

RX62N

R20AN0075EJ0100

Rev.1.00

Web Server Implementation Using M3S-T4-Tiny

Apr 12, 2011

Introduction

This documentation provides the information required to implement a web server that runs on the Renesas Starter Kit+ for RX62N (RSK). This web server is an application that operates over TCP/IP, can be accessed from an ordinary web browser, and provides functions for transferring content stored on the web server to web browsers using TCP/IP.

This documentation describes the operation of a program that combines the following middleware products and middleware evaluation board and implements a web server. It also describes the RSK settings required to run that program. This documentation includes a sample program that implements a web server for the following environment.

Function	Middleware Product	Web Page* ¹
TCP/IP	M3S-T4-Tiny	http://www.renesas.com/mw/t4
File system	M3S-TFAT-Tiny	http://www.renesas.com/mw/tfat
MMC driver	M3S-MMCSLBR1M	http://www.renesas.com/mw/tfat http://www.renesas.com/mw/tfs
MMC extensions	Middleware Evaluation board* ²	http://www.renesas.com/mw/tfat http://www.renesas.com/mw/tfs http://www.renesas.com/mw/s2 http://www.renesas.com/mw/dtmf
USB driver	USB driver	http://www.renesas.com/driver/usb

Notes: 1. The items with multiple page references can be downloaded from the related middleware sites.

There are no differences between the downloadable application notes themselves.

2. The middleware evaluation board must be produced by the user based on these application notes.

Since each of these middleware packages are independent, they can be combined freely if the user implements interface programs. For example, the file system can be replaced by another file system, or the MMC driver can be replaced with a USB driver.

Furthermore, since the web server program itself contains no program code that depends on the microcontroller, it can be easily ported to another microcontroller simply by replacing the TCP/IP software stack with one for the other microcontroller.

Target Device

RX62N

Contents

1. Verifying Sample Program Operation	3
1.1 Hardware Preparation (In case MMC driver)	3
1.1.1 RSK/Middleware Evaluation Board Connection	3
1.1.2 Writing Data to the MMC.....	3
1.2 Hardware Preparation (In case USB driver)	3
1.2.1 Writing Data to the USB stick.....	3
1.3 RSK/PC Network Connection	3
1.4 Software Preparation.....	3
1.4.1 PC IP Address Settings	3
1.4.2 PC Web Browser Settings	3
1.5 Running the Sample Program.....	4
2. Web Server Specifications.....	5
2.1 Overview	5
2.1.1 System Structure	5
2.1.2 Software Structure	5
2.1.3 Performance Overview	6
2.1.4 Program Operation Overview	6
2.2 Configuration	7
2.3 API Reference	8
2.3.1 R_httpd.....	8
2.4 User-Defined Function Reference.....	9
2.4.1 Data Structures	9
2.4.2 change_dir	10
2.4.3 file_close	10
2.4.4 file_delete.....	11
2.4.5 file_open.....	11
2.4.6 file_read	12
2.4.7 file_rename	12
2.4.8 file_exist	13
2.4.9 file_write	13
2.4.10 get_file_info	14
2.4.11 get_file_list_info.....	14
2.4.12 get_file_size	15
2.4.13 make_dir.....	15
2.4.14 remove_dir	16
2.5 Notes	17
Website and Support.....	18
Revision Record	1
General Precautions in the Handling of MPU/MCU Products.....	2

1. Verifying Sample Program Operation

This document explains two versions of sample program which uses MMC or USB.

1.1 Hardware Preparation (In case MMC driver)

This sample program uses MMC as memory to hold the web server content. It uses the microcontroller's serial communications interface (SCI) for communication with the MMC. Note that it will be necessary, at the prototyping stage, for the user to purchase both the RSK and an MMC socket board to connect the SCI to the MMC. Renesas also provides an application note for the middleware board on which the application operates as reference information for connecting these units. These application notes should be used together.

1.1.1 RSK/Middleware Evaluation Board Connection

This sample program uses the RX62N SCI2 module for communication with the MMC. The table below lists the correspondences for connection between the RSK and the middleware evaluation board.

RX62N	RSK	Middleware Evaluation Board
TxD2-A	JA6-8	JA2-6
RxD2-A	JA6-7	JA2-8
SCK2-A	JA6-10	JA2-10
P90	JA20-4	JA1-15
P91	JA20-5	JA1-16
GND	Any ground	JA1-4
VDD	J13-1 or J13-2	JA1-3

1.1.2 Writing Data to the MMC

Write the files to be displayed by the web browser to the MMC that has been formatted for the FAT16 file system. This sample program supports arbitrary file transfers. It can also be used as the M3S-TFAT-Tiny (TFAT) file system.

1.2 Hardware Preparation (In case USB driver)

This sample program uses USB as memory to hold the web server content. It uses the USB for communication with the USB stick. Please connect USB stick to USB1 connector on RSK.

1.2.1 Writing Data to the USB stick

Write the files to be displayed by the web browser to the USB that has been formatted for the FAT16 file system. This sample program supports arbitrary file transfers. It can also be used as the M3S-TFAT-Tiny (TFAT) file system.

1.3 RSK/PC Network Connection

Use LAN cables to connect the RSK and the PC to the same network.

1.4 Software Preparation

1.4.1 PC IP Address Settings

The IP address 192.168.0.3 is allocated to the RSK in the sample program. Set the PC IP address to 192.168.0.xxx, where xxx is a value other than 3. Set the subnet mask to 255.255.255.0.

1.4.2 PC Web Browser Settings

If use of a proxy server is set up in the web browser, change the settings to not use proxy server.

1.5 Running the Sample Program

This sample program runs under the High-performance Embedded Workshop. Open the High-performance Embedded Workshop project by double clicking on "rx62n_webserver.hws". This sample program is set up in advance to operate with RSK and the E1 emulator. Connect RSK to the High-performance Embedded Workshop as directed. When connected, click Execute to run the program. Then, enter **http://192.168.0.3** at the web browser. A file listing of the root directory for the MMC connected to the RSK will be displayed.

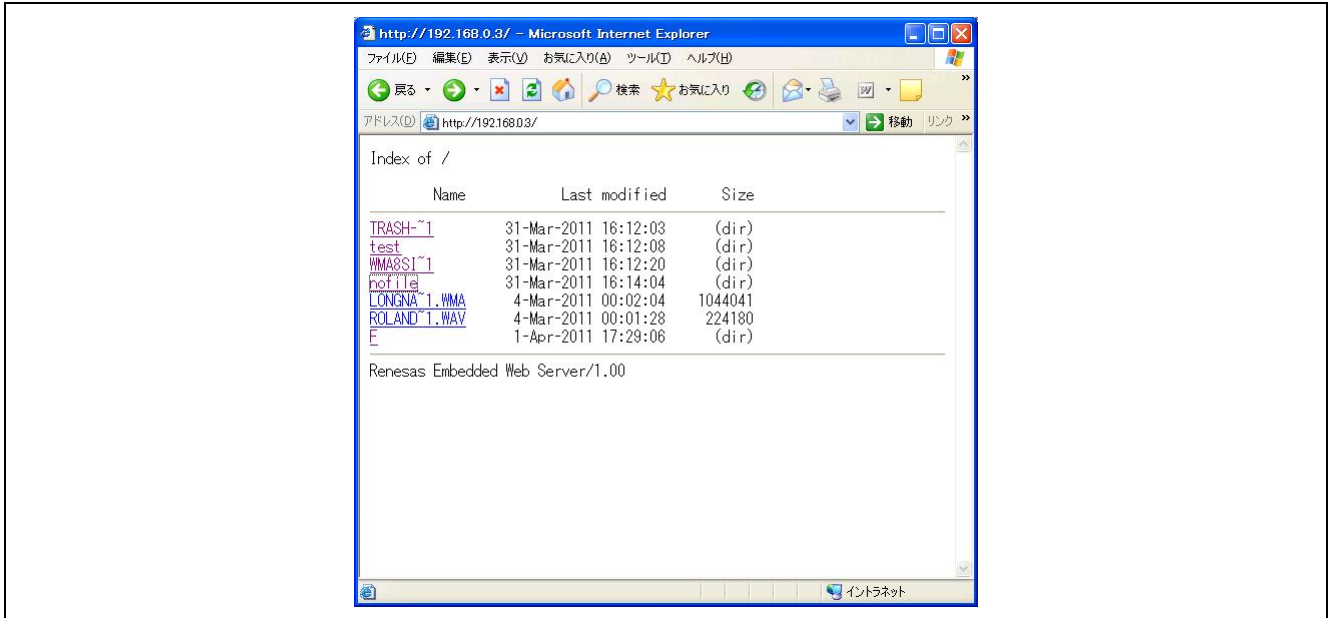


Figure 1 Operation Verification Screen (Root directory display)

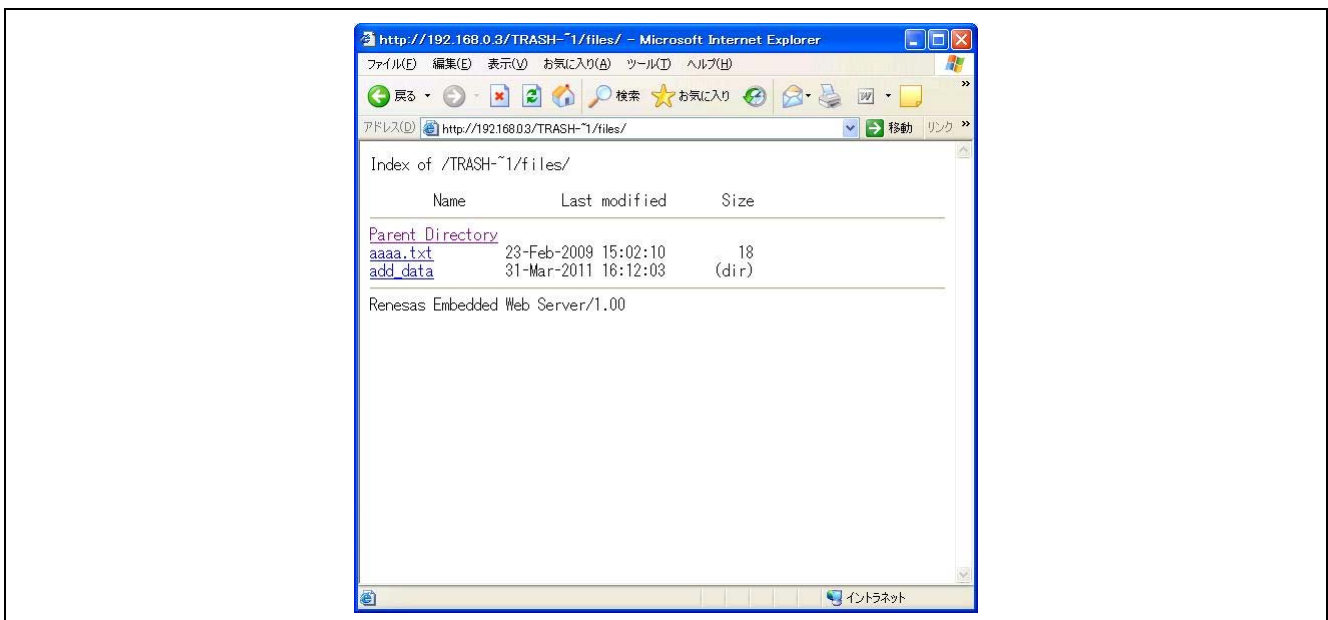


Figure 2 Operation Verification Screen (Display of directory other than root)

2. Web Server Specifications

2.1 Overview

2.1.1 System Structure

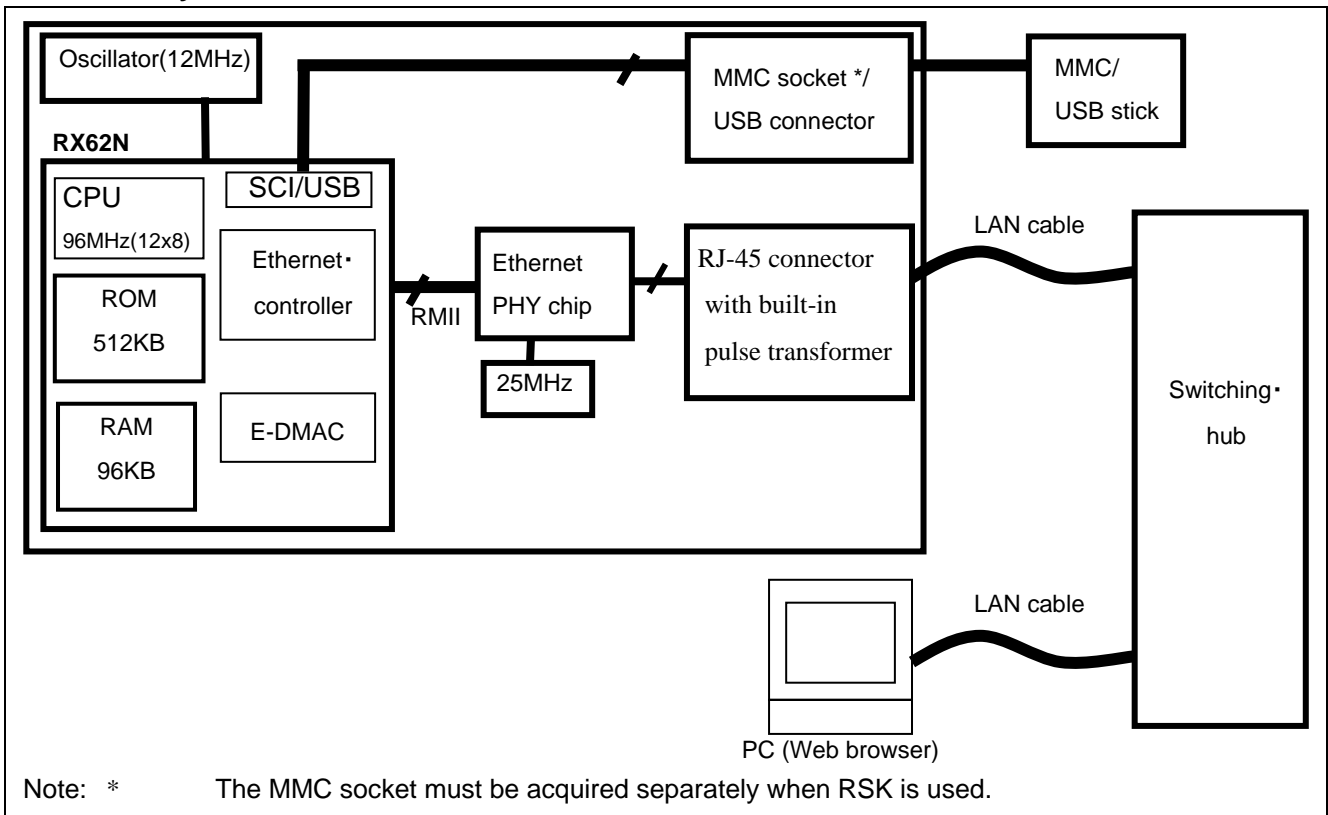


Figure 3 System Structure

2.1.2 Software Structure

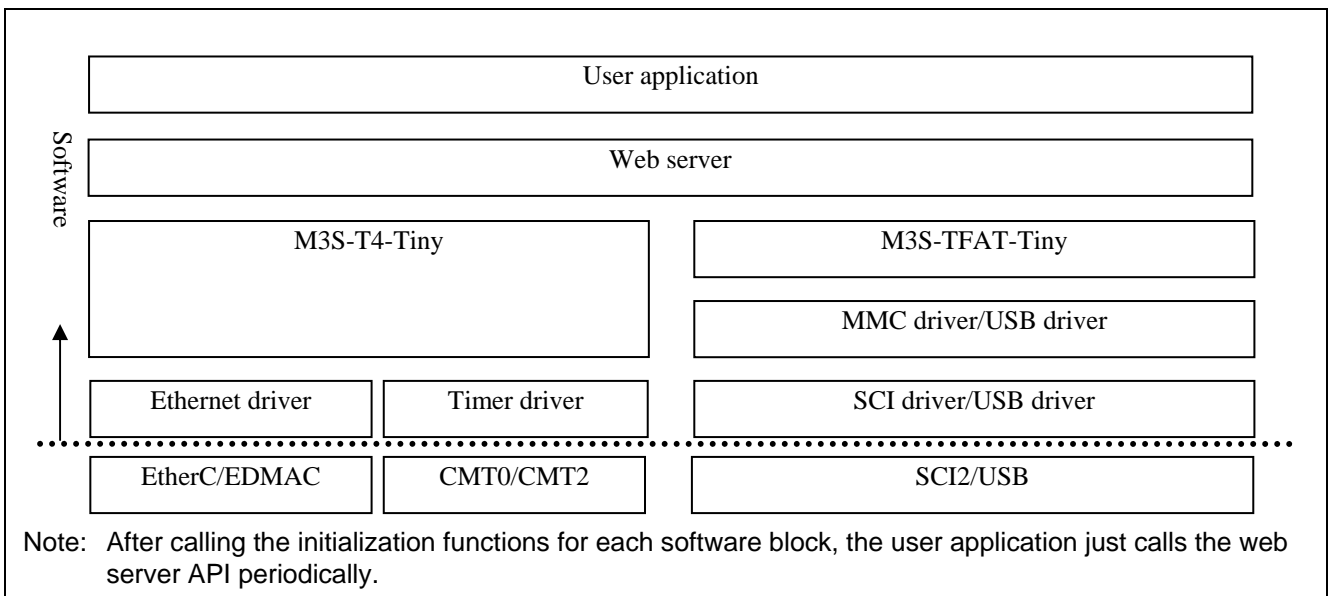


Figure 4 Software Structure

2.1.3 Performance Overview

This sample program implements a simple web server based on the HTTP 1.0 specifications. The purpose of this sample program is to serve as a base for developing a web server program for MS3-T4-Tiny (T4). This sample program does not include measures to protect against SYN-FLOOD and other attacks and also does not include security functions, and therefore is not appropriate for applications such as operating as a server that waits on the www port (80). It was developed assuming that it would only be used on local networks where malicious attackers are not present, such as office or factory networks. The file names that can be handled are also limited to short file names.

Furthermore, this sample program does not require any special memory other than file I/O and operates from microcontroller internal memory only. Although the processing performance is influenced by RAM capacity, it is coded so that this can be set flexibly in the program. This sample program sets its memory usage appropriately for the ROM/RAM capacity of the RX62N used. The ROM/RAM requirements for web server operation are listed below. The file of the measuring object is `r_httpd.c`, `r_file_driver.c`, `r_httpd_sample_main.c`, and `config_tcpudp.c`.

ROM	RAM	Stack
About 6.6 KB	About 36 KB	About 1.7 KB

2.1.4 Program Operation Overview

Compared to common web servers (such as Apache) used with the Internet, this sample program holds the implemented functions to an absolute minimum. Furthermore, it is implemented using non-blocking calls so that it is easy to use in embedded applications, and allows applications to implement a web server simply by calling the function `R_httpd()` periodically. The function `R_httpd()` monitors all communication endpoints (commonly called sockets) used in communication and switches to the connection wait state if a socket becomes disconnected. Communication processing is performed by the T4 API function `_process_tcpip()`, and this sample program calls this API function from the timer interrupt and Ethernet interrupt handlers. The `_process_tcpip()` function calls a callback function to report the completion of communication. The http data analysis and data generation processing are performed from this callback routine.

Since the interrupt processing time including that for activating `_process_tcpip()` can differ greatly depending on the send/receive driver's performance and the implementation of the callback routine, interrupts may be disabled or the application given higher priority.

The web server's behavior can be customized by changing definitions in the configuration file (`r_httpd_config.h`).

2.2 Configuration

The following settings can be made in the sample program by changing the macro definitions in the `r_httpd_config.h` file.

- Server header field: `HTTPD_VERSION_CODE`
The data stored in the server header field transmitted to the web browser during communication with the web browser can be specified.
- Root directory: `ROOT_DIR`
Which directory in external memory is seen as the root directory can be specified.
Examples: `#define ROOT_DIR ""`
`#define ROOT_DIR "user"`
`#define ROOT_DIR "user/root_dir"`
- Displaying or not displaying an index page: `INDEXES`
The behavior when a directory is specified from the web browser can be specified.
When set to 1, the directory contents are returned as the response.
When set to 0, the file specified as the `DEFAULT_FILE_NAME` is returned as the response.
- File returned as response when do not display index page is specified: `DEFAULT_FILE_NAME`
This file is the response when `INDEXES` is set to 0.
A "404 Not Found" response is returned if the specified file cannot be found.
- Corresponding content-type: `EXTENSION_TYPE_TABLE_LIST`
This is a list of the file extensions stored in external memory.
If a file with an extension not defined here is transferred, the system will respond to that file with the settings for the extension defined at the head of the list.
- New line code used for index page generation: `LF_CODE`
- Maximum number of clients that can be accepted at the same time: `TCP_CEP_NUM`
This value must be set to match the number of sockets defined in `config_tcpudp.c`.
- Maximum number of files that can be displayed on the index page: `MAX_FILE_LIST`
Set this value so that `BODY_BUF_SIZE` is not exceeded.
- Reception buffer size: `RCV_BUF_SIZE`
- Header file transmission buffer size: `HDR_BUF_SIZE`
- Body field transmission buffer size: `BODY_BUF_SIZE`

2.3 API Reference

2.3.1 R_httpd

Description

The application calls this function periodically. This function manages the sockets required for HTTP communication. This function only performs socket management; communication itself is performed automatically by T4 as driven by interrupts.

Usage

```
#include "r_httpd.h"  
void R_httpd(void);
```

Parameters

None

Return Value

None

Remark

None

2.4 User-Defined Function Reference

The web server calls these functions. The user must code the processing performed by these functions appropriately for the file system used. Also, the web server can use this data structure to acquire information from external memory. In this sample program, these functions are coded for the TFAT file system.

User-Defined Function Table

name	function	name	function
change_dir()	change current directory	file_write()	write file
file_close()	close file	get_file_info()	get file information
file_delete()	delete file	get_file_list_info()	get file list information
file_open()	open file	get_file_size()	get file size
file_read()	read file	make_dir()	make directory
file_rename()	rename file	remove_dir()	remove directory
file_exist()	confirm existing file		

The gray out function is not used this sample program.

2.4.1 Data Structures

Date Information Structure

```
typedef struct date_info_{
    uint16_t year;           // 2011, 2012, ...
    uint8_t month[4];       // Jan, Feb, Mar, ...
    uint8_t day;            // 1-31
    uint8_t day_of_the_week[4]; // Sun, Mon, Tus, ...
    uint16_t hour;          // 0-23
    uint16_t min;           // 0-59
    uint16_t sec;           // 0-59
}DATE_INFO;
```

File List Structure

```
typedef struct file_list_ {
    uint8_t file_name[13];
    uint32_t file_size;
    uint32_t file_attr;
    DATE_INFO date_info;
}FILE_LIST;
```

Macro definition

```
#define FILE_WRITE      (0x10)
#define FILE_READ      (0x01)
#define FILE_ATTR_RDO  0x01 /* Read only */
#define FILE_ATTR_HID  0x02 /* Hidden */
#define FILE_ATTR_SYS  0x04 /* System */
#define FILE_ATTR_VOL  0x08 /* Volume label */
#define FILE_ATTR_DIR  0x10 /* Directory */
#define FILE_ATTR_ARC  0x20 /* Archive */
```

2.4.2 change_dir

Description

This function sets current directory using specified argument. The argument specifies directory path in full path. Information of current directory is managed in each communication endpoint.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t change_dir(uint8_t *dir_path);
```

Parameters

dir_path	Input	Pointer to directory path
----------	-------	---------------------------

Return Value

-1	No directory to change
0	Normal completions

Remark

There are two cases. The argument "dir_path" has '/' termination and does not have. Please adjust for user file system.

2.4.3 file_close

Description

This function closes the file corresponding to the ID specified by the argument and discards the file management information.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t file_close(int32_t file_id);
```

Parameters

file_id	Input	ID value of the file to close
---------	-------	-------------------------------

Return Value

-1	Error
0	Normal completion

Remark

None

2.4.4 file_delete

Description

This function deletes the file corresponding to the ID specified by the argument. The specification of file is full path from root directory.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t file_delete(uint8_t *file_path);
```

Parameters

file_path	Input	Pointer to file path to delete
-----------	-------	--------------------------------

Return Value

-1	Error
0	Normal completion

Remark

None

2.4.5 file_open

Description

This function opens the file specified in its argument in exclusive read mode and saves file management information independently. It also specifies an ID value for this file management information as the return value so that the web server can reference the saved file management information by ID. The saved file management information must be saved until this ID value is passed to the file close function.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t file_open(uint8_t *file_path, uint8_t mode_flag);
```

Parameters

file_path	Input	Pointer to file path to open
mode_flag	Input	Mode value of file open (FILE_WRITE or FILE_READ)

Return Value

-1	Error
0 and positive integer	The ID value for the opened file.

Remark

The file opened state must be maintained until the corresponding ID value is passed to the file close function.

2.4.6 file_read

Description

This function reads the file corresponding to the ID value passed as an argument and advances the file pointer by the amount read. The file pointer is recorded in the file management information for each ID value and is maintained until the file close function is called.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t file_read(int32_t file_id, uint8_t *buf, int32_t read_size);
```

Parameters

file_id	Input	ID value of the file to read
buf	Output	Storage area for the file data read
read_size	Input	Size of the file to read

Return Value

-1	Error
0 and positive integer	Data size of receiving

Remark

None

2.4.7 file_rename

Description

This function renames the file specified first argument to second argument. These arguments are specified in full path from root directory.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t file_rename(uint8_t *old_name, uint8_t *new_name);
```

Parameters

old_name	Input	Pointer to target file name
new_name	Input	Pointer to after file name

Return Value

-1	Error
0	Normal completion

Remark

None

2.4.8 file_exist**Description**

This function verifies the file or directory existing. The argument is specified in full path from root directory.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t file_exist(uint8_t *file_path);
```

Parameters

file_path	Input	Pointer to file or directory path
-----------	-------	-----------------------------------

Return Value

-1	Not exist
0	Exist

Remark

None

2.4.9 file_write**Description**

This function writes the file corresponding to the ID value passed as an argument and advances the file pointer by the amount write. The file pointer is recorded in the file management information for each ID value and is maintained until the file close function is called.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t file_write(uint32_t file_id, uint8_t *buf, int32_t write_size);
```

Parameters

file_id	Input	ID value of the file to write
buf	Input	Storage area for the file data write
write_size	Input	Size of the file to write

Return Value

-1	Error
0	Normal completion

Remark

None

2.4.10 get_file_info**Description**

This function reads the file management information for the file corresponding to the ID value specified as an argument and writes the file date information to a date information structure.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t get_file_info(int32_t file_id, DATE_INFO *date_info);
```

Parameters

file_id	Input:	ID value for the file to read
date_info	Output	Pointer to information of date structure to store

Return Value

-1	Error
0	Normal completion

Remark

None

2.4.11 get_file_list_info**Description**

This function writes the file list stored at the directory path specified as an argument to a file list structure.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t get_file_list_info(uint8_t dir_path, FILE_LIST *file_list, uint32_t num_file_list, int32_t read_index);
```

Parameters

dir_path	Input	Pointer to directory path to read
file_list	Output	Pointer to file list to store This function stores '\0' to end of structure
num_file_list	Input	Max number of file list to read at one time
read_index	Input	Index of read starting

Return Value

-1	Error
0 and positive integer	Number of file

Remark

In case return value is smaller than num_file_list, it's the end of file list. In case return value is same value num_file_list, there is the data continuing. When this function needs continuing data, this function is called with 0 and positive integer with in read_index.

There are two cases. The argument "dir_path" has '/' termination and does not have. Please adjust for user file system.

2.4.12 get_file_size

Description

This function reads the file management information for the file corresponding to the ID value specified as an argument and returns the file size.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t get_file_size(int32_t file_id);
```

Parameters

file_id Input: ID value for the file to read

Return Value

-1 Error
0 and positive integer File size

Remark

None

2.4.13 make_dir

Description

This function makes the directory. The argument is specified in full path from root directory.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t make_dir(uint8_t *dir_path);
```

Parameters

dir_path Input Pointer to file path to make

Return Value

-1 Error
0 Normal completion

Remark

There are two cases. The argument "dir_path" has '/' termination and does not have. Please adjust for user file system.

2.4.14 remove_dir

Description

This function removes the directory. The argument is specified in full path from root directory.

Usage

```
#include <stdint.h>
#include "r_file_driver.h"
int32_t remove_dir(uint8_t *dir_path);
```

Parameters

dir_path	Input	Pointer to file path to remove
----------	-------	--------------------------------

Return Value

-1	Error
0	Normal completion

Remark

There are two cases. The argument "dir_path" has '/' termination and does not have. Please adjust for user file system.

2.5 Notes

This program uses `stdio.h`, `stdlib.h`, `string.h`, and `ctype.h`. Specify `stdio`, `stdlib`, `string`, and `ctype` as compiler options when compiling user programs.

The sample program that uses USB confirms the operation only in the little endian.

Website and Support

Renesas Electronics Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins

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

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

2. Processing at Power-on

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

- The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

Access to reserved addresses is prohibited.

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

4. Clock Signals

After applying a reset, only release the reset line after the operating clock signal has become stable.

When switching the clock signal during program execution, wait until the target clock signal has stabilized.

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

5. Differences between Products

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

- The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

- All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
 - Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
 - You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
 - Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
 - When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
 - Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
 - Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
Standard: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
High Quality: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
Specific: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
 - You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
 - Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
 - Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that include the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
 - This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
 - Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



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: +44-1628-585-100, Fax: +44-1628-585-900

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.

7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.

1 HarbourFront Avenue, #06-10, Keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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 Laviel' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141