory named "abc"
res = R_tfat_f_readdir(&dir, &finfo);     // Read directory "abc" into finfo
```

R_tfat_f_getfree

— Get free clusters

Format

#include "r_tfat_lib.h"

```
FRESULT R_tfat_f_getfree (
                const uint8_t *Path ,
                uint32_t *Clusters ,
                FATFS **FileSystemObject );
```

Argument

Argument	I/O	Explanation
Path	I	Pointer to the null-terminated string that specifies the root directory of the logical drive. Always specify a null-string.
Clusters	O	Pointer to the variable to store the number of free clusters.
FileSystemObject	O	Pointer to a pointer storing the corresponding file system object.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

Description

The R_tfat_f_getfree function gets the number of free clusters. The member csize in the file system object gives the number of sectors per cluster, so that the free space in unit of sector can be calculated.

Using Example

```
FATFS *fs;
uint32_t clust, tot_disk_space, free_disk_space;
FRESULT res;                              // Result
res = R_tfat_f_getfree("", &clust, &fs);     // Get free clusters
// Total disk space available
tot_disk_space = (fs->max_clust - 2) * fs->csize / 2;
// Free disk space available
free_disk_space = clust * fs->csize / 2;
```

R_tfat_f_stat
— Get file status

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_stat (
                const uint8_t *FileName ,
                FILINFO *FileInfo );
```

Argument

Argument	I/O	Explanation
FileName	I	Pointer to the null-terminated string that specifies the file or directory to get its information.
FileInfo	O	Pointer to blank FILINFO structure to store information.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

Description

The function R_tfat_f_stat gets the information of a file or directory.

The information is stored in the structure pointed by FileInfo structure pointer.

Using Example

```
FILINFO filinfo;
FRESULT res;                                 // Result
res = R_tfat_f_stat (”file.txt”, &filinfo);  // Retrieve file information
```

R_tfat_f_mkdir

— Create a directory

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_mkdir (
                const uint8_t *DirName );
```

Argument

Argument	I/O	Explanation
DirName	I	Pointer to the null-terminated string that specifies the name of the directory to be created.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

If return value is TFAT_FR_DENIED, The directory could not be created due to the directory table or disk is full.

Description

The function R_tfat_f_mkdir creates a new directory.

Using Example

```
FRESULT res;                    // Result
res = R_tfat_f_mkdir("abc");    // Make a directory "abc"
```

R_tfat_f_unlink
— Remove a file or directory

Format

#include "r_tfat_lib.h"

```
FRESULT R_tfat_f_unlink (
                const uint8_t *FileName );
```

Argument

Argument	I/O	Explanation
FileName	I	Pointer to the null-terminated string that specifies the file (or directory) to be removed.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

If return value is TFAT_FR_DENIED, The function was denied due to either of the following reasons.

- The file (or directory) has a read-only attribute.
- The directory is not empty.

Description

The function R_tfat_f_unlink removes / deletes a file (or a directory).

Using Example

```
FRESULT res;                        // Result
res = R_tfat_f_unlink(”file.txt”);  // Remove file ”file.txt”
```

R_tfat_f_chmod
— Change attribute

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_chmod (
                const uint8_t *FileName ,
                uint8_t Attribute ,
                uint8_t AttributeMask );
```

Argument

Argument	I/O	Explanation
FileName	I	Pointer to the null-terminated string that specifies the file (or directory).
Attribute	I	Attribute flags to be set in one or more combination of the flags in the following table. The specified flags are set and others are cleared.
AttributeMask	I	Attribute mask that specifies which attribute is changed. The specified attributes are set or cleared.

Attribute, AttributeMask are specified as a combination of the following macros.

macros	attribute
TFAT_AM_RDO	Read only
TFAT_AM_HID	Hidden
TFAT_AM_SYS	System
TFAT_AM_ARC	Archive

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

Description

The R_tfat_f_chmod function changes the attributes of a file (or directory).

Using Example

```
FRESULT res;                    // Result
// Set read-only flag, clear archive flag and retain others
res = R_tfat_f_chmod(”file.txt”, TFAT_AM_RDO, TFAT_AM_RDO | TFAT_AM_ARC);
```

R_tfat_f_utime
— Change timestamp

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_utime (
                const uint8_t *FileName ,
                const FILINFO *TimeDate );
```

Argument

Argument	I/O	Explanation
FileName	I	Pointer to the null-terminated string that specifies the file (or directory).
TimeDate	I	Pointer to the file information structure that has the timestamp stored in the members fdate and ftime. The other structure member values are irrelevant.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

Description

The R_tfat_f_utime function changes the timestamp of a file (or directory). The date and time values need to be set in the corresponding members of the FILINFO structure to be passed as the argument.

Using Example

```
FILINFO filinfo;
FRESULT res;                               // Result
// please refer sec 3.4 FILINFO structure explanation
filinfo.fdate = 0x3A4D;                    // Date set to 13th Feb 2009
filinfo.ftime = 0x7322;                    // Time set to 14:25:04
res = R_tfat_f_utime(”file.txt”, &filinfo);  // Update the time
```

R_tfat_f_rename

— Rename/Move a file or directory

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_rename (
                const uint8_t *OldName ,
                const uint8_t *NewName );
```

Argument

Argument	I/O	Explanation
OldName	I	Pointer to the null-terminated string that specifies the file (or directory) to be renamed.And this function can move a file in other directory.
NewName	I	Pointer to a null-terminated string specifies the new file/directory name without drive number. Existing filename cannot be specified.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, "FRESULT - File function return code" .

If return value is TFAT_FR_DENIED, The new name for the file could not be created due to some reason.

Description

The R_tfat_f_rename function renames a file (or directory).

Using Example

```
FRESULT res;                      // Result
// Rename file or directory
res = R_tfat_f_rename("oldname.txt", "newname.txt");

// Rename and move file to other directory simultaneously
res = R_tfat_f_rename("oldname.txt", "dir1/newname.txt");
```

R_tfat_f_forward
— Forward file data to the stream directly

Format

```
#include "r_tfat_lib.h"

FRESULT R_tfat_f_forward (
                FIL *FileObject ,
                uint16_t (*Func)(const uint8_t*,uint16_t) ,
                uint16_t ByteToFwd ,
                uint16_t *ByteFwd );
```

Argument

Argument	I/O	Explanation
FileObject	I/O	Pointer to open file object structure
Func	I	Pointer to the user-defined data streaming function. For details, refer to the sample code.
ByteToFwd	I	Number of bytes to forward in range of uint16_t.
ByteFwd	O	Pointer to uint16_t variable to return number of bytes forwarded.

Return Value

Return value	Explanation
FRESULT	Result of the function execution as explained in Section 4.1, “FRESULT - File function return code” .

Description

R_tfat_f_forward reads the data from the file and forward it to the outgoing stream without data buffer.

Please refer to R_tfat_outstream for the details of the user difinition function to specify in the second argment.

Using Example

```
FRESULT res;                    // Result
FIL file;                       // File structure
uint16_t file_forwarded_cnt;    // Data forwarded count

R_tfat_f_open(&file, ”0:file.txt”, TFAT_FA_READ); // Open file in read mode

// File output stream
res = R_tfat_f_forward(&file,R_tfat_outstream,10,&file_forwarded_cnt);
```

6. Memory driver interface

This section explains the details of the memory driver interface functions. The prototype of these functions along with the processing necessary in the implementation of each function has been explained. The implementation of these functions should be written by the user such that they can be used in conjunction with the memory driver available with the user. These are user-defined functions.

R_tfat_disk_initialize

— Initialize disk drive

Format

```
#include "r_tfat_lib.h"

DSTATUS R_tfat_disk_initialize (
                uint8_t pdrv);
```

Argument

Argument	I/O	Explanation
Drive	I	Specifies the initialize drive number.

Return Value

Return Value	Explanation
DSTATUS	Status of the disk after function execution

Description

This function should consist of the code to initialize the disk drive. This function

- enables power to the memory card.
- checks if the card is compatible with available voltage.
- identifies the card type.

R_tfat_disk_read
— Read sectors

Format

```
#include "r_tfat_lib.h"

DRESULT R_tfat_disk_read (
                uint8_t Drive ,
                uint8_t *Buffer ,
                uint32_t SectorNumber ,
                uint8_t SectorCount );
```

Argument

Arguments	I/O	Explanation
Drive	I	Specifies the physical drive number. The value will always be 0.
Buffer	O	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.
SectorNumber	I	Specifies the start sector number in logical block address (LBA).
SectorCount	I	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

Return Value	Explanation
DRESULT	Result of the function execution as explained in Section 4.4, "Enum - DRESULT"

Description

This function should consist of the code to read data from the disk drive. The details about the data location to be read are given by the arguments.

R_tfat_disk_write
— Write sectors

Format

```
#include "r_tfat_lib.h"

DRESULT R_tfat_disk_write (
                uint8_t Drive ,
                uint8_t *Buffer ,
                uint32_t SectorNumber ,
                uint8_t SectorCount );
```

Argument

Arguments	I/O	Explanation
Drive	I	Specifies the physical drive number. The value will always be 0.
Buffer	I	Pointer to the data to be written.
SectorNumber	I	Specifies the start sector number in logical block address (LBA).
SectorCount	I	Specifies number of sectors to read. The value can be 1 to 255.

Return Value

Return Value	Explanation
DRESULT	Result of the function execution as explained in Section 4.4, "Enum - DRESULT"

Description

This function should consist of the code to write data to the disk drive. The details about the data to be written are given by the arguments.

R_tfat_disk_ioctl
— Control device dependent features

Format

```
#include "r_tfat_lib.h"

DRESULT R_tfat_disk_ioctl (
                uint8_t Drive ,
                uint8_t Command ,
                void *Buffer );
```

Argument

Arguments	I/O	Explanation
Drive	I	Specifies the physical drive number. The value will always be 0.
Command	I	Specifies the command code. The command code will always be 0.
Buffer	I	Pointer should always be a NULL pointer.

Return Value

Return Value	Explanation
DRESULT	Result of the function execution as explained in Section 4.4, "Enum - DRESULT"

Description

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.

R_tfat_disk_status

— Get disk status

Format

```
#include "r_tfat_lib.h"

DSTATUS R_tfat_disk_status (
                uint8_t Drive );
```

Argument

Arguments	I/O	Explanation
Drive	I	Specifies the physical drive number. The value will always be 0.

Return Value

Return Value	Explanation
DSTATUS	Status of the disk after function execution

Description

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 4.3, “Macros for Disk Status” . The disk status can be returned by updating the return value with the macros related to disk status.

R_tfat_get_fattime

— Get current time

Format

```
#include "r_tfat_lib.h"

uint32_t R_tfat_get_fattime (
               void  );
```

Argument

None

Return Value

uint32_t Please refer the following table for explanation of return value

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

Description

This function returns the current date and time.

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

R_tfat_outstream
— forward to stream

Format

```
uint16_t R_tfat_outstream (
                 uint8_t *ptr ,
                 uint16_t cnt );
```

Argument

Arguments	I/O	Explanation
ptr	I	When "cnt" is 0,this argument is invalid value. When "cnt" is not 0,this argument means top address of transmit data.
cnt	I	When "cnt" is 0,This argument means state confirmation of stream When "cnt" is not 0,This argument means forward data size.

Return Value

Return Value	Explanation
uint16_t	(When "cnt" is 0)0: Busy, 1: Ready (When "cnt" is not 0)Forwarded data size

Explanation

When user executes R_tfat_f_forward function,this function is necessary.The function name does not need to "R_tfat_outstream".

Open Source FAT File System [M3S-TFAT-Tiny]
User's Manual

Publication Date	1 Apr 2014	Rev.3.01
	1 Apr 2014	Rev.3.01
Publisher	Renesas Electronics Corporation 1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa 211-8668, Japan	

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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, 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 906, 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 Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Open Source FAT File System
[M3S-TFAT-Tiny]
User's Manual

Renesas Electronics Corporation

R20UW0078EJ0301