RX Family

Open Source FAT File System M3S-TFAT-Tiny Module

Firmware Integration Technology

Introduction

This application note describes the RX Family Open Source FAT Filesystem M3S-TFAT-Tiny which uses Firmware Integration Technology (FIT). In this document, this module is referred to as the TFAT FIT module.

TFAT is no relation to Microsoft Transaction-Safe FAT File System (TFAT).

The LFN extension on the FAT filesystem was a patent of Microsoft Corporation. But the related patents all have expired and using the LFN feature has got free for any projects.

Also, enabling LFN will increase the module size depending on the code page selected. Please refer to the FatFs Web page described in 1.1 What is FatFs?.

Target Device

- RX Family

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details of the confirmed operation contents of each compiler, refer to “6.1 Confirmed Operation Environment".

Contents

1. Outline

TFAT FIT is FAT File system software that concerned about low-memory usage.

The TFAT FIT was made based on FatFs.

1.1 What is FatFs?

FatFs is the File system module for the small embedded system. FatFs is developed by ChaN Software. FatFs is provided as non-payment for embedded system. Please refer to the Website below for more details about ChaN Software and FatFs.

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

In addition, you can download documents related to FatFs (FatFs application note, configuration options, and APIs detail description etc.) from below URL. Please download appreciate FatFs version that matches the "Base program" item of Table 1.1 Specification of TFAT FIT.

http://elm-chan.org/fsw/ff/archives.html

Or, they are included in this module downloaded from TFAT FIT's Web page.

https://www.renesas.com/us/en/products/software-tools/software-os-middleware-driver/file-system/m3s-tfat-tiny-for-rx.html

1.2 Specification of TFAT

1.2.1 Specification of TFAT

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

Table 1.1 Specification of TFAT FIT

Item	Specifications
Base program	Fatfs (R0.13c)
Supported FAT type	FAT16, FAT32
Filename support	8.3 format (8 lettered filename & 3 lettered extension), LFN: long file name (max 255 characters)
Filesystem format function	None
Number of drives supported	10
Logical sector size	512byte
RTOS	FreeRTOS, RI600V4 (μITRON for RX family)
Memory storage	SD card, USB memory, USB-mini (Note) Note: USB-mini is not supported RTOS.

1.2.2 Structure of software stack

Following are structure of software stack of the TFAT FIT.

This module is work with the M3S-TFAT-Tiny Memory Driver Interface module FIT (TFAT driver FIT) and various device drivers.

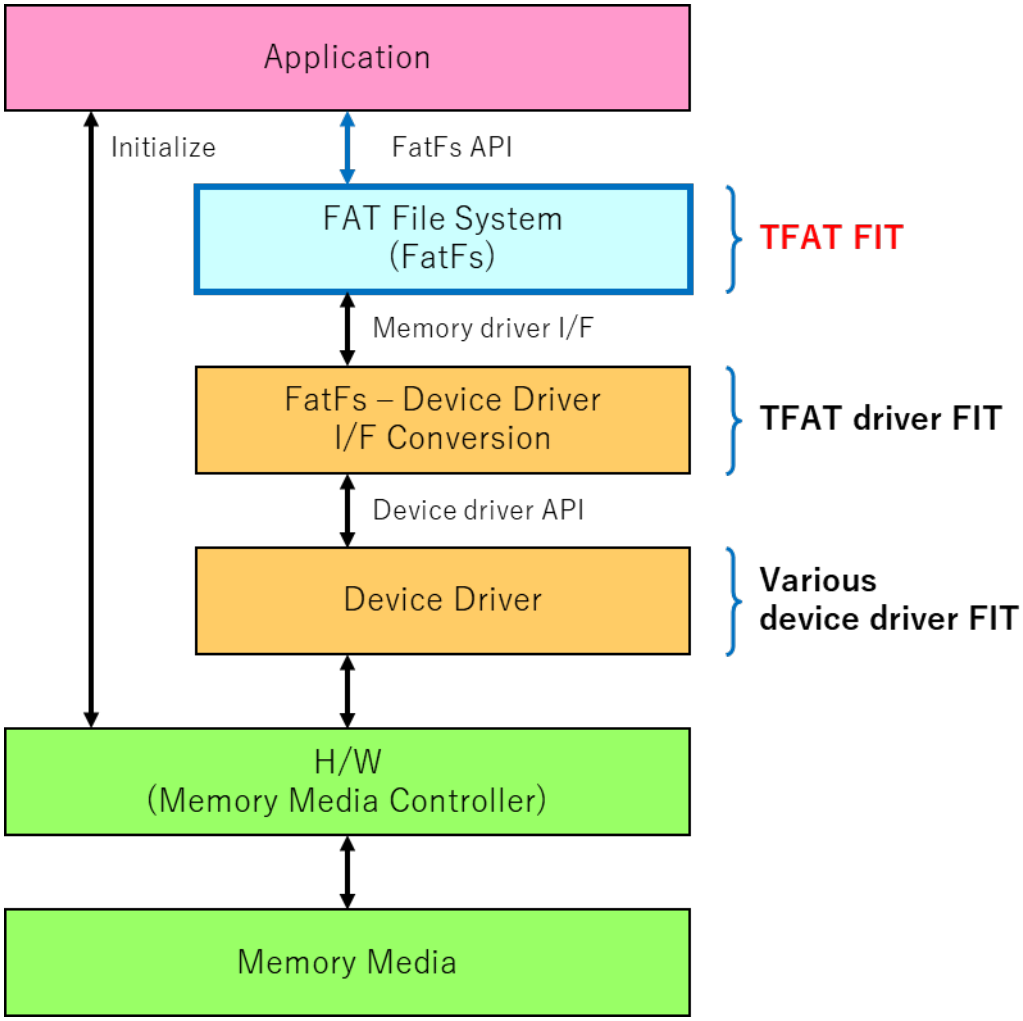

Figure 1.1 structure of software stack of the TFAT FIT

1.2.3 Configuration options of FatFs

FatFs has multiple configuration options (file name, etc.) defined in “ffconf.h”. You can customize available APIs and function feature by changing these define value.

We confirmed operation of the following configuration options. Please refer to 1.3 Overview of API function for available APIs by confirmed options.

Table 1.2 Confirmed configuration options

Configuratiton Option	Define Value	Remark
FF_USE_LFN	0, 1, 2, 3	File name setting
FF_FS_REENTRANT	0, 1	RTOS setting (automatically determined define value)
FF_SYNC_t	SemaphoreHandle_t, ID	RTOS setting (automatically determined define value)
FF_FS_TIMEOUT	1000	RTOS setting (automatically determined define value) You can change any value
Other	Default	-

1.2.4 Supported RTOS

This module supported FreeRTOS and RI600V4.

(1) Outline of operation when using RTOS

Considering multi-task operation of RTOS, each API of FatFs has a mechanism to ensure reentrancy (exclusive control) for the volume, except for a part. Specifically, each API uses the mutex with timeout feature of RTOS. Mutexes are deleted / generated by the f_mount function. Then, mutexes are taken immediately after starting the execution of each API and released immediately before is completed.

While taking the mutex by an API, when an API is executed to same volume by another task etc., the API executed later is shifted to the state of waiting release for mutex and waits specified time which defined by the FatFs configuration option “FF_FS_TIMEOUT”. Then, the return value of the API executed later returns “FR_OK” if the mutex is released within the specified time (“FF_FS_TIMEOUT”) and returns “FR_TIMEOUT” if it is not released.

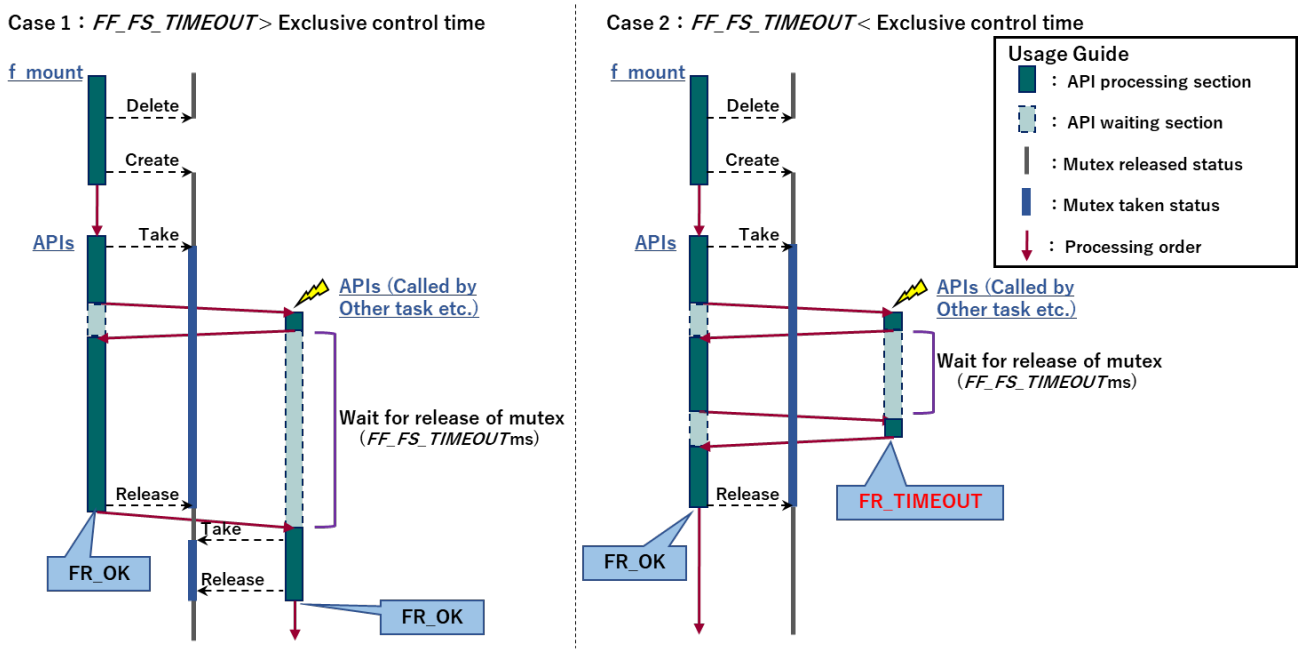

Figure 1.2 Exclusive control for a volume

(2) Condition of exclusive control

The condition which each API applies to exclusive control is different by FatFs configuration options and the accessed volume.

Table 1.3 Condition of exclusive control (✓: Exclusive control, - : No exclusive control)

Configuratiton Option	Access to Same Volume	Access to Different Volume
FF_FS_REENTRANT = 0 && FF_USE_LFN != 1	-	✓
FF_FS_REENTRANT = 0 && FF_USE_LFN = 1	-	-
FF_FS_REENTRANT = 1 && FF_USE_LFN != 1	✓ (Note)	✓
FF_FS_REENTRANT = 1 && FF_USE_LFN = 1	-	-

Note: Excluding f_mount, f_disk, and f_mkfs function. These functions are always not reentrancy for access to same volume. Therefore, you must prevent another task etc. from accessing to the same volume by application programs.

In addition, FatFs does not take attention to the reentrancy of the memory driver interface function and its lower layer. Therefore, it is necessary to implement these low-level I/O functions separately to ensure reentrancy.

(3) FatFs configuration options when using RTOS

TFAT FIT automatically sets the following FatFs configuration options (in "ffconf.h") to ensure reentrancy when using RTOS.

- FF_FS_REENTRANT: Determined whether reentrancy or not. You cannot change this value.
- FF_SYNC_t: The depended type on RTOS to identify a mutex. You cannot change this value.
- FF_FS_TIMEOUT: Timeout time (milli sec.) for waiting a mutex. You **can** change this value.

1.2.5 Conditions of use

This module is based on open source FatFs. Therefore, you must comply with conditions of the FatFs license clauses.

http://elm-chan.org/fsw/ff/doc/appnote.html#license

Other conditions of use are governed by the end user license agreement of FIT.

https://www.renesas.com/us/en/common/disclaimers/disclaimer002.html

For questions related to FatFs contents, consider using the FatFs user forum.

http://elm-chan.org/fsw/ff/bd/

1.2.6 Version compatibility of TFAT FIT

TFAT FIT Rev.4.00 is not compatible with previous TFAT FIT. This is because the specifications of FatFs API functions are different.

1.3 Overview of API function

The following API functions are used in TFAT FIT. "✓" indicates that API is available by default configuration options or is confirmed operation.

Table 1.4 API function

Name of API function	Description	Default	Confirmed Operation
File Access			
f_open()	Open/Create a file	✓	✓
f_close()	Close an open file	✓	✓
f_read()	Read data from the file	✓	✓
f_write()	Write data to the file	✓	✓
f_lseek()	Move read/write pointer, Expand size	✓	✓
f_truncate()	Truncate file size	✓	✓
f_sync()	Flush cached data	✓	✓
f_forward()	Forward data to the stream		
f_expand()	Allocate a contiguous block to the file		
f_gets()	Read a string		
f_putc()	Write a character		
f_puts()	Write a string		
f_printf()	Write a formatted string		
f_tell()	Get current read/write pointer	✓	✓
f_eof()	Test for end-of-file	✓	✓
f_size()	Get size	✓	✓
f_error()	Test for an error	✓	✓
Directory Access			
f_opendir()	Open a directory	✓	✓
f_closedir()	Close an open directory	✓	✓
f_readdir()	Read a directory item	✓	✓
f_findfirst()	Open a directory and read the first item matched		
f_findnext()	Read a next item matched		
File and Directory Management			
f_stat()	Check existence of a file or sub-directory	✓	✓
f_unlink()	Remove a file or sub-directory	✓	✓
f_rename()	Rename/Move a file or sub-directory	✓	✓
f_chmod()	Change attribute of a file or sub-directory		
f_utime()	Change timestamp of a file or sub-directory		
f_mkdir()	Create a sub-directory	✓	✓
f_chdir()	Change current directory		
f_chdrive()	Change current drive		
f_getcwd()	Retrieve the current directory and drive		
Volume Management and System Configuration			
f_mount()	Register/Unregister the work area of the volume	✓	✓
f_mkfs()	Create a FAT volume on the logical drive		
f_fdisk()	Create partitions on the physical drive		
f_getfree()	Get free space on the volume	✓	✓
f_getlabel()	Get volume label		
f_setlabel()	Set volume label		
f_setcp()	Set active code page		

1.4 Overview of Memory Driver Interface Function

API functions of TFAT FIT use the following low-level functions. Do not call these functions from the application programs.

Table 1.5 Memory Driver Interface Function

Name of low-level function	Description
disk_initialize()	Initialize device
disk_status()	Get device status
disk_read()	Read data
disk_write()	Write data
disk_ioctl()	Control device dependent functions
get_fattime()	Get current time

2. API Information

2.1 Hardware Requirements

None

2.2 Software Requirements

This driver is dependent on the following FIT module:

- Renesas Board Support Package (r_bsp) Rev.5.50 or later
- M3S-TFAT-Tiny Memory Driver Interface Firmware Integration Technology (r_tdat_driver) Rev.2.00 or later
- CMT Module Using Firmware Integration Technology (r_cmt) Rev.4.31 or later
- System Timer Module Firmware Integration Technology (r_sys_time) Rev.1.01 or later

2.3 Supported Toolchains

The operation of the ADC FIT module has been confirmed with the toolchain listed in 6.1, Operation Confirmation Environment.

2.4 Limitations

1) When using TFAT with Real Time OS, TFAT APIs are used in the same thread.
2) The f_mount, f_disk, and f_mkfs function is not reentrancy for same volume. Please realize exclusive control by application programs when using it.
3) TFAT is using the following standard function

   memset, memcmp, memcpy

   malloc (Note), free (Note)

   Note: When using FF_USE_LFN == 3

2.5 Header Files

All API and memory driver interface calls are accessed by including file " ff.h and diskio.h".

2.6 Configuration Overview

All configurable options that can be set at build time are located in the file “r_tfat_rx_config.h” (Note).

FatFs Configuration options are located in “ffconf.h”.

Note: There is no options on the TFAT FIT Rev.4.00.

2.7 Code Sizes

The sizes of ROM, RAM and maximum stack usage associated with this module are listed below. Information is listed for a single representative device of the RX100 Series, RX200 Series, and RX600 Series, respectively.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.7, Configuration Overview.

The values in the table below are confirmed under the following conditions.

Module Revision: r_tfat_rx rev.4.00

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.02.00

(The option of "-lang = c99" is added to the default settings of the integrated development environment.)

GCC for Renesas RX 8.3.0.201904

(The option of "-std=gnu99" is added to the default settings of the integrated development environment.)

IAR C/C++ Compiler for Renesas RX version 4.13.1

(The default settings of the integrated development environment.)

Configuration Options: Default settings

ROM, RAM and Stack Code Sizes

Device	Category	Memory Used		
		Renesas Compiler	GCC	IAR Compiler
RX113	ROM (Note)	7,046 bytes	4,096 bytes	8,239 bytes
	RAM (Note)	32 bytes	36 bytes	32 bytes
	STACK (Note)	144 bytes	-	288 bytes
RX231	ROM (Note)	6,788 bytes	13,840 bytes	7,920 bytes
	RAM (Note)	30 bytes	32 bytes	30 bytes
	STACK (Note)	144 bytes	-	288 bytes
RX65N	ROM (Note)	7,069 bytes	14,231 bytes	8,284 bytes
	RAM (Note)	32 bytes	36 bytes	32 bytes
	STACK (Note)	144 bytes	-	288 bytes

Note The sizes of maximum usage stack of Interrupts functions is included.

2.8 Type definition of TFAT FIT

The type definition used by TFAT is shown below (In case of C99. FIT uses C99.).

Table 2.1 Type of TFAT FIT

Datatype	Typedef
unsigned char	BYTE
unsigned char	DSTATUS
uint16_t	WORD
uint16_t	WCHAR
unsigned int	UINT
uint32_t	DWORD
uint64_t	QWORD

Also, TCHAR changes depending on the setting, and is defined as follows.

- WCHAR if FF_USE_LFN && FF_LFN_UNICODE = 1
- Char when FF_USE_LFN && FF_LFN_UNICODE = 2
- DWORD when FF_USE_LFN && FF_LFN_UNICODE = 3
- Char other than the above

2.9 TFAT FIT structure

This section describes in detail the structures used by TFAT FIT.

2.9.1 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 f_mount function. Application program must not modify any member in this structure.

The details of the members of FATFS structure are shown below.

```
typedef struct {
      BYTE  fs_type;          /* Filesystem type (0:not mounted) */
      BYTE  pdrv;             /* Associated physical drive */
      BYTE  n_fats;                 /* Number of FATs (1 or 2) */
      BYTE  wflag;                  /* win[] flag (b0:dirty) */
      BYTE  fsi_flag;         /* FSINFO flags (b7:disabled, b0:dirty) */
      WORD  id;                     /* Volume mount ID */
      WORD  n_rootdir;        /* Number of root directory entries (FAT12/16) */
      WORD  csize;            /* Cluster size [sectors] */
#if FF_MAX_SS != FF_MIN_SS
      WORD  ssize;            /* Sector size (512, 1024, 2048 or 4096) */
#endif
#if FF_USE_LFN
      WCHAR*      lfnbuf;     /* LFN working buffer */
#endif
#if FF_FS_EXFAT
      BYTE* dirbuf;           /* Directory entry block scratchpad buffer for
                                 exFAT */
#endif
#if FF_FS_REENTRANT
      FF_SYNC_t   sobj;       /* Identifier of sync object */
#endif
#if !FF_FS_READONLY
      DWORD last_clst;        /* Last allocated cluster */
      DWORD free_clst;        /* Number of free clusters */
#endif
#if FF_FS_RPATH
      DWORD cdir;             /* Current directory start cluster (0:root) */
#if FF_FS_EXFAT
      DWORD cdc_scl;          /* Containing directory start cluster (invalid
                                 when cdir is 0) */
      DWORD cdc_size;         /* b31-b8:Size of containing directory, b7-b0:
                                 Chain status */
      DWORD cdc_ofs;          /* Offset in the containing directory (invalid
                                 when cdir is 0) */
#endif
#endif
      DWORD n_fatent;         /* Number of FAT entries (number of clusters+2)*/
      DWORD fsize;            /* Size of an FAT [sectors] */
      DWORD volbase;          /* Volume base sector */
      DWORD fatbase;          /* FAT base sector */
      DWORD dirbase;          /* Root directory base sector/cluster */
      DWORD database;         /* Data base sector */
#if FF_FS_EXFAT
      DWORD bitbase;          /* Allocation bitmap base sector */
#endif
      DWORD winsect;          /* Current sector appearing in the win[] */
      BYTE  win[FF_MAX_SS];   /* Disk access window for Directory, FAT (and file
                                 data at tiny cfg) */
```

```
} FATFS;
```

2.9.2 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 f_opendir() or f_readdir() functions. Application program must not modify any member in this structure.

The details of the members of FATFS structure are shown below.

```
typedef struct {
      FFOBJID     obj;        /* Object identifier */
      DWORD       dptr;       /* Current read/write offset */
      DWORD       clust;      /* Current cluster */
      DWORD       sect;       /* Current sector (0:Read operation has
                              terminated) */
      BYTE*       dir;        /* Pointer to the directory item in the win[] */
      BYTE        fn[12];     /* SFN (in/out) {body[8],ext[3],status[1]} */
#if FF_USE_LFN
      DWORD       blk_ofs;    /* Offset of current entry block being processed
                               (0xFFFFFFFF:Invalid) */
#endif
#if FF_USE_FIND
      const TCHAR* pat;       /* Pointer to the name matching pattern */
#endif
} DIR;
```

2.9.3 FIL - File object structure

The FIL structure (file object) holds state of a file. It is created by f_open function and discarded by f_close function. Application program must not modify any member in this structure except for “cltbl”.

```
typedef struct {
      FFOBJID     obj;        /* Object identifier (must be the 1st member
                              to detect invalid object pointer) */
      BYTE        flag;       /* File status flags */
      BYTE        err;        /* Abort flag (error code) */
      FSIZE_t     fptr;       /* File read/write pointer (Zeroed on file open)*/
      DWORD       clust;      /* Current cluster of fpter (invalid when fptr
                               is 0) */
      DWORD       sect;       /* Sector number appearing in buf[] (0:invalid) */
#if !FF_FS_READONLY
      DWORD       dir_sect;   /* Sector number containing the directory entry
                               (not used at exFAT) */
      BYTE*       dir_ptr;    /* Pointer to the directory entry in the win[]
                               (not used at exFAT) */
#endif
#if FF_USE_FASTSEEK
      DWORD*      cltbl;      /* Pointer to the cluster link map table
                              (nulled on open, set by application) */
#endif
#if !FF_FS_TINY
      BYTE        buf[FF_MAX_SS];   /* File private data read/write window */
#endif
} FIL;
```

2.9.4 FILINFO - File status structure

The FILINFO structure holds the file information returned by f_stat() and f_readdir() functions.

```
typedef struct {
      FSIZE_t     fsize;                        /* File size */
      WORD        fdate;                        /* Modified date */
      WORD        ftime;                        /* Modified time */
      BYTE        fattrib;                      /* File attribute */
#if FF_USE_LFN
      TCHAR       altname[FF_SFN_BUF + 1];      /* Altenative file name */
      TCHAR       fname[FF_LFN_BUF + 1];        /* Primary file name */
#else
      TCHAR       fname[12 + 1]                 /* File name */
#endif
} FILINFO;
```

2.9.5 FFOBJID - Object ID and assignment information structure

The FFOBJID structure holds the object ID and assignment information.

```
typedef struct {
      FATFS*      fs;         /* Pointer to the hosting volume of this object */
      WORD        id;               /* Hosting volume mount ID */
      BYTE        attr;             /* Object attribute */
      BYTE        stat;             /* Object chain status (b1-0: =0:not
                                     contiguous,=2:contiguous, =3:fragmented
                                     in this session,
                                    b2:sub-directory stretched) */
      DWORD       sclust;           /* Object data start cluster (0:no cluster
                                    or root directory) */
      FSIZE_t     objsize;          /* Object size (valid when sclust != 0) */
#if FF_FS_EXFAT
      DWORD       n_cont;           /* Size of first fragment - 1 (valid when
                                    stat == 3) */
      DWORD       n_frag;           /* Size of last fragment needs to be written
                                    to FAT (valid when not zero) */
      DWORD       c_scl;            /* Containing directory start cluster (valid
                                    when sclust != 0) */
      DWORD       c_size;           /* b31-b8:Size of containing directory,
                                    b7-b0: Chain status
                                    (valid when c_scl != 0) */
      DWORD       c_ofs;            /* Offset in the containing directory
                                     (valid when file object and sclust != 0) */
#endif
#if FF_FS_LOCK
      UINT        lockid;           /* File lock ID origin from 1
                                    (index of file semaphore table Files[]) */
#endif
} FFOBJID;
```

2.10 TFAT FIT constant

This section describes in detail the constants used. The following constants are defined in ff.h.

2.10.1 FRESULT - API function return value

The return value of API function is defined as enum type.

```
typedef enum
{
      FR_OK = 0,                    /* (0) Succeeded */
      FR_DISK_ERR,                  /* (1) A hard error occurred in the low
                                       level disk I/O layer */
      FR_INT_ERR,                   /* (2) Assertion failed */
      FR_NOT_READY,                 /* (3) The physical drive cannot work */
      FR_NO_FILE,                   /* (4) Could not find the file */
      FR_NO_PATH,                   /* (5) Could not find the path */
      FR_INVALID_NAME,              /* (6) The path name format is invalid */
      FR_DENIED,                    /* (7) Access denied due to prohibited
                                       access or directory full */
      FR_EXIST,                     /* (8) Access denied due to prohibited
                                       access */
      FR_INVALID_OBJECT,            /* (9) The file/directory object is
                                       invalid */
      FR_WRITE_PROTECTED,           /* (10) The physical drive is write
protected */
      FR_INVALID_DRIVE,             /* (11) The logical drive number is
                                       invalid */
      FR_NOT_ENABLED,               /* (12) The volume has no work area */
      FR_NO_FILESYSTEM,             /* (13) There is no valid FAT volume */
      FR_MKFS_ABORTED,              /* (14) The f_mkfs() aborted due to any
                                       problem */
      FR_TIMEOUT,                   /* (15) Could not get a grant to access
                                       the volume within defined period */
      FR_LOCKED,                    /* (16) The operation is rejected according
                                       to the file sharing policy */
      FR_NOT_ENOUGH_CORE,           /* (17) LFN working buffer could not be
                                       allocated */
      FR_TOO_MANY_OPEN_FILES,       /* (18) Number of open files > FF_FS_LOCK */
      FR_INVALID_PARAMETER          /* (19) Given parameter is invalid */
}FRESULT;
```

2.10.2 File attribute information

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

Table 2.2 Attribute information macros

Name	Value	Explanation
AM_RDO	0x01	When this flag is set the applicable file(or directory) is read only.
AM_HID	0x02	When this flag is set the applicable file(or directory) is hidden.
AM_SYS	0x04	When this flag is set the applicable file(or directory) is system file.
AM_DIR	0x10	When this flag is set the applicable file is directory.
AM_ARC	0x20	When this flag is set the applicable file(or directory) is Archive.

2.10.3 Macros for Disk Status

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

Table 2.3 Macros for Disk Status

Name	Value	Explanation
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 disk_initialize function, and cleared on: success of disk_initialize function.
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.
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 STA_NODISK is set.

2.10.4 Return value of memory driver interface function

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

Table 2.4 DRESULT Value

Name	Value	Explanation
RES_OK	0	Function execution is successful.
RES_ERROR	1	Error occurred during function execution.
RES_WRPRT	2	Disk is write protected.
RES_NOTRDY	3	Disk drive is not initialized.
RES_PARERR	4	Invalid argument passed to the function.

2.10.5 Format Options

The following macros show the contents of the format options. It is used as the second argument of f_mkfs function. In addition, the actual formatted FAT type is affected on cluster size (the third argument of f_mkfs).

For details, refer to f_mkfs.

Table 2.5 File attribute information macro

Name	Value
FM_FAT	0x01
FM_FAT32	0x02
FM_EXFAT	0x04
FM_ANY	0x07
FM_SFD	0x08

2.11 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends using “Smart Configurator” described in (1) or (3). However, “Smart Configurator” only supports some RX devices. Please use the methods of (2) or (4) for unsupported RX devices.

(1) Adding the FIT module to your project using “Smart Configurator” in e^2 studio
By using the “Smart Configurator” in e^2 studio, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User Guide: e^2 studio (R20AN0451)” for details.

(2) Adding the FIT module to your project using “FIT Configurator” in e^2 studio
By using the “FIT Configurator” in e^2 studio, the FIT module is automatically added to your project. Refer to “Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using “Smart Configurator” on CS+
By using the “Smart Configurator Standalone version” in CS+, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User Guide: CS+ (R20AN0470)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

3. API functions

About detail API's descriptions, please refer to FatFs documents indicated 1.1 What is FatFs?.

4. Memory driver interface function

Memory driver interface functions are prototyped in “diskio.h” or “ff.h”. The actual function entity exists in “diskio.c” or “r_tfat_drv_if.c” of TFAT driver FIT.
For details on these functions, refer to the application note (r20an0335xxxxxx) of TFAT driver FIT.

5. Sample program

5.1 Outline

The sample program is e^2 studio project that works at the board (hereafter referred to as "CPU board") shown in 6.1 Operation Confirmation Environment The sample program prepares for two kinds of following projects

- Sample program using the SD mode SD memory card driver
- Sample program using the USB driver

In addition, above sample programs combine with RTOS (FreeRTOS and RI600V4).

— Document No.: R01AN3852
— Document Title: RX Family SDHI Module Using Firmware Integration Technology: Application note

— Document No.: R01AN4233
— Document Title: RX Family SD mode SD memory Card Firmware Integration Technology: Application note

— Document No.: R01AN2025
— Document Title: USB Basic Host and Peripheral Driver Firmware Integration Technology: Application note

— Document No.: R01AN2026
— Document Title: USB Host Mass Storage Class Driver (HMSC) Firmware Integration Technology: Application note

— Document No.: R01AN2166
— Document Title: USB Basic Mini Host and Peripheral Driver (USB Mini Firmware) Using Firmware Integration Technology: Application note

— Document No.: R01AN2169
— Document Title: USB Host Mass Storage Class Driver (HMSC) for USB Mini Firmware Integration Technology: Application note

5.2 Sample software execution

5.2.1 The sample program with the SD mode SD memory card driver

When the program is run, a FAT filesystem work area is registered. A directory and a file are created on the memory media and text data of 2 KB is written to the file. The file is then closed. For confirmation of the data that is written, the file is opened again in the read mode. The entire contents of the file are read, and they are compared with the write buffer data in the program. Whether the contents of the data are matching or not is indicated on Debug Console (Renesas Virtual Debug Console) on e2studio.

Table 5.1 Explanation of Debug Console display

Characters	Explanation
Detected attached SD card.	Insertion of the SD card was detected.
Detected detached SD card.	The SD card removal was detected.
!!! Attach SD card. !!!!	Insert the SD card.
!!! Detach SD card. !!!!	Remove the SD card.
Start TFAT sample	Started sample program.
Finished TFAT sample	Finished sample program.
!!!!! TFAT error !!!!!	An error occurred.

The sample data for file read / write is stored in the “r_data_file.c”. The data is stored in an array of 2048 elements giving a total size of 2 KB (2048 Bytes). The data array consists of the text string "Renesas\n" written repeatedly. If required, the user can modify this array and the corresponding macro `FILESIZE`.

The primary functions used in sample program is the following. Regardless of using RTOS, the processing of sample is same. However, FreeRTOS and RI600V4 perform them as tasks.

Table 5.2 Primary functions of sample program

Processing	Function Name with None RTOS	Function Name with FreeRTOS	Function Name with RI600V4
Initialization	main()	main_task()	main_task()
Idle for device detection	idle_sdc_detection()	idle_detection_task()	idle_detection_task()
TFAT FIT API execution	tfat_sample()	tfat_sample_task() (Note)	tfat_sample_task() (Note)

Note: Exclusive control should be performed when this function is executed, but it is omitted for simplicity of sample program.

5.2.2 Flow (SD card driver)

Flow of a sample program with the SD card driver is shown below.

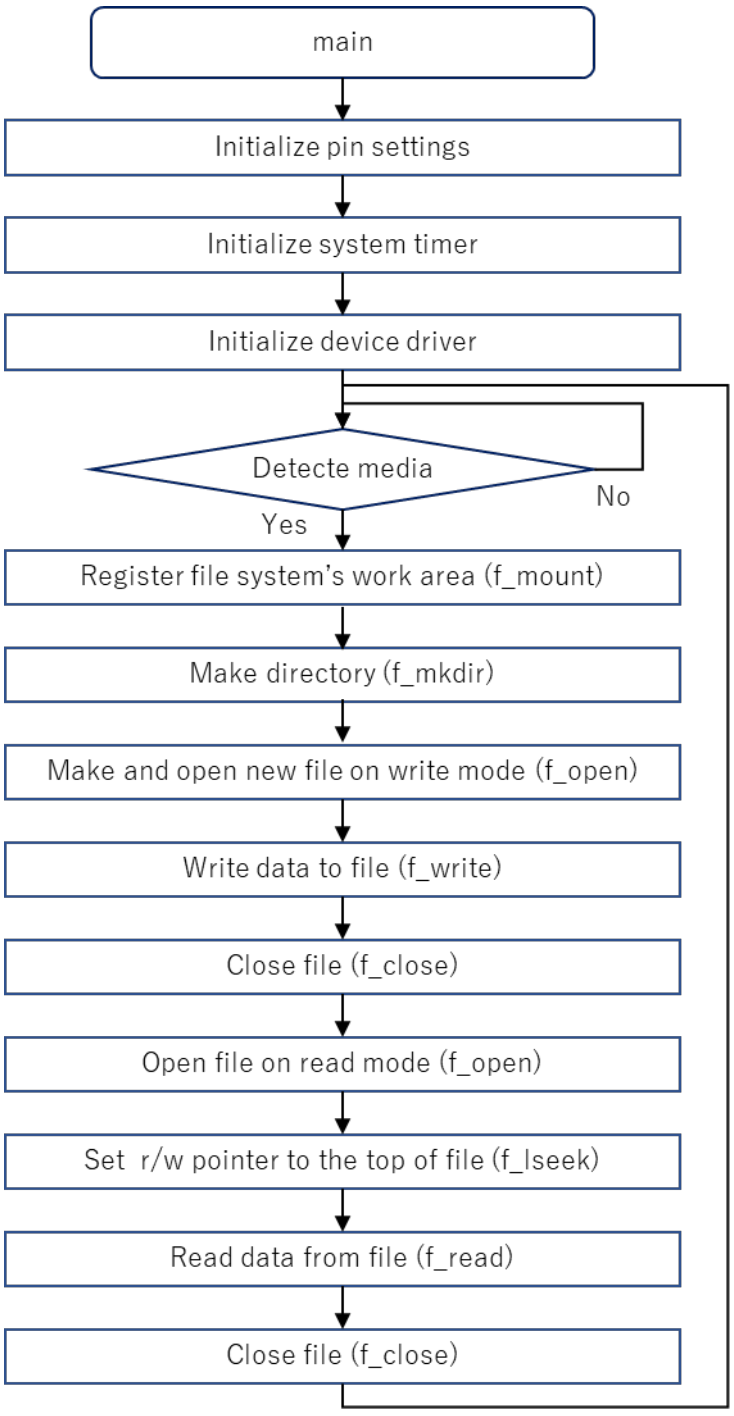

Figure 5.1 Flow of sample program with the SD card driver

5.2.3 The sample program with the USB driver

When the program is run, a FAT filesystem work area is registered. A directory and a file are created on the memory media and text data of 2 KB is written to the file. The file is then closed. For confirmation of the data that is written, the file is opened again in the read mode. The entire contents of the file are read, and they are compared with the write buffer data in the program. Whether the contents of the data are matching or not is indicated on Debug Console (Renesas Virtual Debug Console) on e2studio.

Table 5.3 Explanation of Debug Console display

Characters	Explanation
Detected attached USB memory.	Insertion of the USB memory was detected.
Detected detached USB memory.	The USB memory removal was detected.
!!! Attach USB memory. !!!!	Insert the USB memory.
!!! Detach USB memory. !!!!	Remove the USB memory.
Start TFAT sample	Started sample program.
Finished TFAT sample	Finished sample program.
!!!!! TFAT error !!!!!	An error occurred.

The sample data for file read / write is stored in the “r_data_file.c”. The data is stored in an array of 2048 elements giving a total size of 2 KB (2048 Bytes). The data array consists of the text string "Renesas," written repeatedly. If required, the user can modify this array and the corresponding macro `FILESIZE`.

The primary functions used in sample program is the following. Regardless of using RTOS, the processing of sample is same. However, FreeRTOS and RI600V4 perform them as tasks.

Table 5.4 Primary functions of sample program

Processing	Function Name with None RTOS	Function Name with FreeRTOS	Function Name with RI600V4
Initialization	main()	main_task()	main_task()
Idle for device detection	idle_sdc_detection()	idle_detection_task()	idle_detection_task()
TFAT FIT API execution	tfat_sample()	tfat_sample_task() (Note)	tfat_sample_task() (Note)

Note: Exclusive control should be performed when this function is executed, but it is omitted for simplicity of sample program.

5.2.4 Flow (USB driver)

Flow of a sample program with the USB driver is shown below.

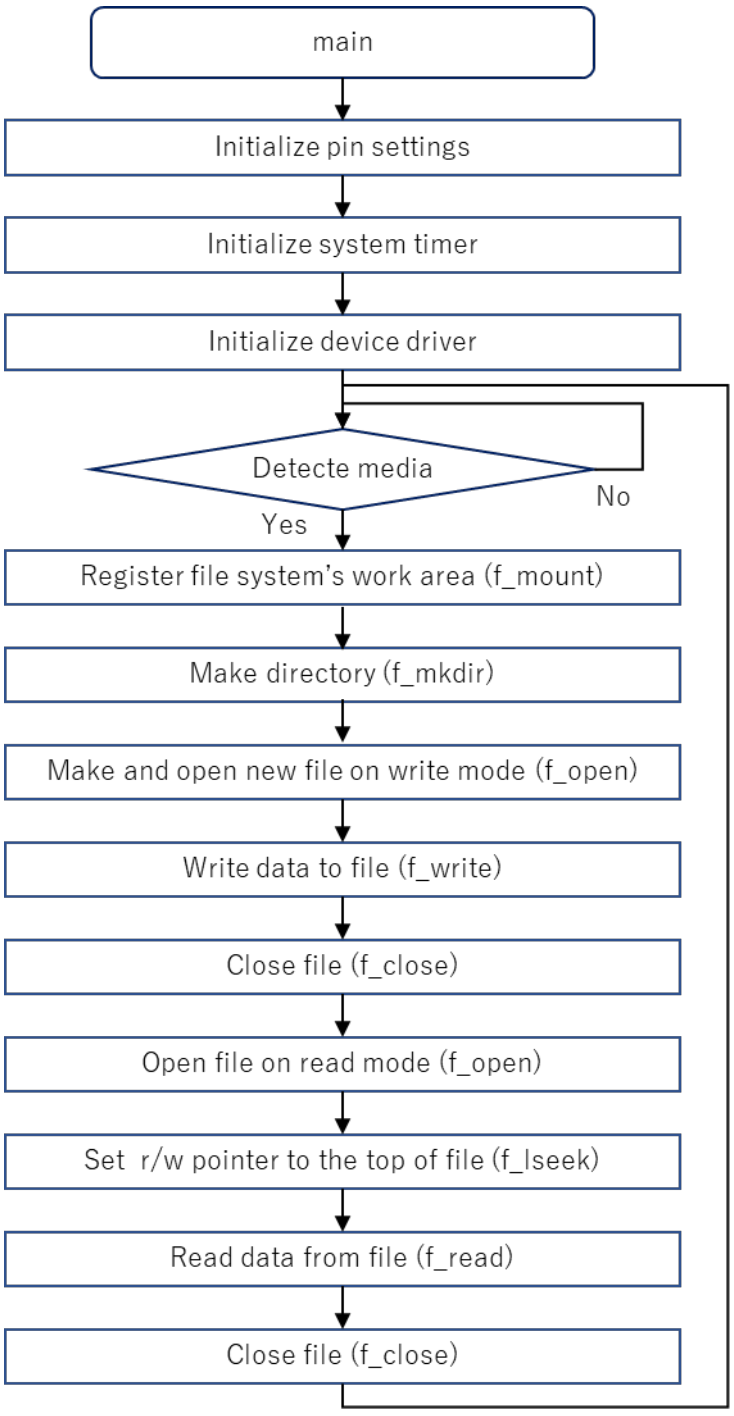

Figure 5.2 Flow of sample program with the SD card driver

6. Appendices

6.1 Confirmed Operation Environment

This section describes operation confirmation environment for TFAT FIT.

Table 6.1 Confirmed Operation Environment (Rev.3.04)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio V7.1.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.00.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Little endian
Revision of the module	Rev.3.04
Board used	Renesas Starter Kit for RX231 (product No.:R0K505231Sxxxxx) Renesas Starter Kit+ for RX65N-1MB (product No.:RTK500565NSxxxxxxx)
RTOS	None

Table 6.2 Confirmed Operation Environment (Rev.4.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.7.0 IAR Embedded Workbench for Renesas RX 4.13.1
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.02.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.201904 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.13.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.4.00
Board used	Renesas Starter Kit+ for RX72M (product No.:RTK5572Mxxxxxxxxxx)
RTOS	FreeRTOS V10.0.00 RI600V4 V1.06.00

6.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file "platform.h".

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note "Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)"

- Using e^2 studio:

  Application note "Adding Firmware Integration Technology Modules to Projects (R01AN1723)"

When using this FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note "Board Support Package Module Using Firmware Integration Technology (R01AN1685)".

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_sdc_sd_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

7. Reference Documents

User's Manual: Hardware
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools
RX Family C/C++ Compiler CC-RX User's Manual (R20UT3248)
The latest version can be downloaded from the Renesas Electronics website.

Related Technical Updates

This module reflects no technical updates.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Oct 08, 2010	-	First edition issued
1.01	Sep 01, 2012	-	RX210 correspondence
1.02	Nov 08, 2013	-	Changed document title Changed the structure of sections Added Fatfs copyright to library source
1.03	Nov 30, 2013	-	Changed the base version of the open source into V0.09b from V0.06.
3.00	Apr 01, 2014	-	FIT Module correspondence
3.01	Dec.28.2014	-	Corresponded to RX71M/RX113. Updated the xml file for FIT.
3.02	May.01.2015	-	Corresponded to RX231. Updated the xml file for FIT.
3.03	Oct.01.2016	-	Corresponded to RX family. Updated the xml file for FIT.
3.04	Nov.30.2018	-	Chapter 2.4, added limitation when using Real Time OS. Chapter 4 and 6 was added Updated the xml file for FIT.
4.00	Feb.25.2020	-	Updated the open source base version from V0.09b to V0.13c. Supported the following compilers. - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX Supported the following RTOS. - FreeRTOS - RI600V4 Removed "R_TFAT_" from the function names. Added Fatfs copyright notice to source.

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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. 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 the 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.

5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, 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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.